

# TUSTEP · Handbuch und Referenz



Kuno Schälkle · Wilhelm Ott

# TUSTEP

Tübinger System  
von Textverarbeitungs-Programmen

Version 2020

Handbuch und Referenz

Tübingen 2020

Das Tübinger System von Textverarbeitungsprogrammen TUSTEP wurde seit 1966 am Zentrum für Datenverarbeitung der Universität Tübingen entwickelt. Es ist seit mehr als 40 Jahren im Einsatz; seinen heutigen Namen erhielt es 1978. TUSTEP wird ständig weiterentwickelt, seit 2003 auf Initiative der International Tustep User Group (ITUG) mit finanzieller Unterstützung durch akademische Partnereinrichtungen. Seit Juni 2011 steht TUSTEP (einschließlich seiner Quellen) als Open-Source-Produkt unter der »Revised BSD Licence«.

Dieses Handbuch enthält eine Beschreibung des Leistungsumfangs der Version 2020 von TUSTEP. Es ist als Nachschlagewerk und nicht zum Selbststudium für Anfänger gedacht.

Zur Einführung in TUSTEP wird die Teilnahme an einem Kurs empfohlen. Hinweise auf angebotene Kurse finden Sie auf der TUSTEP-homepage <http://www.tustep.uni-tuebingen.de>

Die International TUSTEP User Group ITUG (<http://www.itug.de>) bietet weitere Informationen zu TUSTEP und betreibt die »Tustep-Liste«, ein Diskussionsforum zu Themen rund um TUSTEP. Das Tustep-Wiki <http://tustep.wikispaces.com/TUSTEP-wiki> bietet Informationen und Materialien zu Tustep. »Ziel ist eine Sammlung von Problemen, Lösungen und praktischen Beispielen für alle, die mit TUSTEP arbeiten, für Anfänger ebenso wie für fortgeschrittene Anwender.«

Das Handbuch TUSTEP 2016 ist auch als gedrucktes Buch (2 Bände) erhältlich:  
Es kann für 59,-- EUR + Porto bezogen werden bei:

pagina GmbH  
Herrenberger Straße 51  
72070 Tübingen  
[www.pagina.gmbh](http://www.pagina.gmbh)  
Fax: +49 7071 987622

Gesetzt aus der ITCStone mit dem TUSTEP-Satzprogramm.  
Satz: Wilhelm Ott

ISBN der gedruckten Ausgabe: 978-3-938529-07-2

# Inhalt

Einleitung . . . . .	9
Installation . . . . .	17
Dateien . . . . .	23
Dateistruktur . . . . .	43
Daten-Transfer . . . . .	53
Datensicherung . . . . .	61
Systemumgebung . . . . .	71
TUSTEP-Aufruf . . . . .	77
Kommandos . . . . .	107
Allgemeines . . . . .	110
Eingabe . . . . .	113
Kommando-Manager . . . . .	117
#ABMELDE      Abmelden von Dateien . . . . .	119
#AENDERE      Ändern von Dateinamen und Projektnamen . . . . .	122
#ANMELDE      Anmelden von Dateien . . . . .	124
#BEENDE      Beenden/Unterbrechen einer TUSTEP-Sitzung . . . . .	127
#DATEI      Einrichten von Dateien und Projekten . . . . .	128
#DEFINIERE     Definieren einer Makro-Datei, Variablen u. a. . . . .	132
#DRUCKE      Druckausgabe . . . . .	142
#DUMPE      Analysieren von Dateien . . . . .	147
#DVORBEREITE    Daten zum Drucken vorbereiten . . . . .	148
#EDIERE      Edieren von Daten/Dateien . . . . .	149
#EINFUEGE     Einfügen von Textteilen . . . . .	156
#FAUFBEREITE    Formulare aufbereiten zum Drucken . . . . .	157
#FEHLERHALT    Fehlerhalt ein/ausschalten . . . . .	158
#FORMATIERE    Formatieren (autom. Umbruch) von Texten . . . . .	160
#HALT      Halt vor Ausführung noch anstehender Kommandos . . . . .	161
#HILFE      Online-Hilfe . . . . .	162
#HOLE      Holen von Daten . . . . .	165
#INFORMIERE    Informieren über Makro-Datei, Variablen u. a. . . . .	167
#KAUSFUEHRE    Korrekturanweisungen ausführen . . . . .	171
#KOPIERE      Kopieren, Auswählen, Modifizieren von Texten . . . . .	172
#LISTE      Listen von Dateinamen u. a. . . . .	173
#LOESCHE      Löschen von Daten/Dateien und Projekten . . . . .	182
#MAKRO      Zusammenstellen einer Kommandofolge . . . . .	186
#MBAUSGABE    Ausgabe in eine Band-Datei . . . . .	187
#MBEINGABE    Eingabe von einer Band-Datei . . . . .	189
#MBINFORMIERE    Informieren über eine Band-Datei . . . . .	191
#MBKOPIERE    Kopieren einer Band-Datei . . . . .	193

#MBLABEL	Labeln einer Band-Datei . . . . .	197
#MBTEST	Testen einer Band-Datei . . . . .	199
#MISCHE	Mischen von Daten/Dateien . . . . .	201
#MODI	Modi merken/zurücksetzen . . . . .	203
#NORMIERE	Normieren/Beenden von TUSTEP . . . . .	205
#NUMMERIERE	Neu-Nummerieren von Verweisen . . . . .	206
#PARAMETER	Parameter-Modi einstellen . . . . .	207
#PROTOKOLL	Zweitprotokoll ein/ausschalten u. a. . . . .	209
#RAUFBEREITE	Register nach dem Sortieren aufbereiten . . . . .	215
#RETTE	Retten von Daten . . . . .	216
#RVORBEREITE	Register zum Sortieren vorbereiten . . . . .	220
#SATZ	Satzherstellung . . . . .	221
#SIGNAL	Ausgeben von Signaltönen . . . . .	222
#SORTIERE	Sortieren von Daten/Dateien . . . . .	224
#SPRUEFE	Sortierschlüssel prüfen . . . . .	227
#SUCHE	Durchsuchen von Texten . . . . .	228
#SVORBEREITE	Daten zum Sortieren vorbereiten . . . . .	229
#TESTE	Testen/Vergleichen einer Datei . . . . .	230
#TITEL	Titel einer Datei definieren/übertragen . . . . .	231
#TUE	Ausführen einer Kommandofolge . . . . .	233
#UMWANDLE	Umwandeln/Verschlüsseln von Daten/Dateien . . . . .	237
#VAUFBEREITE	Vergleichsergebnisse aufbereiten zum Drucken . . . . .	247
#VERGLEICHE	Vergleichen fortlaufender Texte . . . . .	248
#WAHLSCHALTER	Wahlschalter setzen/löschen . . . . .	249
#WISCHEN	Wischen (Überschreiben) von Daten ein/ausschalten . . . . .	250
#ZEIT	Zeitabfrage . . . . .	251
Editor	. . . . .	253
Makros	. . . . .	403
Suche	. . . . .	667
{ }-Parameter	. . . . .	701
<>-Parameter	. . . . .	735
Zeichenvorrat	. . . . .	759
Code-Tabellen	. . . . .	811
Parametergesteuerte Programme		
#DVORBEREITE	Daten zum Drucken vorbereiten . . . . .	845
#EINFUEGE	Einfügen von Textteilen . . . . .	859
#FAUFBEREITE	Formulare aufbereiten zum Drucken . . . . .	875
#FORMATIERE	Formatieren (autom. Umbruch) von Texten . . . . .	893
#KAUSFUEHRE	Korrekturanweisungen ausführen . . . . .	917
#KOPIERE	Kopieren, Auswählen, Modifizieren . . . . .	929
#NUMMERIERE	Neu-Nummerieren von Verweisen . . . . .	997
#RAUFBEREITE	Register nach dem Sortieren aufbereiten . . . . .	1007
#RVORBEREITE	Register zum Sortieren vorbereiten . . . . .	1039

#SPRUEFE	Sortierschlüssel prüfen . . . . .	1073
#SVORBEREITE	Daten zum Sortieren vorbereiten . . . . .	1081
#VAUFBEREITE	Vergleichsergebnisse aufbereiten . . . . .	1105
#VERGLEICHE	Vergleichen fortlaufender Texte . . . . .	1117

## Programme und Makros zur Satzherstellung

#SATZ	Satzherstellung . . . . .	1143
#*AUMBRUCH	Umbrechen von Texten mit Apparaten . . . . .	1331
#*BOOKMARKS	Lesezeichen (bookmarks) für PDF-Dateien erzeugen . . . . .	1339
#*FPROT	Fehlerprotokoll aus SATZ-Protokoll extrahieren . . . . .	1342
#*FUNO	Fußnoten aus dem Text extrahieren . . . . .	1345
#*GRAFIK	EPS-Grafiken für *PSAUS aufbereiten . . . . .	1348
#*HIEROGR	Hieroglyphe in Grafikdatei einfügen . . . . .	1353
#*KERNPAR	Kerningtabellen in KOPIERE-Parameter unwandeln . . . . .	1355
#*MASKE	Masken für die SATZ-Ausgabe definieren . . . . .	1356
#*MONT	SATZ-Ausgabedateien zusammenmontieren . . . . .	1358
#*PRECOMPOSED	PS-Dateien nach OpenType-Konventionen umcodieren . . . . .	1359
#*PROT2QU	Quelldaten und Parameter aus Satzprotokoll extrahieren . . . . .	1361
#*PS4A4	PostScript-Datei ausdrucken: 4 Seiten/A4-Blatt . . . . .	1362
#*PSAUS	SATZ-Ausgabe auf PostScript-Geräten . . . . .	1364
#*PSAUSWAHL	Seiten aus PostScript-Datei auswählen . . . . .	1376
#*PSDICKTEN	Dickenliste für PostScript-Font drucken . . . . .	1377
#*PSDR	PostScript-Datei ausdrucken . . . . .	1379
#*PSFONT	Dickenwerte von PostScript-Fonts bereitstellen . . . . .	1381
#*PSFONTVOR	Vorbereiten einer AFM-Datei für *PSFONT . . . . .	1384
#*PSGLYPHS	Zeichenliste für PostScript-Font drucken . . . . .	1385
#*PSKERNFILE	Kerning-Information bereitstellen . . . . .	1387
#*PSMONT	PostScript-Dateien zusammenmontieren . . . . .	1388
#*REKLAM	mit Klammern codierte Registereinträge umwandeln . . . . .	1390
#*SILARCH	Silbentrennungs-Archiv erstellen . . . . .	1393
#*SILKOR	Silbentrennungen korrigieren . . . . .	1395
#*SILLIST	Silbentrennungs-Liste erstellen . . . . .	1397
#*SILMARKE	Silbentrennungs-Markierungen einfügen . . . . .	1400
#*SILMARKO	Silbentrennungs-Markierungen optimieren . . . . .	1402
#*SUMBRUCH	Umbrechen von mehrspaltigen Einschüben . . . . .	1406
#*TAGS	SGML/XML-Tags in Makros für SATZ umwandeln . . . . .	1408
#*TAGUEB	Makro-Auflösungen in *TAGS-Ergebnis übertragen . . . . .	1412
#*TEXGRAF	TeX-Seiten (PostScript) in Grafikdatei einfügen . . . . .	1413
#*XMLATTSOR	Attribute in XML-Tags sortieren . . . . .	1416
#*XMLZIEL	XML-Tags in Satz-Zieldatei wiederherstellen . . . . .	1417

Register . . . . .	1419
--------------------	------





# **Einleitung**

Was ist TUSTEP? . . . . .	11
Grundoperationen der Textdaten-Verarbeitung in TUSTEP . . . . .	13
Organisatorische Leistungen in TUSTEP . . . . .	15

## Was ist TUSTEP?

Das »Tübinger System von Textverarbeitungs-Programmen« TUSTEP wurde am Zentrum für Datenverarbeitung der Universität Tübingen in der Abteilung »Literarische und Dokumentarische Datenverarbeitung« mit dem Ziel entwickelt, ein leistungsfähiges Werkzeug zum wissenschaftlichen Umgang mit Textdaten zur Verfügung zu stellen.

Für die wichtigsten Grundoperationen der Textdaten-Verarbeitung wurden Programme bereitgestellt, deren Leistung vom Benutzer über Parameter spezifiziert und die in vielfältiger Weise für die Lösung verschiedenster Aufgabenstellungen kombiniert werden können.

Der Begriff *Textdaten*-Verarbeitung soll TUSTEP von dem, was heute üblicherweise unter Textverarbeitung verstanden wird, abgrenzen. Selbstverständlich gehören auch die für die Dokumenten-Erstellung notwendigen Funktionen: Eingabe, Korrektur, Formatieren, Drucken von Texten (auch von fremdsprachlichen Texten in nicht-lateinischen Alphabeten) zum Leistungsangebot von TUSTEP, da diese in allen Wissenschaftsbereichen zum Zweck der Dokumentation und Publikationsvorbereitung benötigt werden. TUSTEP wurde aber als Werkzeug vor allem für diejenigen Wissenschaften entwickelt, in denen Texte *Objekte* der Forschung sind: Philologien, Sprachwissenschaften, Literaturwissenschaften, historische Wissenschaften, Bibliothekswesen; Wissenschaften also, in denen nicht nur neue Texte als Produkt der eigenen wissenschaftlichen Arbeit erstellt und publiziert werden sollen, sondern in denen schon existierende, überlieferte, schriftlich fixierte oder zu fixierende Texte (einschließlich literarischer Texte und historischer Quellen) durch kritische Neuedition gesichert, sprachlich und stilistisch analysiert, inhaltlich erschlossen, bibliographisch erfasst werden müssen.

Dem tragen Grundoperationen der Textdaten-Verarbeitung (und entsprechende TUSTEP-Programme) Rechnung, die mit folgenden Schlagwörtern grob charakterisiert werden können: *Vergleichen* von verschiedenen Textfassungen; *Korrigieren* nicht nur interaktiv im Editor, sondern auch anhand vorbereiteter (u. U. automatisch erstellter) Korrekturanweisungen; *Zerlegen* von Texten in (frei definierbare) Elemente (z. B. Wortformen); *Sortieren* von Textelementen oder von längeren Texteinheiten nach einer Vielzahl von Alphabeten und anderer Kriterien; *Register erstellen* durch Zusammenfassen sortierter Textelemente; *Bearbeiten* von Textdaten durch vom Benutzer definierte Regeln zum Auswählen, Ersetzen, Umstellen, Ergänzen, Zusammenfassen, Vergleichen von Textteilen, durch Rechnen mit Zahlenwerten, die bereits im Text enthalten sind (z. B. Kalenderdaten) oder aus ihm gewonnen werden können (z. B. die Zahl der Wörter in einem Satz), und Ausgeben in verschiedenen Formaten, einschließlich solcher, die von anderer Software (z. B. zur statistischen Auswertung oder zur elektronischen Publikation) benötigt werden.

Aufgaben, die mit TUSTEP bearbeitet werden, reichen vom Schreiben einer Seminararbeit bis hin zum Erstellen von umfangreichen Bibliographien, Lexika, Indizes, Konkordanzen, Wörterbüchern, Editionen und natürlich auch von Monographien, jeweils einschließlich der automatischen Herstellung der Druckvorlagen für diese Werke in der vom Buchdruck gewohnten Qualität bzw. einschließlich der Bereit-

stellung der Daten in der Form (z. B. HTML, XML) und Codierung (z. B. Unicode), die für die elektronische Publikation erforderlich ist.

Neben den Programmen für die Grundoperationen der Textdaten-Verarbeitung enthält TUSTEP auch eine Reihe organisatorischer Leistungen, wie sie üblicherweise vom Betriebssystem eines Rechners bereitgestellt werden. Dies ermöglicht es, alle für die Textdaten-Verarbeitung notwendigen Funktionen, einschließlich der Datenhaltung und Datensicherung, auf Rechnern mit unterschiedlichen Betriebssystemen in gleicher Weise aufzurufen, und erspart damit dem Benutzer beim Wechsel auf einen Rechner mit einem anderen Betriebssystem nicht nur ein Umlernen, sondern erlaubt auch, bereits erstellte TUSTEP-Kommandofolgen unverändert zu übernehmen.

Eine für alle Rechner identische Benutzer-Oberfläche konnte nur dadurch erreicht werden, dass auf die Ausnutzung spezieller Eigenschaften einzelner Rechner und Betriebssysteme verzichtet wurde. Aus diesem Grund konnten insbesondere die graphischen Möglichkeiten der Bildschirme nicht voll ausgenutzt werden.

Die Leistungen von TUSTEP werden ständig verbessert und erweitert, damit auch für neue Aufgabenstellungen der wissenschaftlichen Textdaten-Verarbeitung Lösungsmöglichkeiten bereitstehen; dabei werden auch neue Hardware und neue Betriebssystem-Entwicklungen berücksichtigt. Zu den Neuerungen der letzten Jahre zählen ein CGI-Interface sowie Erweiterungen zur einfacheren Bearbeitung von Texten, die nach SGML / XML / TEI codiert sind.

TUSTEP verdankt viele seiner Leistungen der Anregung, Kritik und Mitarbeit von Benutzern aus fast allen geisteswissenschaftlichen Disziplinen, auch von außerhalb der Universität Tübingen. Die für TUSTEP Verantwortlichen sind auch weiterhin für Verbesserungsvorschläge dankbar.

## Grundoperationen der Textdaten-Verarbeitung in TUSTEP

### EDIEREN

Eingeben, Ändern, Ersetzen und Durchsuchen von Textdaten am Bildschirm [#EDIERE]

Eingeben, Ändern, Durchsuchen und automatisches Abprüfen von strukturierten Daten über selbst definierte Masken (auch mit Web-Browser) [#MAKRO]

Automatische Korrektur von Textdaten über vorbereitete Korrekturanweisungen [#KAUSFUEHRE]

### VERGLEICHEN

Vergleichen verschiedener Fassungen eines Textes; Protokollieren und Abspeichern der Unterschiede [#VERGLEICHE]

Zeilensynoptische Ausgabe der zu einem Grundtext festgestellten Textvarianten [#VAUFBEREITE]

### BEARBEITEN

Auswählen, Ersetzen, Umstellen, Ergänzen, Zusammenfassen, Vergleichen von Textteilen nach angegebenen Regeln und Bedingungen; Rechnen mit Zahlenwerten (einschließlich Kalenderdaten), die bereits im Text enthalten sind oder aus ihm gewonnen werden können; Ausgeben in verschiedenen Formaten (auch für die elektronische Publikation oder zur Weiterverarbeitung außerhalb von TUSTEP) [#MAKRO, #KOPIERE]

Ersetzen von Kürzeln durch Textteile (auch umfangreiche Textbausteine), die in einer eigenen Datei stehen [#EINFUEGE]

Verwalten und Aktualisieren von Querverweisen [#NUMMERIERE]

### REGISTER VORBEREITEN

Erstellen von Registerinträgen durch Zerlegen von Texteinheiten in ihre Bestandteile oder durch Extrahieren gekennzeichnete Textteile; ggf. Ergänzen und Ändern von Textteilen; Ergänzen der Referenz; Unterscheidung zwischen verschiedenen Eintragstypen; Aufbau von Sortierfeldern (für das anschließende Sortieren) nach frei wählbaren Sortierkriterien und Sortieralphabeten [#RVORBEREITE]

## SORTIERUNG VORBEREITEN

Bilden von Sortiereinheiten aus logisch zusammengehörenden Textteilen; Aufbau von Sortierfeldern (für das anschließende Sortieren) nach frei wählbaren Sortierkriterien (Auswahl und Reihenfolge bestimmter Textteile der Sortiereinheiten, Vorgabe von Sortierwerten für beliebige Zeichenfolgen) und Sortieralphabeten. [#SVORBEREITE]

## SORTIEREN

Ordnen von Datensätzen in eine aufsteigende oder fallende Reihenfolge, die durch die Sortierkriterien bestimmt ist, die in den Sortierfeldern enthalten sind [#SORTIERE]

Mischen bereits sortierter Daten [#MISCHE]

Prüfen auf korrekte Sortierung [#SPRUEFE]

## REGISTER AUFBEREITEN

Zusammenfassen sortierter, ggf. hierarchisch gegliederter Registereinträge (Texteinheiten); Ergänzen und Ersetzen von Textteilen und Referenzen; Unterscheidung zwischen verschiedenen Eintragsstypen; Berechnen absoluter und relativer Häufigkeiten [#RAUFBEREITE]

## DRUCK AUFBEREITEN

Aufbereiten von Textdaten zum Drucken

- in der Form, in der die Daten in der Datei stehen, wobei Steuerzeichen nicht interpretiert, sondern wie die anderen Zeichen ausgegeben werden [#DVORBEREITE]
- in frei wählbarem Format und frei wählbarer Anordnung (über im Text enthaltene Steuerzeichen), mit automatischer Silbentrennung und automatischem Zeilen- und Seitenumbruch einschließlich Randausgleich und Fußnoten [#FORMATIERE]
- als Formulare (auch Adressaufkleber und Bibliothekskärtchen) [#FAUFBEREITE]

## SETZEN

Typographisches Aufbereiten von Textdaten zur Ausgabe auf PostScript-Druckern oder (für professionellen Satz) auf PostScript-Satzbelichtern; dabei automatischer Zeilenumbruch (Block-, Flatter-, Tabellensatz) und automatischer Seitenumbruch mit lebenden Kolumnentiteln, Überschriften, Grundtext, Einschaltungen, Grafiken, Fußnoten, Marginalien, bis zu neun kritischen Apparaten; große Auswahl von Schriften und Sonderzeichen [#SATZ]

## Organisatorische Leistungen in TUSTEP

### DATENAUSTAUSCH

Import von Textdaten aus Dateien von anderen Programmen (Fremd-Dateien) in TUSTEP-Dateien [#UMWANDLE, #\*IMPORT]

Export von Textdaten aus TUSTEP-Dateien in Fremd-Dateien (z. B. ASCII-Dateien, XML-Dateien oder RTF-Dateien) [#UMWANDLE, #\*EXPORT]

### DATEIVERWALTUNG

Einrichten, Anmelden, Abmelden, Umbenennen und Löschen von Dateien [#DATEI, #ANMELDE, #ABMELDE, #AENDERE, #LOESCHE, #TITEL]

### ARCHIVIERUNG, SICHERUNG

Archivierung von Dateien (auch mehrerer Versionen der gleichen Datei) in einer »Band-Datei«, auch zum Daten-Transfer zwischen unterschiedlichen Rechnern; Information über den Inhalt der »Band-Datei« [#MBEINGABE, #MBAUSGABE, #MBKOPIERE, #MBINFORMIERE, #MBLABEL, #MBTEST, #\*MBUPDATE, #DATEI, #RETTE, #HOLE, #LOESCHE, #LISTE]

### ABLAUFSTEUERUNG

Ausführung/Steuerung von Kommandofolgen und Programmen; Einrichten und Ausführen eigener Kommandos (Makros) [#MAKRO, #TUE]





# Installation

---

Installation und Konfiguration . . . . .	19
TUSTEP für Windows auf einem USB-Speicher-Stick . . . . .	20
TUSTEP für macOS auf einem USB-Speicher-Stick . . . . .	21

## Installation und Konfiguration

Was bei der Installation von TUSTEP beachtet werden muss, ist in der »Installationsanleitung« beschrieben. Sie kann über die Internet-Adresse [www.tustep.uni-tuebingen.de](http://www.tustep.uni-tuebingen.de) heruntergeladen werden.

Falls Tasten oder Tastenkombinationen nicht die in dieser Beschreibung angegebene Wirkung haben, kann es daran liegen, dass die in der Installationsanleitung angegebenen Einstellungen nicht (alle) vorgenommen wurden oder dass TUSTEP nicht in der Weise aufgerufen wurde, wie in der Installationsanleitung und in dieser Beschreibung ab Seite 77 angegeben ist.

Die Installationsanleitung enthält auch noch Angaben zur Konfiguration von TUSTEP (z. B. wie die Schriftgröße eingestellt werden kann) und weitere betriebssystemabhängige Tipps und Tricks. Deshalb kann sie auch in TUSTEP mit dem Kommando `#*ZEBE, CONFIG` angezeigt werden.

## TUSTEP für Windows auf einem USB-Speicher-Stick

Mit dem Makro \*TUSTEP2STICK kann TUSTEP so auf einen USB-Speicher-Stick kopiert werden, dass TUSTEP auf anderen Windows-Rechnern, auf denen TUSTEP nicht installiert ist, aufgerufen werden kann.

Nach dem Aufruf des Makros mit dem Kommando #\*TUSTEP2STICK wird folgende Eingabemaske angezeigt:

TUSTEP auf einen USB-Stick kopieren

Laufwerksbuchstabe des USB-Sticks

Text für die Titelzeile

INI-Datei

Sonstige Dateien

Scratch-Dateien  auf Festplatte  
 auf USB-Stick

Eingabebeispiel:

Im ersten Feld muss der Laufwerksbuchstabe des Sticks angegeben werden.

Im zweiten Feld muss der Text eingetragen werden, der in der Kopfzeile des TUSTEP-Fensters als Titel angezeigt werden soll.

Im dritten Feld kann eine Datei angegeben werden, die als INI-Datei verwendet wird.

Falls auch eigene Dateien auf den USB-Speicher-Stick kopiert werden sollen, können diese im vierten Feld angegeben werden. Die Dateinamen müssen durch Apostroph getrennt sein.

Nachdem noch festgelegt wurde, ob die Scratch-Dateien auf der Festplatte oder auf dem USB-Speicher-Stick angelegt werden, kann durch Aktivieren der Schaltfläche »TUSTEP kopieren« TUSTEP auf den USB-Speicher-Stick kopiert werden.

Aufruf von TUSTEP auf dem USB-Speicher-Stick:

- Im Explorer den USB-Speicher-Stick öffnen,
- Ins Verzeichnis WIN-TUSTEP wechseln,
- Programm tustep.exe ausführen.

## TUSTEP für macOS auf einem USB-Speicher-Stick

Mit dem Makro \*TUSTEP2STICK kann TUSTEP so auf einen USB-Speicher-Stick kopiert werden, dass TUSTEP auf anderen macOS-Rechnern, auf denen TUSTEP nicht installiert ist, aufgerufen werden kann.

Achtung: Der USB-Speicher-Stick muss den Namen »TUSTEP« haben! Der Name eines USB-Speicher-Sticks kann unter macOS mit folgenden Schritten geändert werden:

- Finder öffnen.
- USB-Speicher-Stick mit der rechten Maustaste anklicken.
- Menü-Punkt »... umbenennen« auswählen.
- Neuen Namen eintragen.
- Mit der Eingabetaste bestätigen.

Nach dem Aufruf des Makros mit dem Kommando #\*TUSTEP2STICK wird folgende Eingabemaske angezeigt:

The screenshot shows a dialog box titled "TUSTEP auf einen USB-Stick kopieren". It has several input fields and buttons. The first field is "Name des USB-Sticks" with the value "TUSTEP". The second field is "Text für die Titelzeile" with the value "TUSTEP vom USB-Stick". There are three empty fields for "INI-Datei", "Sonstige Dateien", and "Scratch-Dateien". The "Scratch-Dateien" field has two radio buttons: "auf Festplatte" (selected) and "auf USB-Stick". At the bottom, there are two buttons: "TUSTEP kopieren" and "Beenden". Below the buttons, there is a text field with the example "Eingabebeispiel: TUSTEP vom USB-Stick".

Im ersten Feld der Eingabemaske steht der Name des Sticks. Er kann nicht geändert werden.

Im zweiten Feld muss der Text eingetragen werden, der in der Kopfzeile des TUSTEP-Fensters als Titel angezeigt werden soll.

Im dritten Feld kann eine Datei angegeben werden, die als INI-Datei verwendet wird.

Falls auch eigene Dateien auf den USB-Speicher-Stick kopiert werden sollen, können diese im vierten Feld angegeben werden. Die Dateinamen müssen durch Apostroph getrennt sein.

Nachdem noch festgelegt wurde, ob die Scratch-Dateien auf der Festplatte oder auf dem USB-Speicher-Stick angelegt werden, kann durch Aktivieren der Schaltfläche »TUSTEP kopieren« TUSTEP auf den USB-Speicher-Stick kopiert werden.

Da mit dem in macOS enthaltene Terminal-Programm »Terminal« in TUSTEP nicht mit der Maus gearbeitet werden kann, weil die Mausektionen nicht an TUSTEP weitergegeben werden, ist für TUSTEP das Terminal-Programm »iTerm« erforderlich.

Weil nicht vorauszusehen ist, ob auf dem Rechner, auf dem der USB-Speicher-Stick verwendet wird, »iTerm« installiert ist, wird auch noch das Terminal-Programm »iTerm« auf den Stick kopiert. Es muss dann aber nicht auf dem Rechner installiert werden.

Aufruf von TUSTEP auf dem USB-Speicher-Stick:

- Im Finder den Stick öffnen,
- Ins Verzeichnis MAC-TUSTEP wechseln,
- Programm TUSTEP.app ausführen.

Einschränkungen:

Damit ein Klick mit der rechten oder mittleren Maustaste (auch ohne gleichzeitiges Drücken der Shift- und/oder Ctrl-Taste) an TUSTEP weitergegeben wird, müssen auf dem Rechner, auf dem der USB-Speicher-Stick verwendet wird, für iTerm entsprechende Einstellungen vorgenommen werden. Sie sind in der Installationsanleitung, die mit dem Kommando `#*ZEBE, CONFIG` angezeigt wird, im Kapitel »Konfiguration von TUSTEP unter macOS« im Abschnitt »Maus« beschrieben. Alternativ kann im TUSTEP-Editor unten rechts die Schaltfläche »Mouse« verwendet werden. Dies ist auf Seite 345 beschrieben.

Die Funktionstasten sind unter macOS doppelt belegt. Sie dienen einerseits als Funktionstasten F1, F2 usw. wie dies unter Windows und Linux üblich ist. Andererseits kann z. B. mit F1 und F2 der Bildschirms dunkler bzw. heller gestellt werden. Die Voreinstellungen von macOS sind so, dass für den erstgenannten Fall zusätzlich die Taste Fn gedrückt werden muss. Wie die Einstellung so geändert werden kann, dass für den zweitgenannten Fall die Taste Fn gedrückt werden muss, ist in der Installationsanleitung im Kapitel »Konfiguration von TUSTEP unter macOS« im Abschnitt »Tastaturbelegung« beschrieben. In diesem Abschnitt ist auch angegeben, welche Tastenkombinationen nicht an TUSTEP weitergegeben werden, und wie dies geändert werden kann.

# Dateien

---

Projekt und Träger . . . . .	25
Projekt- und Dateinamen . . . . .	25
Ergänzen des Projektnamens . . . . .	26
Identifizieren einer Datei . . . . .	27
Dateien mit unzulässigem Namen . . . . .	29
Definierte Dateinamen . . . . .	29
Scratch-Dateien . . . . .	29
Standard-Dateien . . . . .	30
Dateiarten . . . . .	31
Fremd-Dateien . . . . .	31
ASCII-Dateien . . . . .	31
TUSTEP-Dateien . . . . .	31
Band-Dateien . . . . .	32
Einteilung der Daten in Sätze . . . . .	32
Satznummern . . . . .	32
Zugriff auf die Sätze . . . . .	34
Text-Dateien . . . . .	34
Programm-Dateien . . . . .	35
Segment-Dateien . . . . .	35
Makro-Dateien . . . . .	36
Editor-Datei . . . . .	36
Protokoll-Dateien . . . . .	36
Schwester-Dateien . . . . .	37
Einrichten von Dateien . . . . .	37
Anmelden von Dateien . . . . .	37
Abmelden von Dateien . . . . .	38
Löschen von Dateien . . . . .	38
Drucken von Dateien . . . . .	38
Retten von Dateien . . . . .	40
Reorganisieren von Dateien . . . . .	40
Sperren der Daten . . . . .	41
Sperren von Dateien . . . . .	41



## Projekt und Träger

Unter den ersten Betriebssystemen, auf denen TUSTEP implementiert war, wurde eine Datei durch den Namen des Eigentümers und den Namen der Datei identifiziert. Welche Namen als Eigentümer zugelassen waren, wurde vom Betriebssystem (Systemverwalter) vorgegeben. Meist wurde für jeden Arbeitsbereich, z. B. für jedes Forschungsprojekt, ein solcher Name vergeben. Bei eigenen Dateien brauchte der Name des Eigentümers nicht explizit angegeben zu werden, bei Dateien aus anderen »Projekten« musste er angegeben werden.

Die Dateien wurden meist auf Festplatten gespeichert. Sie konnten aber auch auf Wechselplatten gespeichert werden. Diese wurden vom Operateur bei Bedarf gewechselt; jede Wechselplatte musste deshalb einen Namen haben. Er wurde als Name des »Datenträgers« bezeichnet. Dieser Name musste bei der Anforderung einer Datei, die auf einer solchen Wechselplatte gespeichert war, mit angegeben werden.

Diese Vorgaben wurden von TUSTEP übernommen und führten auch zu der Namensgebung »Projekt« und »Träger«. Diese Begriffe werden weiterhin verwendet, um in TUSTEP die Angaben zu bezeichnen, die neben dem (eigentlichen) Dateinamen notwendig sind, um eine Datei eindeutig zu identifizieren.

Unter neueren Betriebssystemen wird eine Datei über einen Pfad (path) und den Dateinamen identifiziert. Beim Einrichten einer Datei wird ihr Name in ein Verzeichnis (directory) eingetragen. Ein Verzeichnis ist eine spezielle, vom Betriebssystem verwaltete Datei. Jedes Verzeichnis, außer dem Hauptverzeichnis (root directory), ist wiederum in ein übergeordnetes Verzeichnis eingetragen. Dem Hauptverzeichnis sind alle anderen Verzeichnisse direkt oder indirekt über andere Verzeichnisse untergeordnet. Der Pfad für eine Datei enthält die Namen der Verzeichnisse, die zum Auffinden der Datei notwendig sind, beginnend beim Hauptverzeichnis und endend mit dem Verzeichnis, in dem der Name der Datei eingetragen ist.

Das Verzeichnis (u. U. zusammen mit dem unmittelbar übergeordneten Verzeichnis), in dem der Name einer Datei eingetragen ist, entspricht in TUSTEP dem »Projekt«. Der Rest des Pfades wird über den »Träger« angegeben. Für Projekt und Träger sind die Voreinstellungen so gewählt, dass es in einfachen Fällen genügt, jeweils nur den Dateinamen anzugeben, um eine Datei anzusprechen.

## Projekt- und Dateinamen

Projektnamen können in TUSTEP die Form eines Namens haben oder sich aus zwei Namensteilen zusammensetzen; die beiden Namensteile werden durch einen Schrägstrich oder einen Backslash getrennt.

Besteht der Projektname aus einem Namen, darf dieser aus 1 bis 8 Zeichen (Buchstaben a bis z, Ziffern, Minuszeichen und »\_«) bestehen, muss mit einem Buchstaben beginnen und darf nicht mit Minuszeichen oder »\_« enden.

Besteht der Projektname aus zwei Namensteilen, darf der erste Namensteil aus 1 bis 8, der zweite aus 1 bis 3 Zeichen (Buchstaben a bis z, Ziffern, Minuszeichen und »\_«) bestehen. Der erste Namensteil muss mit einem Buchstaben beginnen, der zweite darf nicht mit Minuszeichen oder »\_« enden. Die beiden Namensteile müssen durch Schrägstrich oder Backslash getrennt sein.

Dateinamen dürfen in TUSTEP aus 1 bis 12 Zeichen (Buchstaben a bis z, Ziffern, Minuszeichen und »\_«) bestehen, müssen mit einem Buchstaben beginnen und dürfen nicht mit Minuszeichen oder »\_« enden. Solche Dateinamen können noch mit einer Namensweiterung (Extension) versehen werden. Diese wird durch einen Punkt getrennt an den Dateinamen angehängt. Sie darf in TUSTEP aus 1 bis 4 Zeichen (Buchstaben a bis z, Ziffern, Minuszeichen und »\_«) bestehen und darf nicht mit Minuszeichen oder »\_« enden.

Einschränkungen:

Auf einem Träger kann nicht gleichzeitig eine Datei und ein Projekt mit dem gleichen Namen existieren.

Der Projektname »TUSTEP« ist für TUSTEP reserviert und darf nicht verwendet werden.

Datei- und Projektnamen in der Form »TUSTEP.xxx« (wobei xxx für eine beliebige Buchstaben-Ziffern-Kombination steht) sind für TUSTEP reserviert und dürfen nicht verwendet werden. Dazu gibt es eine Ausnahme:

Der Name TUSTEP.INI ist für die INI-Datei reserviert. In sie können Makros (siehe Kapitel »Makros« ab Seite 403), die beim Initialisieren und/oder beim Fortsetzen einer TUSTEP-Sitzung (siehe Seite 89) ausgeführt werden sollen, und Editor-Definitionen eingetragen werden.

In der Datei \*TUSTEP.MCR werden die Makro-Aufrufe mit ihren Spezifikationswerten gespeichert, die mit dem Makro \*M (siehe Seite 408) aufgerufen bzw. gemerkt werden.

In der Datei \*TUSTEP.CDS werden die Code-Auswahlen gespeichert, die mit dem Makro \*DECO (siehe Seite 138) definiert werden.

## Ergänzen des Projektnamens

Ein Dateiname kann in TUSTEP um einen Projektnamen ergänzt und in der Form »Projektname\*Dateiname« angegeben werden. Wird der Projektname nicht oder nur teilweise angegeben, wird er automatisch ergänzt bzw. vervollständigt. Dazu wird entweder der ursprüngliche oder der aktuelle Projektname herangezogen.

Als ursprünglicher Projektname gilt derjenige, der beim Initialisieren der TUSTEP-Sitzung eingestellt wurde. Er ist durch die System-Variable TUSTEP\_PRJ vorgegeben.

Als aktueller Projektname gilt derjenige, der zuletzt mit dem Kommando #DEFINIERE (siehe Seite 132) eingestellt wurde. Solange damit noch kein Projektname eingestellt wurde, gilt der ursprüngliche Projektname zugleich als aktueller Projektname.

Der Projektname wird nach folgenden Regeln ergänzt:

- Wird der Projektname einschließlich des Sterns weggelassen, so wird der aktuelle Projektname ergänzt.
- Wird nur der Projektname weggelassen, der Stern vor dem Dateinamen jedoch mit angegeben, so wird der ursprüngliche Projektname ergänzt.

- Wird von einem zweiteiligen Projektnamen nur der erste Namensteil (nicht aber der nachfolgende Schrägstrich bzw. Backslash) weggelassen, so wird der erste Namensteil des aktuellen Projektnamens ergänzt.
- Wird an Stelle eines Projektnamens vor dem Stern ein »-« angegeben, so wird kein Projektname ergänzt.

## Identifizieren einer Datei

Eine Datei kann dadurch identifiziert werden, dass der Dateibezeichnung (Projekt- und Dateiname), die in TUSTEP verwendet werden soll, die Dateibezeichnung (Pfad mit Dateiname), die auf Betriebssystemebene erforderlich ist, zugeordnet wird (siehe »Definierte Dateinamen« Seite 29).

In der Regel wird in TUSTEP eine Datei durch drei Angaben identifiziert:

1. Träger: Über diese Angabe wird der Anfang des Pfades bis vor das letzte oder vorletzte Verzeichnis (directory) für die Datei festgelegt.
2. Projekt: Damit wird der Rest des Pfades (der Teil, der noch nicht durch den Träger definiert ist) für die Datei angegeben. In der Regel ist dies der Name des Verzeichnisses, in dem die Datei eingetragen ist. Es können aber auch die Namen der beiden letzten Verzeichnisse im Pfad sein, falls sich daraus ein zulässiger Projektname ergibt (siehe »Projekt- und Dateinamen« Seite 25).
3. Datei: Name der Datei ohne Pfadangaben.

Für den Träger muss beim Definieren der TUSTEP-Sitzung mit dem Makro \*`DESI` eine System-Variable mit entsprechendem Inhalt definiert werden. Dabei ist darauf zu achten, dass das entsprechende Verzeichnis jeweils schon existiert; nur bei den beiden System-Variablen `TUSTEP_DSK` und `TUSTEP_SCR` werden noch nicht existierende Verzeichnisse nach Rückfrage eingerichtet.

Wenn der Inhalt einer System-Variablen, die als Träger in TUSTEP verwendet wird, mit einer Tilde beginnt, wird diese Tilde durch den Inhalt der System-Variablen `HOME` ersetzt. Unter Linux und macOS wird diese vom Betriebssystem definiert und enthält z. B. `/home/userid` bzw. `/Users/userid`, wobei an Stelle von `userid` die jeweilige Benutzerkennung steht. Unter Windows wird die System-Variable `HOME` beim Start von TUSTEP definiert und enthält den Inhalt der beiden System-Variablen `HOMEDRIVE` und `HOMEPATH`, die vom Betriebssystem definiert werden; sie enthält z. B. `C:\Users\userid`, wobei an Stelle von `userid` die jeweilige Benutzerkennung steht.

Die Angabe des Trägers ist nur beim Einrichten und beim Anmelden einer Datei erforderlich; sie kann jedoch entfallen, falls der voreingestellte Träger (für permanente Dateien `TUSTEP_DSK`, für temporäre Dateien `TUSTEP_SCR`) verwendet werden soll.

Beim Anmelden einer Datei (beim Einrichten wird eine Datei automatisch auch angemeldet) wird jeweils der zugehörige Träger gemerkt, so dass er bei allen anderen Zugriffen auf die Datei nicht mehr angegeben werden muss (und auch gar nicht angegeben werden kann). Daraus ergibt sich folgende Einschränkung: Soll eine Datei eingerichtet/angemeldet werden, bei der Projekt- und Dateiname, nicht aber der

Träger mit einer bereits angemeldeten Datei übereinstimmt, so muss diese angemeldete Datei zuerst abgemeldet werden.

Die Angabe des Projekts ist nur notwendig, wenn das Projekt nicht mit dem aktuellen Projekt (siehe »Ergänzen des Projektnamens« Seite 26) übereinstimmt. Das Projekt muss ggf. dem Dateinamen vorangestellt und durch einen Stern von ihm getrennt werden.

Soll z. B. eine Datei angesprochen werden, die unter Windows die Dateibezeichnung

```
C:\Users\userid\Projekte\projekt\name
```

bzw. unter Linux die Dateibezeichnung

```
/home/userid/Projekte/projekt/name
```

bzw. unter macOS die Dateibezeichnung

```
/Users/userid/Projekte/projekt/name
```

hat, so wird diese Bezeichnung im Regelfall wie folgt aufgeteilt:

```
Träger:    C:\Users\userid\Projekte bzw.
           /home/userid/Projekte bzw.
           /Users/userid/Projekte
Projekt:   projekt
Datei:    name
```

Eine andere mögliche Aufteilung dieser Dateibezeichnung wäre:

```
Träger:    C:\Users\userid\Projekte\projekt bzw.
           /home/userid/Projekte/projekt bzw.
           /Users/userid/Projekte/projekt
Projekt:   -
Datei:    name
```

Diese Art der Aufteilung der Dateibezeichnung, bei der der komplette Pfad mit dem Träger angegeben wird, ist immer dann erforderlich, wenn der Name des Verzeichnisses, in dem die Datei eingetragen ist (im obigen Beispiel das Verzeichnis »projekt«), nicht den Konventionen eines Projektnamens in TUSTEP entspricht (siehe »Projekt- und Dateinamen« Seite 25). Wenn der Name der Datei nicht den Konventionen eines Dateinamens in TUSTEP entspricht, muss ein zusätzlicher Dateiname (Alias-Name) definiert werden, mit dem die Datei in TUSTEP angesprochen werden kann (siehe »Definierte Dateinamen« Seite 29).

Unter Windows kann beim Einrichten bzw. Anmelden einer Datei als Träger auch ein Laufwerksbuchstabe (z. B. eines USB-Speicher-Sticks) angegeben werden. Soll z. B. eine Datei angesprochen werden, die die Dateibezeichnung

```
E:\name bzw. E:\projekt\name
```

hat, so wird diese Bezeichnung wie folgt aufgeteilt:

```
Träger:    E
Projekt:   - bzw. projekt
Datei:    name
```

## Dateien mit unzulässigem Namen

In TUSTEP ist die Namensgebung für Dateien einerseits aus Kompatibilitätsgründen zu den verschiedenen Betriebssystemen und andererseits aus syntaktischen Gründen bei der Eingabe von Kommandos und Editoranweisungen eingeschränkt.

Dies kann dazu führen, dass eine nicht von TUSTEP eingerichtete Datei auf Grund ihres Namens nicht, wie im vorangehenden Abschnitt beschrieben, angesprochen werden kann. Dies betrifft in der Regel solche Dateien, deren Namen Sonderzeichen oder Umlaute enthalten. Unter Linux betrifft es auch Dateien, deren Namen Großbuchstaben enthalten, da in TUSTEP alle Buchstaben in Dateinamen (gleichgültig, ob sie als Groß- oder Kleinbuchstaben angegeben werden) beim Umsetzen des Dateinamens für das Betriebssystem als Kleinbuchstaben gewertet werden.

Abhilfe kann dadurch geschaffen werden, dass für die Datei, die einen für TUSTEP unzulässigen Namen hat, entweder auf Betriebssystemebene ein Alias-Name (link) definiert wird, oder in TUSTEP ein zusätzlicher Dateiname definiert wird (siehe folgenden Abschnitt), mit dem sie innerhalb von TUSTEP angesprochen werden kann.

## Definierte Dateinamen

Um auch auf Dateien zugreifen zu können, für die kein geeigneter Träger (System-Variable) definiert ist (siehe »Identifizieren einer Datei« Seite 27), oder die einen Dateinamen haben, der nicht den Konventionen eines Dateinamens in TUSTEP entspricht (siehe »Projekt- und Dateinamen« Seite 25), können für Dateien zusätzliche Namen (Alias-Namen) definiert werden, mit denen sie in TUSTEP angesprochen werden können. Dazu kann das Kommando `#DEFINIERE` (siehe Seite 132) oder die Makroanweisung `DEFINE` (siehe Seite 423) verwendet werden. Dabei wird dem definierten Dateinamen der vollständige Pfad einschließlich des Namens der Datei, wie er auf Betriebssystemebene erforderlich ist, zugeordnet.

Wird eine Datei mit einem solchen »definierten Dateinamen« (z. B. mit dem Kommando `#DATEI` oder der Makrofunktion `CREATE`) eingerichtet bzw. (z. B. mit dem Kommando `#ANMELDE` oder mit der Makrofunktion `OPEN`) angemeldet, so muss als Träger anstatt einer System-Variablen ein Minuszeichen angegeben werden. Beim Einrichten einer solchen Datei ist darauf zu achten, dass jedes im Pfad enthaltene Verzeichnis schon existiert. Falls eines nicht existiert, muss es zuvor eingerichtet werden.

## Scratch-Dateien

In TUSTEP wird zwischen permanenten Dateien und temporären Dateien unterschieden. Permanente Dateien bleiben erhalten, bis sie explizit gelöscht werden. Temporäre Dateien werden am Ende einer TUSTEP-Sitzung automatisch gelöscht, wenn sie nicht schon vorher explizit gelöscht wurden.

Temporäre Dateien werden auch Scratch-Dateien genannt; sie werden nur zum Speichern von Zwischenergebnissen verwendet, da Scratch-Dateien jeweils nur in der

TUSTEP-Sitzung zur Verfügung stehen, in der sie eingerichtet wurden. Die Verwendung von Scratch-Dateien bietet die Möglichkeit, Dateien mit Zwischenergebnissen durch Beenden der TUSTEP-Sitzung automatisch zu löschen. Außerdem werden Namenskonflikte mit anderen Dateien vermieden, wenn mehrere Sitzungen parallel existieren.

Die Namensgebung in TUSTEP ist für temporäre Dateien gleich wie für permanente Dateien, aber die Dateien erhalten aus der Sicht des Betriebssystems nicht den innerhalb von TUSTEP angegebenen Namen.

Scratch-Dateien werden unabhängig davon, welches Projekt mit dem Dateinamen angegeben wird, immer im ursprünglichen Projekt eingerichtet. Das ursprüngliche Projekt ist mit der System-Variablen `TUSTEP_PRJ` vorgegeben.

Außerdem werden die Namen der Scratch-Dateien durch einen Standardnamen ersetzt, damit keine Namenskonflikte auftreten und die Scratch-Dateien auch außerhalb von TUSTEP noch als solche zu erkennen sind. Dieser Name hat die Form `mmmmnnnn.tsf`, wobei `mmmm` die Nummer der TUSTEP-Sitzung und `nnnnn` eine laufende Nummer der Scratch-Dateien ist. Dadurch werden auch Namenskonflikte vermieden, wenn eine temporäre Datei mit einem Namen angelegt wird, der (zufällig) mit dem Namen einer nicht angemeldeten, permanenten Datei übereinstimmt.

Sollen z. B. die Scratch-Dateien unter dem Verzeichnis `/tmp` angelegt werden, so muss die System-Variable `TUSTEP_SCR` mit dem Wert `/tmp` definiert werden. Scratch-Dateien werden dann im Verzeichnis `/tmp/projekt` angelegt, falls die System-Variable `TUSTEP_PRJ` mit `projekt` belegt ist.

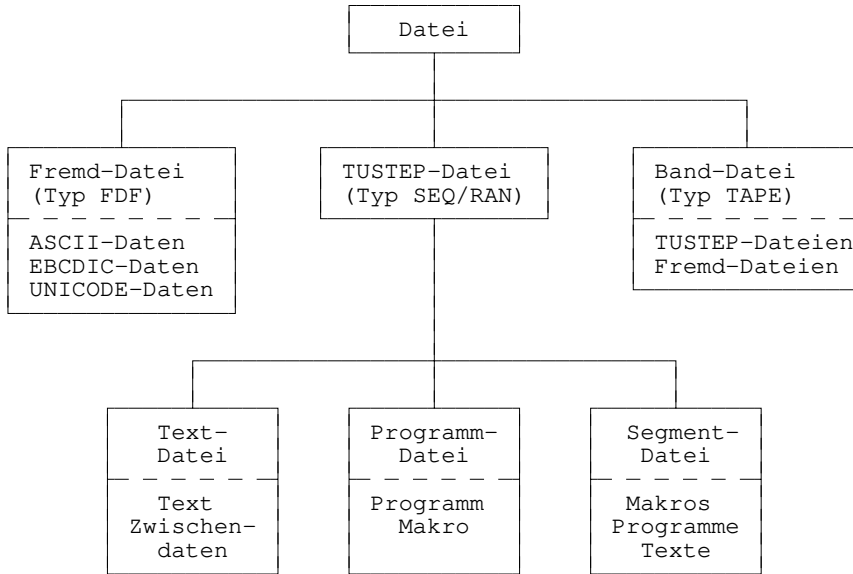
## Standard-Dateien

Außer den Dateien, die explizit angemeldet oder eingerichtet werden, stehen noch folgende Standard-Dateien zur Verfügung:

- Standard-Daten-Datei
- Standard-Editor-Datei
- Standard-Protokoll-Datei
- Standard-Text-Datei

Um Konflikte beim Zugriff auf diese Dateien zu vermeiden, sollten diese Dateien grundsätzlich nur mit dem Dateinamen `-STD-` angesprochen werden; sie werden bei Bedarf automatisch eingerichtet. Welche dieser Dateien wann verwendet werden kann, ist bei den jeweiligen TUSTEP-Kommandos beschrieben.

## Dateiarten



### Fremd-Dateien

Um Daten von anderen Programmen übernehmen bzw. an andere Programme weitergeben zu können, kennt TUSTEP Fremd-Dateien. Die Datenstruktur dieser Dateien ist vom jeweiligen Programm abhängig. Solche Dateien können auch mit dem Kommando #DATEI (siehe Seite 128) oder der Makrofunktion CREATE (siehe Seite 479) eingerichtet werden; sie haben den Typ FDF.

### ASCII-Dateien

ASCII-Dateien sind Fremd-Dateien, deren Daten in der Regel im internationalen ASCII-Code (siehe Tabelle Seite 815) codiert sind; sie können in TUSTEP jedoch auch in ISO-8859-1 (siehe Tabelle Seite 826) codiert sein. Als Trennzeichen zwischen den Datensätzen dienen unter Windows die Codes CR LF (= 0D0A) und unter Linux und macOS der Code LF (= 0A).

### TUSTEP-Dateien

Die TUSTEP-Programme erwarten, dass die zu verarbeitenden Daten in TUSTEP-Dateien stehen. Das sind Dateien mit TUSTEP-Struktur, die mit dem Kommando #DATEI (siehe Seite 128) oder der Makrofunktion CREATE (siehe Seite 479) eingerichtet wurden und vom Typ SEQ oder RAN sind.

Es gibt drei Arten von TUSTEP-Dateien: Text-, Programm- und Segment-Dateien. Sie

werden alle in gleicher Weise eingerichtet. Sie unterscheiden sich nur in der Art, in der sie verwendet werden, und in der Art der Daten, die sie enthalten.

## Band-Dateien

Band-Dateien sind Dateien auf Festplatten, USB-Speicher-Sticks oder anderen Datenträgern und haben innerhalb von TUSTEP die gleichen Eigenschaften wie Magnetbänder; sie dienen zum Daten-Transfer auf andere Rechner und zur Datensicherung. Band-Dateien können sowohl mit den Kommandos eingerichtet, bearbeitet und gelöscht werden, die für Magnetbänder vorgesehen waren (diese Kommandos beginnen alle mit `#MB...` und sind ab Seite 187 beschrieben), als auch mit dem Kommando `#DATEI` (siehe Seite 128) eingerichtet, mit `#RETTE` (siehe Seite 216) beschrieben, mit `#HOLE` (siehe Seite 165) gelesen und mit `#LOESCHE` (siehe Seite 182) gelöscht werden.

## Einteilung der Daten in Sätze

Die Daten in einer TUSTEP-Datei sind in Sätze (records) unterteilt. Enthält eine Datei z. B. den Text eines Buches, so kann ein Satz der Datei jeweils eine Zeile des Textes enthalten. Die Einteilung eines Textes in Sätze ist jedoch frei wählbar; sie kann z. B. nach Gesichtspunkten der Übersichtlichkeit gewählt werden. Die einzige Einschränkung betrifft die Satzlänge. Soll mit dem Editor ein Satz geändert werden, so darf dieser nicht länger als 8000 Zeichen sein bzw. auf Grund der Änderung nicht länger als 8000 Zeichen werden. In allen anderen Fällen dürfen die Sätze bis zu 64000 Zeichen lang sein.

## Satznummern

Die Satznummer wird von TUSTEP automatisch jedem Satz mitgegeben. Sie ist in eine Seiten- und eine Zeilennummer unterteilt und ermöglicht so auf einfache Weise eine Gliederung/Einteilung des Textes analog zu derjenigen in gedruckten Medien. Enthält eine Datei z. B. den Text eines Buches, wobei jeder Satz eine Zeile enthält, so können die Satznummern so gewählt werden, dass die Seitennummer der Sätze in der Datei mit der Seitennummer im Buch übereinstimmt und die Zeilennummer der Sätze einer fortlaufenden Nummer der Zeilen auf der jeweiligen Seite entspricht. Dadurch kann z. B. im Editor jede Textstelle über die Stellenangabe der gedruckten Vorlage direkt angesprochen werden. Jede Zeilennummer kann noch durch eine Unterscheidungsnummer ergänzt werden. Die Unterscheidungsnummer wird benötigt, damit Zeilen, die zwischen fortlaufend nummerierte Zeilen eingeschoben werden sollen, ebenfalls eine eindeutige Nummer erhalten können, ohne dass die Nummern der nachfolgenden Sätze verändert werden müssen. Ein eingefügter Satz erhält in der Regel die gleiche Seiten- und Zeilennummer wie derjenige, nach dem er eingefügt wird, aber eine andere Unterscheidungsnummer.

Wird in einer Datei ein Programm oder ein Makro gespeichert, so ist eine Einteilung in Seiten nicht sinnvoll. Deshalb wird dafür eine Nummerierungsart verwendet, bei der die Seitennummer immer Null ist und als nicht vorhanden gilt. Diese Numme-



rierungsart wird »Nummerierung im Programmmodus« genannt. Sie kann jedoch auch für kurze Texte verwendet werden, bei denen eine Seitennummer überflüssig ist. Die zuvor beschriebene Nummerierungsart mit Seitennummer wird »Nummerierung im Textmodus« genannt.

Bei Dateien vom Typ RAN müssen die Satznummern alle aufsteigend und voneinander verschieden sein. Wird beim Beschreiben einer Datei vom Typ RAN nicht darauf geachtet, bricht das Programm mit der Fehlermeldung »Satznummernkonflikt« ab.

Bei Dateien vom Typ SEQ gilt diese Bedingung für die Satznummern im Prinzip nicht. Sie muss jedoch erfüllt sein, wenn

- eine Datei mit dem Editor bearbeitet werden soll,
- eine Datei als Segment-Datei verwendet werden soll,
- Daten in der Datei über Satznummern ausgewählt werden sollen.

Nummerierung im Textmodus:

Diese Art der Nummerierung wird in TUSTEP für Text-Dateien verwendet.

Bei der Nummerierung im Textmodus besteht die Satznummer aus Seitennummer, Zeilennummer und Unterscheidungsnummer. Seitennummern sind maximal 6-stellig, Zeilen- und Unterscheidungsnummern sind je maximal 3-stellig. Die Schreibweise einer Satznummer ist:

Seitennummer.Zeilennummer/Unterscheidungsnummer

Die Unterscheidungsnummer ist in der Regel 0. In diesem Fall wird sie, einschließlich des davor stehenden Schrägstrichs, weggelassen. Bei der Unterscheidungsnummer müssen führende Nullen geschrieben werden, abschließende Nullen können weggelassen werden. Die Satznummern 1.2/3, 1.2/30 und 1.2/300 sind also gleichbedeutend.

In der Datei wird die Satznummer in zwei Zahlenwerten gespeichert. Der erste Wert gibt die Seitenzahl an, der zweite die Zeilen- und die Unterscheidungsnummer. Für den zweiten Zahlenwert wird die Zeilennummer mit 1000 multipliziert und die Unterscheidungsnummer aufaddiert; ein- und zweistellige Unterscheidungsnummern werden dazu logisch durch Anhängen von Nullen auf drei Stellen erweitert. Der Zeilennummer 2/3 entspricht also der Zahlenwert 2300.

Nummerierung im Programmmodus:

Diese Art der Nummerierung wird in TUSTEP für Programm-Dateien verwendet; der Editor verwendet sie standardmäßig, wenn die zu bearbeitende Datei beim Aufruf des Editors leer ist.

Bei der Nummerierung im Programmmodus besteht die Satznummer aus Zeilennummer und Unterscheidungsnummer. Zeilennummern sind maximal 4-stellig, Unterscheidungsnummern sind maximal 2-stellig. Die Schreibweise einer Satznummer ist:

Zeilennummer/Unterscheidungsnummer

Die Unterscheidungsnummer ist in der Regel 0. In diesem Fall wird sie, einschließ-

lich des davor stehenden Schrägstrichs, weggelassen. Bei der Unterscheidungsnummer müssen führende Nullen geschrieben werden, abschließende Nullen können weggelassen werden. Die Satznummern 1/2 und 1/20 sind also gleichbedeutend.

In der Datei wird die Satznummer in zwei Zahlenwerten gespeichert. Der erste Wert ist immer Null (0), der zweite gibt die Zeilen- und die Unterscheidungsnummer an. Für den zweiten Zahlenwert wird die Zeilennummer mit 100 multipliziert und die Unterscheidungsnummer aufaddiert; einstellige Unterscheidungsnummern werden dazu logisch durch Anhängen einer Null auf zwei Stellen erweitert. Der Zeilennummer 2/3 entspricht also der Zahlenwert 230.

## Zugriff auf die Sätze

In TUSTEP-Dateien können die einzelnen Sätze in ihrer logischen Reihenfolge sowohl vor- als auch rückwärts verarbeitet werden.

Soll auf einen Satz mit einer bestimmten Satznummer zugegriffen werden, so muss intern der Reihe nach jeder einzelne Satz daraufhin geprüft werden, ob er die geforderte Satznummer hat. Dies kann dadurch eventuell verkürzt werden, dass zuerst auf Grund der Satznummer geprüft wird, ob der geforderte Satz vor oder nach dem zuletzt verarbeiteten Satz (von dem die Position ja noch bekannt ist) steht. Steht der geforderte Satz danach, kann von der aktuellen Position aus vorwärts, andernfalls rückwärts gelesen werden.

Dieses Verfahren wird bei Dateien vom Typ SEQ angewandt. Es erscheint zeitaufwändig, ist aber für Dateien bis zu einer Größe von einigen Megabyte tragbar. Bei Dateien vom Typ RAN wird zusätzlich automatisch eine Tabelle verwaltet, die Verweise auf die Stelle in der Datei enthält, an der die Nummerierung der Sätze mit einer neuen Seitennummer beginnt. Dadurch kann der Zugriff auf einen Satz mit einer bestimmten Satznummer erheblich beschleunigt werden. Technische Einzelheiten dazu sind im Kapitel »Dateistruktur« ab Seite 51 beschrieben.

Dateien vom Typ RAN sind nur sinnvoll, wenn die Sätze im Textmodus (siehe Abschnitt »Satznummern« ab Seite 32) nummeriert sind und die Satznummern alle aufsteigend und voneinander verschieden sind; wenn die Sätze im Programmmodus nummeriert sind, haben alle Sätze die Seitennummer 0 (Null).

## Text-Dateien

Eine Text-Datei ist eine TUSTEP-Datei, die Text oder Zwischendaten (z. B. Sortierschlüssel) enthält.

Enthält eine Text-Datei Text, so sind die Sätze normalerweise im Textmodus nummeriert (siehe Seite 33). Wenn es sich um einen Text handelt, der in einer Segment-Datei abgespeichert werden soll (z. B. eine Adresse oder ein Textbaustein), müssen die Sätze im Programmmodus nummeriert sein (siehe Seite 33).

Enthält eine Text-Datei Zwischendaten, so ist die Nummerierung der Sätze in geeigneter Weise von TUSTEP vorgenommen worden und wird bei Bedarf entsprechend interpretiert. Das Bearbeiten einer solchen Datei mit dem Editor ist meist nicht möglich und mit Ausnahme von Protokoll-Dateien auch nicht sinnvoll.

## Programm-Dateien

Eine Programm-Datei ist eine TUSTEP-Datei, die ein Programm (= eine Kommandofolge) oder ein Makro enthält, und deren Sätze im Programmmodus nummeriert sind (siehe Seite 33).

## Segment-Dateien

Eine Segment-Datei ist eine TUSTEP-Datei, die in Segmente unterteilt ist. Ein Segment kann ein Programm, ein Makro oder Text enthalten.

Jedes Segment hat einen Namen, mit dem es angesprochen werden kann; er kann aus 1 bis 12 Zeichen (Buchstaben, Ziffern und »\_«) bestehen, muss mit einem Buchstaben beginnen und darf nicht mit »\_« enden.

Am Anfang einer Segment-Datei steht ein Inhaltsverzeichnis mit den Namen der in der Datei enthaltenen Segmente. Dieses Inhaltsverzeichnis wird automatisch angelegt bzw. erweitert, wenn im Editor mit der Rette-Anweisung die Daten der Editor-Datei oder mit dem Kommando #RETTE die Daten der angegebenen Datei als neues Segment mit einem Segmentnamen in eine leere TUSTEP-Datei bzw. in eine Segment-Datei abgespeichert werden.

Eine »normale« Segment-Datei kann bis zu 9999 Segmente aufnehmen. Sie kann jedoch mit dem Kommando #RETTE, *segmentdatei, -std-, +;+, +* (siehe Seite 216) in eine Mega-Segment-Datei konvertiert werden. Eine Mega-Segment-Datei kann bis zu 999999 Segmente aufnehmen; weitere Unterschiede zur normalen Segment-Datei gibt es nicht. Beim Einrichten einer Segment-Datei sollte als Dateityp RAN angegeben werden, damit schnell auf die einzelnen Segmente zugegriffen werden kann.

Eine Segment-Datei sollte nicht mit dem Editor ediert werden, sondern nur mit Hilfe der Hole- und der Rette-Anweisung (siehe Anweisung h Seite 281 und Anweisung r Seite 282) oder mit Hilfe der Kommandos #HOLE und #RETTE (siehe Seite 165 und 216) erstellt und bearbeitet werden. Da die Satznummern als Verweise dienen, dürfen zumindest die Sätze einer solchen Datei nicht umnummeriert werden.

Mit dem Kommando #RETTE (siehe Seite 216) können die Sätze einer Segment-Datei neu nummeriert werden. Außerdem bietet das Kommando die Möglichkeit, für eine Segment-Datei das Inhaltsverzeichnis neu zu erstellen; dies ist z. B. notwendig, nachdem zwei Segment-Dateien in eine Datei zusammenkopiert wurden.

Sollen die einzelnen Segmente einer Segment-Datei alphabetisch nach ihrem Namen sortiert werden, so steht dafür das Makro \*SESO zur Verfügung. Weitere Informationen über das Makro werden mit dem Kommando #INFORMIERE, \*SESO ausgegeben.

## Makro-Dateien

Eine Makro-Datei ist eine Segment-Datei, deren einzelne Segmente jeweils ein Makro (= Definition eines eigenen Kommandos; siehe Kapitel »Makro« ab Seite 403) enthalten. Der Name eines Segments ist zugleich der Name des Makros. Damit diese Makros angesprochen werden können, muss die Makro-Datei mit dem Kommando #DEFINIERE (siehe Seite 132) als solche definiert werden.

## Editor-Datei

Die Editor-Datei ist die TUSTEP-Datei, die gerade mit dem Editor bearbeitet wird; d. i. die Datei, die zuletzt beim Aufruf des Editors zur Spezifikation DATEI angegeben wurde (siehe Kommando #EDIERE Seite 149), bzw. die Datei, die zuletzt mit der Datei-Anweisung im Editor eingestellt wurde (siehe Anweisung d Seite 284).

Die Editor-Datei sollte nicht mit der Standard-Editor-Datei (siehe Seite 30) verwechselt werden, die automatisch zur Verfügung gestellt wird und die dann Editor-Datei ist, wenn sie mit dem Editor bearbeitet wird.

## Protokoll-Dateien

Eine Protokoll-Datei ist eine TUSTEP-Datei, die zum Drucken aufbereitete Daten enthält. In der Regel ergibt sich eine solche Datei, indem sie bei TUSTEP-Kommandos zur Spezifikation PROTOKOLL angegeben wird. Sie kann mit dem Kommando #DRUCKE (siehe Seite 142) ausgedruckt werden.

Wesentliches Merkmal einer Protokoll-Datei ist, dass das erste Zeichen jedes Satzes entweder ein Vorschubzeichen oder ein Stern ist. Die Vorschubzeichen – 0 1 2 3 4 5 6 7 / bedeuten neue Seite, 0 bis 7 Zeilen Vorschub und anderthalb Zeilen Vorschub. Ein Stern als erstes Zeichen kennzeichnet eine Zeile mit Steuerinformation, deren Bedeutung durch das zweite Zeichen bestimmt wird. Folgt dem Stern ein »/«, wird damit auf anderthalbzeiligen Druck umgeschaltet. Dies bedeutet, dass die Vorschübe um den Faktor 1,5 vergrößert werden. Mit einer »1« nach dem Stern wird wieder auf einzeiligen Druck umgeschaltet.

Zusätzlich zu den üblichen Daten kann eine solche Datei noch Steuerinformation für den Drucker enthalten, deren Bedeutung in der folgenden Tabelle angegeben ist.

```
#^_   Nachfolgendes Zeichen unterstreichen
#=nnn auf Druckposition nnn positionieren
#-nnn um nnn Druckpositionen nach links positionieren
#+nnn um nnn Druckpositionen nach rechts positionieren
#=000 aktuelle Druckposition merken
#-000 auf gemerkte Druckposition positionieren
#+000 auf nächste freie Druckposition positionieren
```

## Schwester-Dateien

Schwester-Dateien sind zwei TUSTEP-Dateien, bei denen von den Sätzen der ersten Datei auf die der zweiten Datei verwiesen wird. Solche Dateien werden mit den Kommandos #RVORBEREITE und #SVORBEREITE (siehe Seite 220 bzw. Seite 229) erzeugt, indem sie zur Spezifikation ZIEL bzw. DATEN angegeben werden. In diesem Fall wird in die zur Spezifikation ZIEL angegebene Datei nur der zum Sortieren benötigte Teil (das sind meist die Referenz und die Sortierschlüssel) mit einem Verweis ausgegeben. Der Rest (das sind die zu sortierenden Daten) wird in die zur Spezifikation DATEN angegebene Datei ausgegeben.

Die Datei mit Verweisen darf nur sortiert oder gemischt werden. Keinesfalls dürfen die Satznummern dieser Datei verändert werden, da sie die Verweise enthalten. Die andere Datei, auf deren Sätze verwiesen wird, darf in keiner Weise verändert werden, da sonst die Verweise nicht mehr stimmen.

Daten von Schwester-Dateien können mit den Kommandos #SORTIERE und #MISCHE (siehe Seite 224 bzw. Seite 201) und mit den Kommandos #RAUFBEREITE und #KOPIERE (siehe Seite 215 bzw. Seite 172) wieder zusammengeführt werden, indem die Datei mit den Verweisen zur Spezifikation QUELLE und die andere Datei zur Spezifikation DATEN angegeben wird.

## Einrichten von Dateien

Außer den Standard-Dateien (siehe Seite 30) werden von TUSTEP keine Dateien automatisch eingerichtet. Dateien, die noch nicht existieren und in TUSTEP verwendet werden sollen, müssen zuvor explizit mit dem Kommando #DATEI (siehe Seite 128) oder der Makrofunktion CREATE (siehe Seite 479) eingerichtet werden. Eine Ausnahme bilden die Band-Dateien; sie können auch mit dem Kommando #MBLABEL (siehe Seite 197) eingerichtet werden.

Da in einer TUSTEP-Sitzung nur etwa 4000 Dateien (ohne die von TUSTEP intern angemeldeten oder eingerichteten Dateien) gleichzeitig angemeldet sein können, kann es notwendig sein, dass Dateien, die vorübergehend nicht mehr benötigt werden, erst abgemeldet werden, bevor weitere Dateien eingerichtet werden können.

## Anmelden von Dateien

Dateien, die schon existieren und in TUSTEP verwendet werden sollen, müssen zuvor mit dem Kommando #ANMELDE (siehe Seite 124) oder mit der Makrofunktion OPEN (siehe Seite 481) angemeldet (= zum Bearbeiten zugelassen) werden. Eine Ausnahme bilden die Band-Dateien; sie müssen nicht angemeldet werden, wenn sie mit einem der mit #MB... beginnenden Kommandos verwendet werden.

Beim Anmelden einer Datei kann bestimmt werden, ob die Datei nur gelesen oder auch beschrieben (geändert) werden darf. Damit kann verhindert werden, dass Dateien, die nur gelesen werden sollen, versehentlich geändert oder gelöscht werden.

Eine Anmeldung gilt immer nur für die jeweilige TUSTEP-Sitzung.

Da in einer TUSTEP-Sitzung nur etwa 4000 Dateien (ohne die von TUSTEP intern angemeldeten oder eingerichteten Dateien) gleichzeitig angemeldet sein können, kann es notwendig sein, dass Dateien, die vorübergehend nicht mehr benötigt werden, erst abgemeldet werden, bevor weitere Dateien angemeldet werden können.

### Abmelden von Dateien

Um zu verhindern, dass versehentlich auf Dateien zugegriffen wird, die in der gleichen TUSTEP-Sitzung eingerichtet oder angemeldet wurden, können Dateien mit dem Kommando #ABMELDE (siehe Seite 119) oder der Makrofunktion CLOSE (siehe Seite 482) abgemeldet werden.

Zu beachten ist, dass Scratch-Dateien gelöscht werden, wenn sie abgemeldet werden; sie existieren dann nicht mehr.

### Löschen von Dateien

Dateien, die nicht mehr (auch später nicht) benötigt werden, können mit dem Kommando #LOESCHE (siehe Seite 182) oder der Makrofunktion DELETE (siehe Seite 484) gelöscht werden. Eine Ausnahme bilden die Band-Dateien; sie können auch mit dem Kommando #MBLABEL (siehe Seite 197) gelöscht werden.

Wird eine TUSTEP-Sitzung beendet (nicht nur unterbrochen), werden die zu dieser Sitzung gehörenden Scratch-Dateien automatisch gelöscht.

### Drucken von Dateien

Sollen die Daten einer Datei ausgedruckt werden, so müssen sie zuvor zum Drucken aufbereitet werden, falls die Datei keine Protokoll-Datei ist. Dies kann mit dem Kommando #FORMATIERE geschehen, wenn die Datei Formatieranweisungen für dieses Kommando enthält, die interpretiert werden sollen, andernfalls mit dem Kommando #DVORBEREITE. Dabei wird die Zeilen- und Seiteneinteilung für den Ausdruck festgelegt und ein Protokoll erstellt, das in eine Protokoll-Datei ausgegeben wird.

Eine Protokoll-Datei kann mit dem Kommando #DRUCKE (siehe Seite 142) ausgedruckt werden. Dabei werden die Daten in die für den jeweiligen Drucker notwendigen Steuerzeichen umgesetzt.

Bei manchen Druckern (z. B. bei PostScript-Druckern) wird mit der Angabe des Druckertyps jeweils auch festgelegt, ob im Hoch- oder im Querformat gedruckt werden soll. Im Querformat können auch automatisch zwei »Seiten« nebeneinander auf eine Seite gedruckt werden. Welche Druckertypen definiert sind, kann mit dem Kommando #LISTE, DRUCKERTYPEN (siehe Seite 173) aufgelistet werden.

Protokoll-Dateien (z. B. das Ergebnis eines Vergleichs zweier Dateien mit dem Kommando #VERGLEICHE) können mit dem Kommando

```
#DRUCKE, datei, typ, +
```

auf dem Bildschirm angezeigt werden. Dabei müssen für *datei* der Name der Pro-

tokoll-Datei und für `typ` ein Druckertyp angegeben werden. Unter Windows kann als Druckertyp `WIN-10` oder `WIN-12`, unter Linux `PS-10` oder `PS-12` und unter macOS `MAC-10` oder `MAC-12` verwendet werden.

Um Protokoll-Dateien auf dem voreingestellten Drucker auszugeben, genügt

```
#DRUCKE, datei, typ
```

Dabei muss zu `typ` der dem Drucker entsprechende Typ angegeben werden. Eine Liste mit den von TUSTEP vorgesehenen Druckertypen wird mit dem Kommando `#LISTE, DRUCKERTYPEN` angezeigt. Falls der Drucker keinem der vorgesehenen Typen entspricht, kann unter Windows `WIN-10` oder `WIN-12` angegeben werden, unter Linux und macOS kann mit dem Kommando

```
#DATEI, name.ps, FDF-P
```

eine Datei eingerichtet werden. Der Name der Datei ist beliebig, er muss jedoch die Endung `ps` haben. Danach kann mit dem TUSTEP-Kommando

```
#DRUCKE, datei, typ, datei=name.ps, +
```

aus der Protokoll-Datei eine PostScript-Datei erstellt werden, wobei `typ` ein mit »PS« beginnender Druckertyp (z. B. `PS-10` oder `PS-12`) ist. Diese PostScript-Datei kann dann wie nachfolgend beschrieben ausgedruckt werden.

Um PostScript-Dateien (z. B. auch das Ergebnis von den Kommandos `#SATZ / #*PSAUS`) auf dem Bildschirm anzuzeigen oder auszudrucken, kann das Kommando

```
#*ZEPS, datei
```

verwendet werden. Für `datei` muss der Name der PostScript-Datei angegeben werden; sie muss die Endung `ps` haben und darf keine Scratch-Datei sein. Nachdem die Datei angezeigt wird, kann sie über die Druckfunktion des Anzeigeprogramms auf einem Drucker ausgegeben werden.

Welches Programm zum Anzeigen verwendet wird, hängt davon ab, welche Programme installiert sind. Unter Linux und macOS ist ein entsprechendes Programm schon installiert, unter Windows muss ggf. noch ein geeignetes Programm (z. B. Ghostscript) installiert werden. Abhängig vom verwendeten Programm kann das Ergebnis variieren.

Um PostScript-Dateien in PDF-Dateien umzuwandeln, kann das Kommando

```
#*PS2PDF, quelle.ps, ziel.pdf
```

verwendet werden. Für `quelle.ps` muss der Name der PostScript-Datei und für `ziel.pdf` der Name der PDF-Datei angegeben werden; die Dateinamen müssen die Endungen `ps` bzw. `pdf` haben.

Welches Programm zum Umwandeln verwendet wird, hängt davon ab, welche Programme installiert sind. Unter Linux und macOS ist ein entsprechendes Programm schon installiert, unter Windows muss ggf. noch ein geeignetes Programm (z. B. Ghostscript) installiert werden. Abhängig vom verwendeten Programm kann das Ergebnis variieren.

Um PDF-Dateien auf dem Bildschirm anzuzeigen oder auszudrucken, kann das Kommando

```
#*ZEPDF, datei
```

verwendet werden. Unter älteren Windows-Versionen ist Voraussetzung, dass ein geeignetes Anzeigeprogramm (z. B. der Adobe-Reader, früher Acrobat-Reader) installiert ist. Für `datei` muss der Name der PDF-Datei angegeben werden; sie muss die Endung `pdf` haben und darf keine Scratch-Datei sein. Nachdem die Datei angezeigt wird, kann sie über die Druckfunktion des Anzeigeprogramms auf einem Drucker ausgegeben werden.

## Retten von Dateien

Dateien werden in TUSTEP nach dem Beschreiben automatisch abgeschlossen. Auch beim Abbruch eines TUSTEP-Programms werden die Dateien nach Möglichkeit korrekt abgeschlossen. Ist dies nicht möglich (z. B. weil eine Datei nicht mehr erweitert werden kann oder weil der Strom ausgefallen ist), so sind die Daten solcher Dateien gesperrt. Damit wird vermieden, dass in Programmen, die diese Daten weiterverarbeiten sollen, unvorhergesehene Folgefehler auftreten. Sollen die Daten weiterverwendet werden, so müssen sie erst mit dem Kommando `#RETTE` (siehe Seite 216) ukopiert oder mit der Makrofunktion `REPAIR` (siehe Seite 489) repariert werden.

## Reorganisieren von Dateien

Beim Beschreiben einer Datei werden die Sätze entsprechend ihrer Reihenfolge lückenlos hintereinander angeordnet. Wird eine Datei mit dem Editor oder mit Makroanweisungen geändert, so kann diese Ordnung aus der Sicht des Rechners aus verschiedenen Gründen in Unordnung geraten:

- Wenn ein Satz gelöscht wird, entsteht eine Lücke.
- Wenn ein Satz verkürzt wird, entsteht ebenfalls eine Lücke.
- Wenn ein Satz verlängert wird, passt er nicht mehr an seinen alten Platz, falls nicht zufällig unmittelbar davor oder unmittelbar dahinter eine bestehende Lücke den Mehrbedarf abdeckt. Der Satz wird dann ans Ende der Datei geschrieben und durch entsprechende Verweise logisch an der richtigen Stelle eingereiht. Am alten Platz entsteht dabei eine Lücke. Der Fall, dass ein Satz nicht mehr an seinen alten Platz passt, tritt insbesondere dann häufig auf, wenn eine Zeichenfolge automatisch durch eine längere ausgetauscht wird.
- Wenn ein Satz hinzugefügt wird, wird geprüft, ob an der entsprechenden Stelle eine ausreichend große Lücke vorhanden ist. Wenn ja, wird er dort hineingeschrieben; wenn nicht, wird er ans Ende der Datei geschrieben und durch entsprechende Verweise logisch an der richtigen Stelle eingereiht.

Umfangreiches Ändern einer Datei kann also dazu führen, dass eine Datei große Lücken enthält und dass die logische Reihenfolge der Sätze (in der sie grundsätzlich bearbeitet werden) nicht mehr der physikalischen Anordnung entspricht. Dadurch werden die Zugriffe auf die einzelnen Sätze verlangsamt, weil einerseits die Lücken



immer mitgelesen werden und andererseits ständig an einer anderen Stelle in der Datei weitergelesen werden muss. Außerdem führen die Lücken auch zu Plattenplatzverschwendung.

Deshalb sollten Dateien, die mit dem Editor oder mit Makroanweisungen geändert worden sind, ab und zu reorganisiert werden. Dies kann z. B. mit dem Kommando `#RETTE, dateiname, -STD-, +, +` (siehe Seite 216) oder mit der Makroanweisung `REARRANGE` (siehe Seite 487) geschehen.

Die gleiche Problematik, wie oben mit Sätzen einer Datei geschildert, besteht auch beim Abspeichern von Segmenten in einer Segment-Datei. Jedoch entstehen hier viel größere Lücken, da es sich bei Segmenten in der Regel um größere Datenmengen handelt. Deshalb ist es für Segment-Dateien besonders wichtig, dass sie reorganisiert werden, wenn der Plattenplatz nicht vergeudet werden soll.

## Sperren der Daten

Dateien werden in TUSTEP nach dem Beschreiben automatisch abgeschlossen. Auch beim Abbruch eines TUSTEP-Programms werden die Dateien nach Möglichkeit korrekt abgeschlossen. Ist dies nicht möglich (z. B. weil die Datei nicht erweitert werden kann oder weil der Strom ausgefallen ist), so sind die Daten solcher Dateien gesperrt. Damit wird vermieden, dass in Programmen, die diese Daten weiterverarbeiten sollen, unvorhergesehene Folgefehler auftreten. Beim Versuch, auf gesperrte Daten zuzugreifen, wird eine Fehlermeldung ausgegeben, die besagt, dass die Datei »nicht abgeschlossen« sei.

Sollen die Daten weiterverwendet werden, so müssen sie erst mit dem Kommando `#RETTE` (siehe Seite 216) umkopiert werden. Im Prinzip können die Daten damit verlustlos gerettet werden, denn beim Löschen, Einfügen und Ersetzen von Sätzen wird immer nur an einer Stelle die Verkettung der Sätze aufgebrochen und wieder geschlossen. Kommt es dazwischen zum Programmabbruch, so wird später beim Umkopieren erkannt, dass die Zeiger an dieser Stelle nicht mehr stimmen. Da die Sätze nicht nur Zeiger auf den folgenden, sondern auch auf den vorangehenden Satz haben, können in diesem Fall die Sätze vom letzten Satz bis zu eben dieser Stelle rückwärts gelesen werden. Damit ist die fehlerhafte Verkettung überbrückt; die auf die fehlerhafte Stelle folgenden Daten können somit auch umkopiert werden.

## Sperren von Dateien

Damit eine Datei problemlos gelesen werden kann, muss die Verkettung der Sätze korrekt sein. Dies ist jedoch nicht gewährleistet, wenn ein anderes Programm gleichzeitig in der Datei Änderungen vornimmt. Um gleichzeitige Änderungen zu verhindern, wird die Datei gesperrt, wenn ein Programm auf sie zugreift.

Soll auf eine Datei lesend zugegriffen werden, so wird zuerst geprüft, ob sie für Lesezugriffe gesperrt ist. Ist dies der Fall, kann auf die Datei nicht zugegriffen werden. Andernfalls wird der Lesezugriff zugelassen; gleichzeitig wird die Datei für Schreibzugriffe gesperrt.

Soll auf eine Datei schreibend zugegriffen werden, so wird zuerst geprüft, ob sie für

Schreibzugriffe gesperrt ist. Ist dies der Fall, kann auf die Datei nicht zugegriffen werden. Andernfalls wird der Schreibzugriff zugelassen; gleichzeitig wird die Datei für Lese- und Schreibzugriffe gesperrt.

Kann ein Programm auf eine Datei nicht zugreifen, wird es abgebrochen. Für den Dateizugriff gibt es Makroanweisungen, mit denen Dateizugriffe koordiniert werden können. Dies ist insbesondere für CGI-Makros von Bedeutung, da bei Makroaufrufen, die über das WWW eingehen, immer damit zu rechnen ist, dass mehr als ein Aufruf gleichzeitig erfolgt.

# Dateistruktur

---

Dateistruktur . . . . .	45
Zeiger . . . . .	45
Einteilung der Daten in Sätze . . . . .	45
Aufbau eines Satzes . . . . .	46
Verkettung der Sätze . . . . .	46
Löschen eines Satzes . . . . .	47
Ersetzen eines Satzes . . . . .	48
Einfügen eines Satzes . . . . .	49
Umstellen eines Satzes . . . . .	50
Zugriff auf die Sätze . . . . .	51
Aufbau der Zeigertabelle . . . . .	51

## Dateistruktur

TUSTEP-Dateien sind in zwei bzw. drei Bereiche unterteilt. Der erste Bereich enthält Verwaltungsinformationen. Dazu gehören u. a. folgende Angaben:

- der Dateititel,
- ob die Datei vom Typ SEQ oder RAN ist,
- wo der erste und der letzte Satz in der Datei stehen,
- wieviele Sätze die Datei enthält,
- wann die Daten der Datei zuletzt geändert wurden,
- ob die Datei ordnungsgemäß abgeschlossen ist.

Der zweite Bereich enthält die eigentlichen Daten. Bei Dateien vom Typ RAN gibt es einen dritten Bereich, der aus einer Zeigertabelle besteht. Sie enthält Angaben, an welchen Stellen die einzelnen Seiten des in der Datei gespeicherten Textes zu finden sind.

### Zeiger

Das TUSTEP-Dateiformat wurde zu einer Zeit festgelegt, als manche Rechner die Daten in einer Datei nicht (wie heute üblich) byte-weise, sondern nur »wortweise« adressieren konnten. Ein »Wort« entsprach 4 Bytes. Um für alle Rechner ein einheitliches Dateiformat zu ermöglichen, mussten für Zeiger (Pointer) Wort-Adressen statt Byte-Adressen verwendet werden. Wird eine Wort-Adresse mit 4 multipliziert, so ist das Ergebnis die entsprechende Byte-Adresse.

### Einteilung der Daten in Sätze

Die Daten in einer TUSTEP-Datei sind in Sätze (records) unterteilt. Enthält eine Datei z. B. den Text eines Buches, so kann ein Satz der Datei jeweils eine Zeile des Textes enthalten. Die Einteilung eines Textes in Sätze ist jedoch frei wählbar; sie kann z. B. nach Gesichtspunkten der Übersichtlichkeit gewählt werden. Die einzige Einschränkung betrifft die Satzlänge. Soll mit dem Editor ein Satz geändert werden, so darf dieser nicht länger als 8000 Zeichen sein bzw. auf Grund der Änderung nicht länger als 8000 Zeichen werden. In allen anderen Fällen dürfen die Sätze bis zu 64000 Zeichen lang sein.

Jeder Satz einer TUSTEP-Datei enthält außer den Daten noch eine Satznummer. Sie ist auf Seite 32 beschreiben.

## Aufbau eines Satzes

Jeder Satz enthält außer dem Text noch einen Vorspann mit fünf Zahlenwerten: Satzlänge, Seitennummer, Zeilennummer und zwei Zeiger (pointer). Die Seiten- und Zeilennummer bilden zusammen die Satznummer.

PN	PV	SN	ZN	SL	Text
----	----	----	----	----	------

- PN Zeiger auf den nachfolgenden Satz
- PV Zeiger auf den vorangehenden Satz
- SN Seitennummer
- ZN Zeilennummer (mit Unterscheidungsnummer)
- SL Satzlänge (Anzahl der Zeichen)

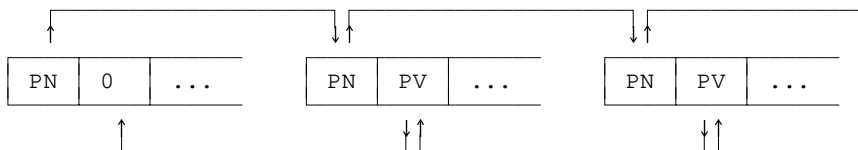
Von diesen organisatorischen Daten wird bei der Arbeit mit TUSTEP nur die Satznummer sichtbar. Die Zeiger werden nur intern verwendet und erlauben das interaktive Arbeiten mit großen Dateien (in TUSTEP bis 2 GB je Datei).

## Verkettung der Sätze

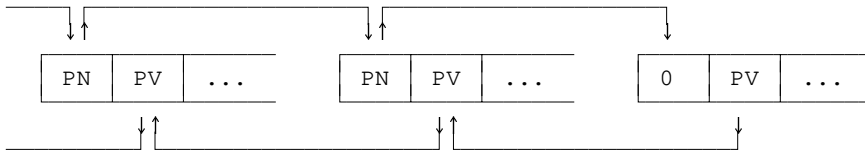
Der erste der beiden Zeiger eines Satzes zeigt jeweils auf den nachfolgenden Satz, der zweite zeigt jeweils auf den vorangehenden Satz. Damit ist jeder Satz mit seinem Vorgänger und seinem Nachfolger verkettet. Da der erste Satz keinen Vorgänger und der letzte Satz keinen Nachfolger hat, haben die entsprechenden Zeiger den Wert 0 (Null).

Anmerkung: In den Dateien zeigen alle Zeiger auf den Anfang der Sätze, also auf den ersten Zeiger des jeweiligen Satzes. Damit sich die Pfeile in den folgenden Darstellungen möglichst wenig kreuzen, zeigen die Pfeile, die einen Zeiger auf den vorangehenden Satz darstellen, nicht auf den Anfang der Sätze, sondern auf den zweiten Zeiger des jeweiligen Satzes.

Verkettung der Sätze am Dateianfang:



Verkettung der Sätze am Dateiende:

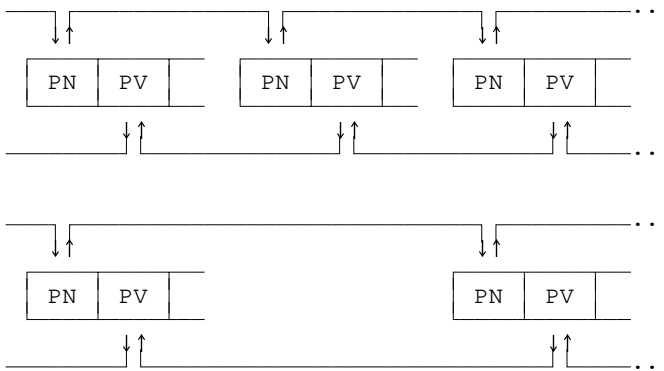


Durch die Verkettung der Sätze ist es nicht notwendig, dass die logische Reihenfolge der Sätze auch der physikalischen Anordnung entspricht. Dies ist z. B. von Bedeutung, wenn beim Korrigieren ein Satz eingefügt werden soll. Er kann vom Rechner ans Dateiende geschrieben werden; die Zeiger werden dann automatisch so geändert, dass der Satz logisch an der richtigen Stelle steht. Würde der Satz an die Stelle geschrieben, an der er logisch stehen soll, müssten alle auf den eingefügten Satz folgenden Sätze nach hinten verschoben werden, d. h. der ganze Rest der Datei müsste vom Rechner gelesen und um ein Stück versetzt wieder geschrieben werden.

### Löschen eines Satzes

Beim Löschen eines Satzes wird dieser aus der Zeigerkette herausgenommen, indem die Zeiger des vorangehenden und des nachfolgenden Satzes entsprechend geändert werden.

Zeiger vor und nach dem Löschen:



Außerdem wird der Platz, den der zu löschende Satz belegt, mit einem eindeutigen Bitmuster überschrieben, damit er als freier Platz erkennbar ist und bei Bedarf neu belegt werden kann.

## Ersetzen eines Satzes

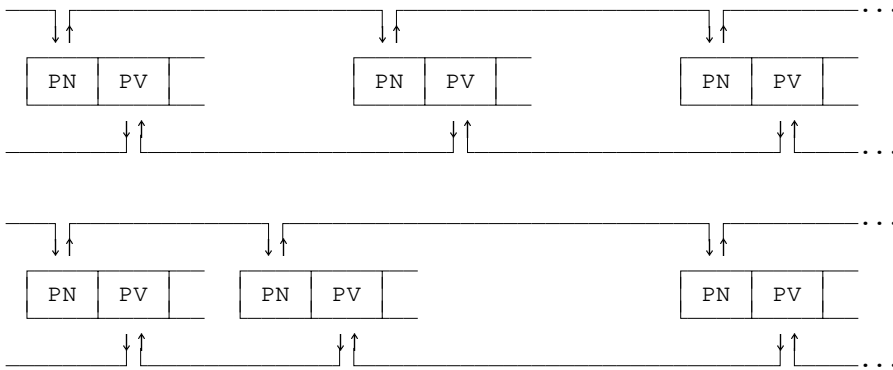
Wenn sich beim Korrigieren die Länge des Satzes nicht ändert, wird er wieder an dieselbe Stelle geschrieben. Das gleiche gilt, wenn er kürzer wird; der in diesem Fall übrig bleibende Platz wird mit einem eindeutigen Bitmuster überschrieben, damit er als freier Platz erkennbar ist und bei Bedarf neu belegt werden kann.

Wenn der Satz länger wird, so wird zunächst geprüft, ob unmittelbar nach dem Satz noch freier Platz vorhanden ist. Reicht der freie Platz aus, wird der Satz ab derselben Stelle geschrieben; in diesem Fall brauchen keine Zeiger geändert zu werden.

Ist unmittelbar nach dem Satz kein oder zu wenig freier Platz vorhanden, wird der Satz mit einem eindeutigen Bitmuster überschrieben, damit er als freier Platz erkennbar ist und bei Bedarf neu belegt werden kann. Dann wird geprüft, ob unmittelbar davor noch freier Platz vorhanden ist.

Reicht der nun insgesamt an dieser Stelle freie Platz aus, wird der Satz an den Anfang des freien Platzes geschrieben; außerdem werden die Zeiger des vorangehenden und des nachfolgenden Satzes entsprechend geändert.

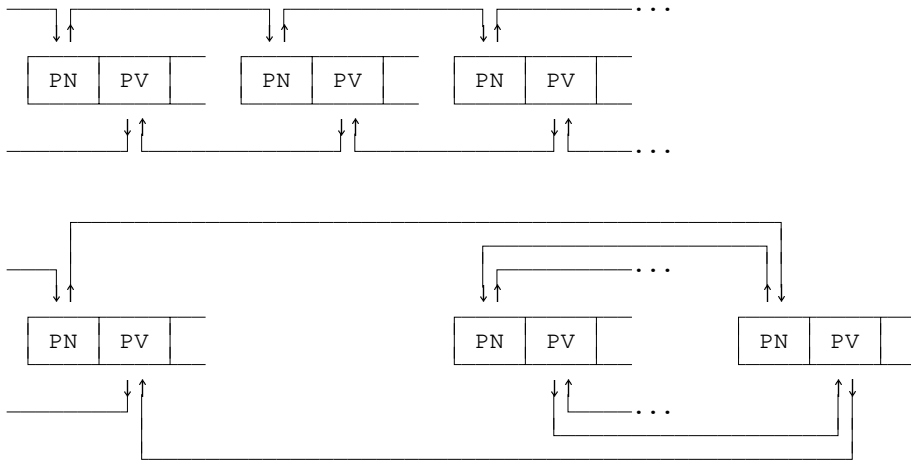
Zeiger vor und nach dem Ersetzen mit Verschieben:



Reicht der an dieser Stelle freie Platz nicht aus, wird der Satz ans Ende der Datei geschrieben; außerdem werden die Zeiger des vorangehenden und des nachfolgenden Satzes entsprechend geändert.



Zeiger vor und nach dem Ersetzen durch Anhängen am Dateiende:

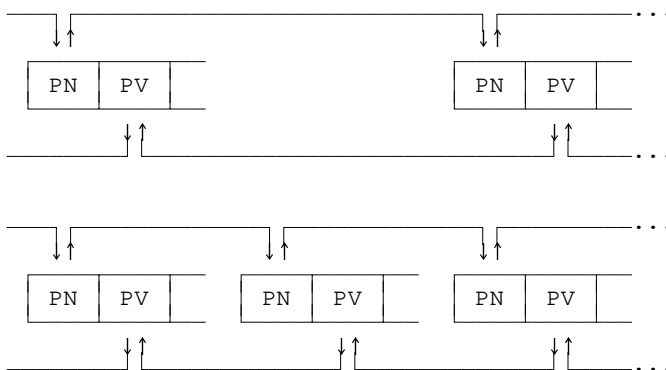


### Einfügen eines Satzes

Beim Einfügen eines Satzes wird zunächst geprüft, ob unmittelbar nach dem Satz, nach dem eingefügt werden soll, noch ausreichend freier Platz vorhanden ist. Reicht der freie Platz aus, wird der Satz an diese Stelle geschrieben; außerdem werden die Zeiger des vorangehenden und des nachfolgenden Satzes entsprechend geändert.

Ist an dieser Stelle kein oder zu wenig freier Platz vorhanden und stehen die beiden Sätze, zwischen denen eingefügt werden soll, in der Datei nicht direkt hintereinander, so wird noch geprüft, ob unmittelbar vor dem Satz, vor dem eingefügt werden soll, noch ausreichend freier Platz vorhanden ist. Reicht der freie Platz aus, wird der Satz an diese Stelle geschrieben; außerdem werden die Zeiger des vorangehenden und des nachfolgenden Satzes entsprechend geändert.

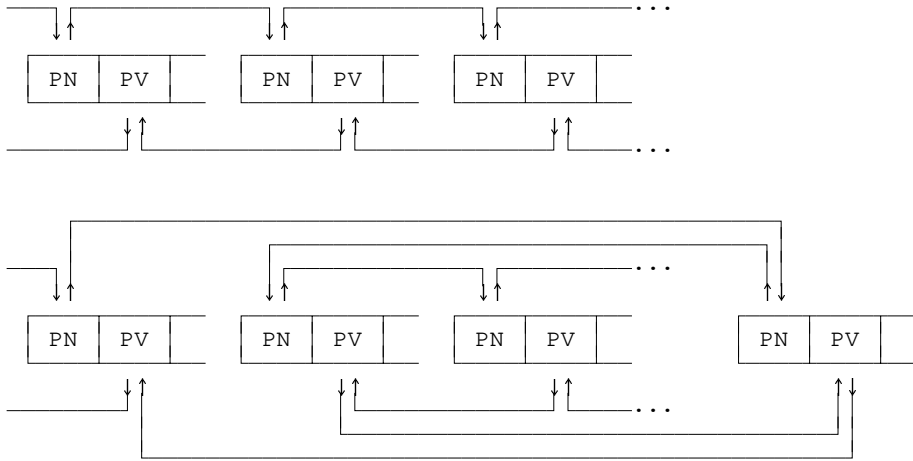
Zeiger vor und nach dem Einfügen an Ort und Stelle:



Reicht der freie Platz an beiden Stellen nicht aus, wird der Satz ans Ende der Datei

geschrieben; außerdem werden die Zeiger des vorangehenden und des nachfolgenden Satzes entsprechend geändert.

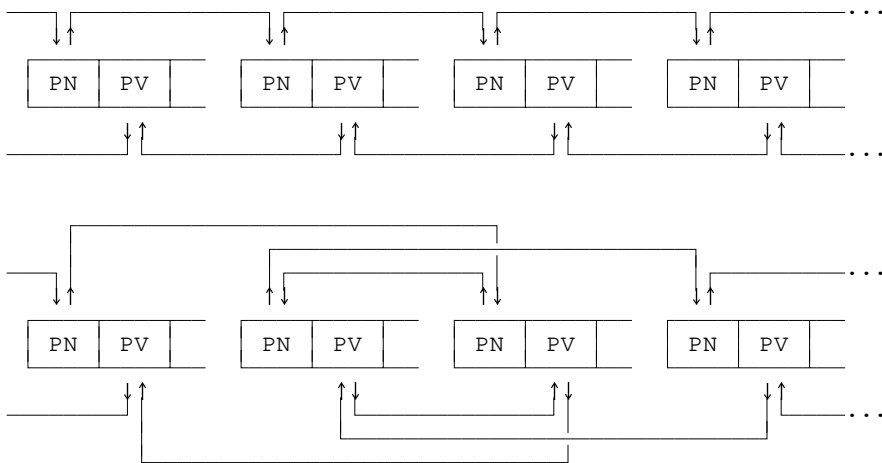
Zeiger vor und nach dem Einfügen durch Anhängen am Dateiende:



### Umstellen eines Satzes

Beim Umstellen eines Satzes bleibt dieser an seiner Stelle stehen. Es werden lediglich seine Zeiger und die Zeiger des vorangehenden und des nachfolgenden Satzes entsprechend geändert.

Zeiger vor und nach dem Umstellen:



Beim Umstellen mehrerer aufeinander folgender Sätze werden nur die Zeiger des

ersten umzustellenden und die des vorangehenden Satzes sowie die Zeiger des letzten umzustellenden und des nachfolgenden Satzes entsprechend geändert. Die Zeiger der restlichen umzustellenden Sätze müssen nicht geändert werden.

## Zugriff auf die Sätze

Mit Hilfe der beiden Zeiger in jedem Satz und der Zeiger zum ersten und letzten Satz in der Verwaltungsinformation am Dateianfang können die einzelnen Sätze in ihrer logischen Reihenfolge sowohl vor- als auch rückwärts verarbeitet werden.

Soll auf einen Satz mit einer bestimmten Satznummer zugegriffen werden, so muss intern der Reihe nach jeder einzelne Satz daraufhin geprüft werden, ob er die geforderte Satznummer hat. Dies kann dadurch eventuell verkürzt werden, dass zuerst auf Grund der Satznummer geprüft wird, ob der geforderte Satz vor oder nach dem zuletzt verarbeiteten Satz (von dem die Position ja noch bekannt ist) steht. Steht der geforderte Satz danach, kann von der aktuellen Position aus vorwärts, andernfalls rückwärts gelesen werden.

Dieses Verfahren wird bei Dateien vom Typ SEQ angewandt. Es erscheint zeitaufwändig, ist aber für Dateien bis zu einer Größe von einem Megabyte tragbar. Bei Dateien vom Typ RAN wird zusätzlich automatisch eine Zeigertabelle verwaltet, die den Zugriff auf einen Satz mit einer bestimmten Satznummer erheblich beschleunigt.

## Aufbau der Zeigertabelle

TUSTEP-Dateien vom Typ RAN enthalten im Anschluss an die eigentlichen Daten noch eine Zeigertabelle. Sie hat vier Spalten und ist bis zu 1024 Zeilen lang.

Von_Seite	Bis_Seite	Umfang	Zeiger
-----------	-----------	--------	--------

Von_Seite	Erste Seite des Bereichs
Bis_Seite	Letzte Seite des Bereichs
Umfang	Platzbedarf des Bereichs
Zeiger	Zeiger auf den ersten Satz des Bereichs

Bei Dateien mit bis zu 1024 Seiten hat die Tabelle für jede Seite eine Zeile; ein Bereich besteht in diesem Fall aus genau einer Seite; in der ersten und zweiten Spalte steht jeweils die gleiche Seitenzahl.

Enthält eine Datei mehr als 1024 Seiten, so sind jeweils mehrere aufeinander folgende Seiten zu einem Bereich zusammengefasst; in der ersten Spalte wird die Nummer der ersten Seite, in der zweiten Spalte die Nummer der letzten Seite des jeweiligen Bereichs angegeben.

Wenn eine neue Seite in der Datei hinzukommt und die Tabelle schon voll ist, werden zwei Bereiche zu einem Bereich zusammengefasst. Dazu wird in der dritten Spalte der Bereich mit dem kleinsten Platzbedarf gesucht und dieser Bereich dann mit

dem nachfolgenden zusammengefasst; die neue Seite kann nun als neuer Bereich in die Tabelle eingetragen werden. Ist der letzte Bereich in der Tabelle der kleinste und die Nummer der neuen Seite größer als die Nummer der letzten Seite, so wird die neue Seite diesem Bereich zugeordnet.

Soll direkt auf einen Satz mit einer bestimmten Satznummer zugegriffen werden, so wird in der Tabelle nachgeschaut, in welchen Seitenbereich der geforderte Satz fällt. In der vierten Spalte kann dann abgelesen werden, an welcher Stelle in der Datei der entsprechende Bereich beginnt. Von dieser Stelle an wird wie bei einer Datei vom Typ SEQ der geforderte Satz gesucht.

Voraussetzung für dieses Vorgehen zum direkten Auffinden eines Satzes in einer Datei vom Typ RAN ist, dass die Satznummern alle aufsteigend und voneinander verschieden sind. Wird beim Beschreiben einer Datei vom Typ RAN nicht darauf geachtet, bricht das Programm mit der Fehlermeldung »Satznummernkonflikt« ab.

# Daten-Transfer

---

Zwischenablage . . . . .	55
Import / Export von Daten . . . . .	56
Daten-Transfer auf lokalem Rechner . . . . .	56
Daten-Transfer auf andere Rechner . . . . .	59

## Zwischenablage

Der TUSTEP-Editor kennt neben der »Zwischenablage« auch noch einen zweiten Speicherbereich, in dem Daten zwischengespeichert werden können; er wird mit »Editor-Zwischenspeicher« bezeichnet (siehe »Zwischenablage – Editor-Zwischenspeicher« Seite 258).

### Windows :

Unter Windows kann innerhalb und außerhalb von TUSTEP direkt auf die Zwischenablage zugegriffen werden, da für die TUSTEP-Zwischenablage und die Windows-Zwischenablage derselbe Speicherbereich verwendet wird.

### Linux :

Unter Linux kann der Inhalt der TUSTEP-Zwischenablage mit dem Kommando

```
#*CB, EXPORT
```

in das Clipboard von Linux übertragen werden; der Inhalt des Clipboard kann mit

```
#*CB, IMPORT
```

in die TUSTEP-Zwischenablage übertragen werden.

### macOS :

Unter macOS kann der Inhalt der TUSTEP-Zwischenablage mit dem Kommando

```
#*PB, EXPORT
```

in das Pasteboard von macOS übertragen werden; der Inhalt des Pasteboard kann mit

```
#*PB, IMPORT
```

in die TUSTEP-Zwischenablage übertragen werden.

An Stellen, an denen es in TUSTEP möglich ist, mit `Ctrl+V` den Inhalt der TUSTEP-Zwischenablage einzufügen, kann auch mit `Cmd+V` der Inhalt des Pasteboard von macOS eingefügt werden.

## Import / Export von Daten

Zum Import von Daten aus Fremd-Dateien in TUSTEP-Dateien bzw. Export von Daten aus TUSTEP-Dateien in Fremd-Dateien kann das Kommando #UMWANDLE (siehe Seite 237) verwendet werden. Dabei können auch noch eventuell notwendige Umcodierungen vorgenommen werden.

Um Daten von Textverarbeitungsprogrammen wie z. B. MS-WORD ohne Formatierung und ohne Auszeichnungen zu übernehmen, können sie mit dem entsprechenden Programm auch als »Nur Text« in eine Datei abgespeichert werden; die Daten dieser Datei müssen dann mit #UMWANDLE in eine TUSTEP-Datei kopiert werden.

Um Daten von Textverarbeitungsprogrammen wie z. B. MS-WORD mit Formatierung und Auszeichnungen zu übernehmen, können sie mit dem entsprechenden Programm als XML-Datei (Word **2003** XML-Dokument, nicht als Word XML-Dokument!) oder als RTF-Datei (Rich-Text-Format) abgespeichert werden; die Daten dieser Datei müssen dann mit dem Makro \*IMPORT in eine TUSTEP-Datei importiert werden. Weitere Informationen zu diesem Makro werden mit dem Kommando #\*ZEBE, IMPORT angezeigt.

Entsprechend können Texte, die in TUSTEP-Dateien gespeichert sind und geeignete Angaben zur Formatierung und Textauszeichnung enthalten, mit dem Makro \*EXPORT für andere Textverarbeitungsprogramme in RTF-Dateien exportiert werden. Weitere Informationen zu diesem Makro werden mit dem Kommando #\*ZEBE, EXPORT angezeigt.

## Daten-Transfer auf lokalem Rechner

Mit den im Folgenden beschriebenen Makros \*UPLOAD und \*DOWNLOAD können innerhalb von TUSTEP Dateien von beliebiger Stelle im Dateisystem (z. B. von einem USB-Speicher-Stick) in ein für TUSTEP direkt zugängliches Verzeichnis (für das eine geeignete System-Variable definiert ist, die als Träger verwendet werden kann; siehe »Identifizieren einer Datei« Seite 27) bzw. von einem solchen Verzeichnis an eine beliebige Stelle im Dateisystem kopiert werden.



## Aufruf des Makros \*UPLOAD

#\*UPLOAD

PFAD	= pfad	<p>Pfad und Name der Datei, die die zu kopierenden bzw. die zu übertragenden Daten enthält. Verzeichnisnamen und Dateiname dürfen kein Leerzeichen und nur Buchstaben von A bis Z, Minuszeichen, Unterstreichungsstriche und Punkte enthalten. Unter Linux ist auf Groß- und Kleinschreibung zu achten.</p> <p>Unter Linux und macOS kann die Pfadangabe auch mit dem Namen eines USB-Speicher-Sticks gefolgt von einem Doppelpunkt beginnen.</p> <p>Falls die Angabe mit \ oder / endet, wird der Name der ZIEL-Datei (ohne Projektname) ergänzt.</p>
ZIEL	= name	Name der Datei, die die zu kopierenden bzw. die zu übertragenden Daten aufnehmen soll.
MODUS	= -STD- = BINAER = TEXT	<p>* Wie BINAER.</p> <p>Datei wird binär (unverändert) übertragen. nur REMOTE: TUSTEP-Dateien (Typ SEQ/RAN) werden binär (d. h. unverändert) übertragen. Bei Fremd-Dateien (Typ FDF) wird angenommen, dass es ANSI-Text-Daten (d. h. nach CP1252 codierte Daten) sind; deshalb werden die Codes für Zeilenwechsel angepasst (aus CR+LF wird LF). Falls andere Dateien übertragen werden, muss als Modus BINAER angegeben werden.</p>
LOESCHEN	= -  = +	<p>* Falls die ZIEL-Datei schon/noch Daten enthält, wird die Datei nicht übertragen.</p> <p>Falls die ZIEL-Datei schon/noch Daten enthält, werden diese überschrieben.</p>
MELDUNG	= +  = -	<p>* Nach dem Kopieren/Übertragen der Daten eine Meldung ausgegeben. Unter Windows muss diese bestätigt werden.</p> <p>Keine Meldung ausgegeben.</p>

## Leistung

Mit dem Makro \*UPLOAD können Daten aus einer Datei, die an beliebiger Stelle im Dateisystem (z. B. auf einem USB-Speicher-Stick) steht, in eine Datei kopiert werden, die zum Schreiben angemeldet ist.

Außerdem können unter Windows in einer REMOTE-Sitzung Dateien vom lokalen WINDOWS-Rechner auf den Linux-Rechner übertragen werden.

**Aufruf des Makros \*DOWNLOAD**

#\*DOWNLOAD

QUELLE	= name	Name der Datei, die die zu kopierenden bzw. die zu übertragenden Daten enthält.
PFAD	= pfad	Pfad und Name der Datei, die die Daten aufnehmen soll. Verzeichnisnamen und Dateiname dürfen kein Leerzeichen und nur Buchstaben von A bis Z, Minuszeichen, Unterstreichungsstriche und Punkte enthalten. Unter Linux ist auf Groß- und Kleinschreibung zu achten. Unter Linux und macOS kann die Pfadangabe auch mit dem Namen eines USB-Speicher-Sticks gefolgt von einem Doppelpunkt beginnen. Falls die Angabe mit \ oder / endet, wird der Name der QUELL-Datei (ohne Projektname) ergänzt.
MODUS	= -STD- = BINAER = TEXT	* Wie BINAER. Datei wird binär (unverändert) übertragen. nur REMOTE: TUSTEP-Dateien (Typ SEQ/RAN) werden binär (d. h. unverändert) übertragen. Bei Fremd-Dateien (Typ FDF) wird angenommen, dass es ANSI-Text-Daten (d. h. nach CP1252 codierte Daten) sind; deshalb werden die Codes für Zeilenwechsel angepasst (aus LF wird CR+LF). Falls andere Dateien übertragen werden, muss als Modus BINAER angegeben werden.
	= BROWSE	nur REMOTE: Nach dem Übertragen der Datei wird auf dem WINDOWS-Rechner die Anwendung gestartet, die für die Endung der Datei im Betriebssystem definiert ist.
	= PREVIEW	nur REMOTE: Die zu übertragende Datei muss eine TUSTEP-Protokoll-Datei sein. Nach dem Übertragen der Datei wird das in der Datei enthaltene Protokoll auf dem Bildschirm angezeigt.
LOESCHEN	= - = +	* Falls die zu PFAD angegebene Datei schon vorhanden ist, wird nachgefragt, ob sie überschrieben werden darf. Falls die zu PFAD angegebene Datei schon vorhanden ist, wird sie ohne nachzufragen überschrieben.
MELDUNG	= + = -	* Nach dem Kopieren/Übertragen der Daten eine Meldung ausgegeben. Unter Windows muss diese bestätigt werden. Keine Meldung ausgegeben.

## Leistung

Mit dem Makro \*DOWNLOAD können Daten aus einer Datei, die zum Lesen oder Schreiben angemeldet ist, in eine Datei kopiert werden, die an beliebiger Stelle im Dateisystem (z. B. auf einem USB-Speicher-Stick) steht. Falls diese Datei noch nicht existiert, wird sie eingerichtet.

Außerdem können unter Windows in einer REMOTE-Sitzung Dateien vom Linux-Rechner auf den WINDOWS-Rechner übertragen werden.

Falls für die vom REMOTE-Rechner übertragene Datei auf dem Windows-Rechner eine geeignete Anwendung vorhanden ist, kann diese automatisch gestartet werden.

Falls die vom REMOTE-Rechner zum Windows-Rechner übertragene Datei eine TUSTEP-Protokoll-Datei ist, kann das Protokoll auf dem Bildschirm automatisch angezeigt werden.

## Daten-Transfer auf andere Rechner

Beim Übertragen von TUSTEP-Dateien auf andere Rechner muss sichergestellt sein, dass die Daten binär übertragen werden. Es darf also kein einziges Byte der Datei dabei verändert werden.

Falls dies nicht sichergestellt ist, müssen TUSTEP-Dateien zuvor mit dem Kommando #UMWANDLE (siehe Seite 237) in ASCII-Dateien kopiert und dabei in ein zum Datenaustausch geeignetes Format (Datenaustauschformat) gebracht werden. Beim Aufruf von UMWANDLE muss dazu die Spezifikation MODUS=TX angegeben werden. Nach dem Übertragen können die Dateien mit dem Kommando #UMWANDLE wieder in die ursprüngliche Form gebracht und in TUSTEP-Dateien kopiert werden. Beim Aufruf von UMWANDLE muss dazu die Spezifikation MODUS=XT angegeben werden. Dieses Verfahren funktioniert zwischen beliebigen Rechnern bzw. Betriebssystemen.

Falls eine TUSTEP-Datei binär übertragen wurde und der Rechner, von dem sie übertragen wurde, und der Rechner, auf den sie übertragen wurde, interne Pointer (Integer-Zahlen) in anderer Weise in Dateien speichern, können die Daten einer so übertragenen Datei von den TUSTEP-Programmen nicht direkt verarbeitet werden. Die Daten müssen in diesem Fall nach dem Übertragen mit dem TUSTEP-Kommando #RETTE (siehe Seite 216) in eine andere Datei kopiert (und evtl. automatisch wieder zurückkopiert) werden. Wird dies versäumt, wird eine Fehlermeldung ausgegeben, die besagt, dass die Datei »illegale Daten« enthalte.

Falls nach dem Übertragen beim Zugriff auf eine TUSTEP-Datei eine Fehlermeldung ausgegeben wird, die besagt, dass die Datei »zerstörte Daten« enthalte, wurde die Datei nicht binär übertragen.

Zum Übertragen bieten sich folgende Möglichkeiten an:

– USB-Speicher-Stick

Dateien können mit den Mitteln, die das jeweilige Betriebssystem bietet, auf bzw. von USB-Speicher-Sticks übertragen werden.

Innerhalb von TUSTEP können Dateien auch mit dem Makro \*DOWNLOAD (siehe Seite 58) auf einen USB-Speicher-Stick und mit dem Makro \*UPLOAD (siehe Seite 57) von einem USB-Speicher-Stick kopiert werden.

– FTP oder SCP

Dieser Weg kann gewählt werden, wenn die Rechner miteinander verbunden sind und eine entsprechende Userid mit Passwort bekannt ist. Beim Übertragen mit FTP ist zu beachten, dass mit der Anweisung »bin« sichergestellt wird, dass die Daten binär übertragen werden.

– E-mail

TUSTEP-Dateien dürfen nur als Anlage (attachment) verschickt werden. Werden die Daten trotzdem nicht binär übertragen, müssen die Daten wie oben beschrieben vor und nach dem Verschicken mit dem Kommando #UMWANDLE umgewandelt werden.

– Upload / Download

Unter Windows können in REMOTE-Sitzungen Dateien mit dem Makro \*UPLOAD (siehe Seite 57) zum Remote-Rechner und mit dem Makro \*DOWNLOAD (siehe Seite 58) vom Remote-Rechner übertragen werden.

Sollen mehrere TUSTEP-Dateien übertragen werden, kann es sinnvoll sein, diese zusammenzufassen, damit nur eine einzige Datei übertragen werden muss. Dazu kann mit dem Kommando #MBLABEL (siehe Seite 197) oder #DATEI (siehe Seite 128) eine »Band-Datei« eingerichtet werden. Die einzelnen Dateien können dann mit dem Kommando #MBAUSGABE (siehe Seite 187) oder #RETTE (siehe Seite 216) oder mit dem Makro \*MBUPDATE in diese Band-Datei kopiert werden. Nach dem Übertragen können sie mit dem Kommando #MBEINGABE (siehe Seite 189) oder #HOLE (siehe Seite 165) oder mit dem Makro \*MBUPDATE wieder aus der Band-Datei herauskopiert werden. Weitere Informationen zum Makro #\*MBUPDATE stehen unter »Datensicherung mit Band-Dateien« ab Seite 65.

# Datensicherung

---

Datensicherung auf USB-Speicher-Sticks . . . . .	63
Datensicherung mit Band-Dateien . . . . .	63

## Datensicherung auf USB-Speicher-Stick

Zur Datensicherung können mit dem Makro `*DOWNLOAD` (siehe Seite 58) Dateien auf einen USB-Speicher-Stick (oder auf andere Speichermedien) kopiert werden:

```
#*DOWNLOAD, datei, E:\datei
```

Soll eine so gesicherte Datei wiederhergestellt werden, kann sie mit dem Makro `*UPLOAD` (siehe Seite 58) wieder zurückkopiert werden:

```
#*UPLOAD, E:\datei, datei
```

In beiden Beispielen wurde davon ausgegangen, dass der USB-Speicher-Stick unter Windows den Laufwerksbuchstaben `E` hat bzw. unter Linux und macOS den Namen `E` hat.

Bei dieser Art von Sicherung auf USB-Speicher-Sticks ist folgendes zu beachten: Werden Dateien wiederholt (z. B. nach jeder größeren Änderung) mit dem gleichen Namen auf demselben USB-Speicher-Stick gesichert, so wird jeweils die bereits vorhandene Sicherungskopie überschrieben. Dies kann z. B. dadurch vermieden werden, dass für die Dateien auf dem USB-Speicher-Stick unterschiedliche Dateinamen (z. B. `text.1`, `text.2` usw. wenn die Datei `text` gesichert werden soll) verwendet werden. Es ist außerdem sinnvoll, nicht immer auf denselben USB-Speicher-Stick zu sichern, sondern abwechselnd auf verschiedene. Dann kann auf die vorletzte Sicherungskopie zurückgegriffen werden, wenn der USB-Speicher-Stick mit der letzten Sicherung aus irgendeinem Grund nicht mehr brauchbar ist.

In der Regel ist es bequemer, auf einem USB-Speicher-Stick für jedes Projekt eine Band-Datei einzurichten und die Dateien jeweils darin zu sichern. Dies ist im nachfolgenden Kapitel beschrieben. Auch dabei sollte nicht immer nur auf denselben USB-Speicher-Stick gesichert werden.

## Datensicherung mit Band-Dateien

Während der Frühzeit von TUSTEP war das Magnetband das einzige Speichermedium, auf dem größere Datenmengen gesichert werden konnten. Zusätzliche Dateien wurden immer hinter die letzte auf dem Magnetband stehende Datei geschrieben. Dabei durfte eine Datei auch mehrfach mit dem gleichen Namen auf einem Magnetband gesichert werden. Wurde eine Datei auf dem Magnetband überschrieben, waren damit (zumindest logisch) auch alle auf dem Magnetband folgenden Dateien überschrieben.

Ein Ersetzen einer Datei auf dem Magnetband war also (wenn es nicht die letzte auf dem Band war) praktisch nicht möglich. Diesem Nachteil stand aber der große Vorteil gegenüber, dass eine Datei wiederholt (z. B. nach jeder größeren Änderung) mit dem gleichen Dateinamen auf ein Magnetband gesichert werden konnte. Auf diese Weise konnte im Bedarfsfall bequem auf die Daten in den einzelnen Entwicklungsstadien der Datei zurückgegriffen werden.

Um diesen Komfort weiterhin zu bieten, können in TUSTEP an Stelle der Magnetbänder sogenannte »Band-Dateien« verwendet werden. Sie haben die oben be-

schriebenen Eigenschaften der Magnetbänder und können auch mit den gleichen Kommandos beschrieben und gelesen werden.

Bei den ursprünglich für Magnetbänder vorgesehenen Kommandos muss zur Spezifikation `BAND` der Name der Band-Datei und zur Spezifikation `GERAET` der Name einer System-Variablen angegeben werden. Für Band-Dateien muss diese System-Variablen den Pfad für das Verzeichnis (directory) enthalten, in der die Band-Datei steht. Dies kann z. B. die System-Variablen `HOME` sein, die das »home-directory« angibt; unter Windows kann statt einer System-Variablen auch ein Laufwerksbuchstabe (z. B. eines USB-Speicher-Sticks) angegeben werden.

In den folgenden Beispielen wird davon ausgegangen, dass die Datensicherung unter Windows auf einem USB-Speicher-Stick erfolgt, der mit dem Laufwerksbuchstaben `E` angesprochen wird:

Band-Datei mit dem Namen `sich01` einrichten:

```
#MBLABEL, sich01, GERAET=E
```

Datei mit dem Namen `erfass` in die Band-Datei retten:

```
#MBAUSGABE, sich01, , erfass, GERAET=E
```

Datei mit dem Namen `vorwort` von der Band-Datei holen:

```
#MBEINGABE, sich01, , vorwort, , +, GERAET=E
```

An Stelle der ursprünglich für Magnetbänder vorgesehenen Kommandos können auch folgende Kommandos verwendet werden:

Band-Datei mit dem Namen `sich01` einrichten bzw. anmelden:

```
#DATEI, -*sich01, TAPE-AP, TRAEGER=E
```

Datei mit dem Namen `erfass` in die Band-Datei retten:

```
#RETTE, erfass, sich01
```

Datei mit dem Namen `vorwort` von der Band-Datei holen:

```
#HOLE, sich01, vorwort, , +
```

Mit dem im Folgenden beschriebenen Makro `*MBUPDATE` kann das Sichern von Dateien in eine Band-Datei automatisiert werden. Mit ihm können z. B. alle Dateien eines vorgegebenen Projekts, die noch nicht in der aktuellen Fassung in der Band-Datei stehen, in die Band-Datei gesichert werden.



**Aufruf des Makros \*MBUPDATE**

#\*MBUPDATE

BAND	= name	Name der Band-Datei.
GERAET	= -	* Band-Datei ist angemeldet.
	= -STD-	Pfad für die Band-Datei ist durch die System-Variable TUSTEP_MT1 vorgegeben.
	= name	Name der System-Variablen, die den Pfad für die Band-Datei enthält.  Unter Windows kann auch direkt der Laufwerksbuchstabe angegeben werden, falls der Pfad für die Band-Datei keinen Verzeichnisnamen enthält.
MODUS	= RETTEN	Dateien in die Band-Datei retten.
	= HOLEN	Dateien aus der Band-Datei holen.
	= TESTEN	Dateien mit der Band-Datei vergleichen.
	= TESTEN/RETTE	Dateien mit der Band-Datei vergleichen und bei Ungleichheit jeweils retten.
PROJEKT	= name	Name des Projekts, von dem Dateien gerettet bzw. in das Dateien geholt werden sollen.
	= +	* Dateien vom aktuellen Projekt retten bzw. in das aktuelle Projekt holen.
	= -STD-	Dateien des Projekts retten bzw. in das Projekt holen, das beim Initialisieren der TUSTEP-Sitzung eingestellt wurde; dieses Projekt ist durch die System-Variable TUSTEP_PRJ vorgegeben.
	= -	Die Dateien retten bzw. holen, die keinem Projekt zugeordnet sind.
TRAEGER	= name	Name der System-Variablen, die den Pfad für die Dateien enthält, die gerettet/geholt werden sollen.  Unter Windows kann auch direkt der Laufwerksbuchstabe angegeben werden, falls der Pfad keinen Verzeichnisnamen enthält.
	= -STD-	* Die Dateien retten bzw. holen, die sich auf dem Standard-Träger für permanente Dateien befinden; dieser ist durch die System-Variable TUSTEP_DSK vorgegeben.
	= -	
AUSWAHL	= -STD-	* Alle Dateien außer SCRATCH-Dateien
	= TUSTEP	Nur TUSTEP-Dateien auswählen

---

	= SYSTEM	Nur Fremd-Dateien auswählen
	= SCRATCH	Nur SCRATCH-Dateien auswählen
	= TUSTEP/SCRATCH	Nur SCRATCH-TUSTEP-Dateien auswählen
	= SYSTEM/SCRATCH	Nur SCRATCH-Fremd-Dateien auswählen
POSITIV	= -	* Normalfall.
	= ...	Nur die Dateien retten/holen, deren Namen mindestens einem der angegebenen Muster entsprechen.  Die Besonderheiten bei der Angabe von Mustern sind unter »Muster zur Auswahl der Namen/Dateien« auf Seite 178 beschrieben.
NEGATIV	= -	* Normalfall.
	= ...	Nur die Dateien retten/holen, deren Namen keinem der angegebenen Muster entsprechen.  Die Besonderheiten bei der Angabe von Mustern sind unter »Muster zur Auswahl der Namen/Dateien« auf Seite 178 beschrieben.
NAMEN	= -	Keine Auswahl über Dateinamen.
	= name	Namen der Dateien, die gerettet/geholt werden sollen; die einzelnen Dateinamen müssen durch Apostroph getrennt sein.
FRAGEN	= -	* Dateien retten/holen ohne anzufragen.
	= +	Jeweils fragen, ob eine Datei gerettet bzw. geholt werden soll.
ANTWORT	= ja	* leere Antwort bei Anfragen heißt ja.
	= nein	leere Antwort bei Anfragen heißt nein.
DATUM	= 00.00.0000	* Keine Auswahl über das Änderungsdatum.
	= dd.mm.yyyy	Nur solche Dateien berücksichtigen, die seit dem angegebenen Datum geändert wurden.
	= yyyy-mm-dd	Nur solche Dateien berücksichtigen, die seit dem angegebenen Datum geändert wurden.
REDUZIEREN	= -	* Protokoll nicht reduzieren
	= +	Keine Meldung ausgeben, wenn die Daten aktuell/identisch sind.
EINRICHTEN	= -STD-	* Falls Band-Datei nicht vorhanden, nachfragen, ob sie eingerichtet werden soll.
	= -	Falls Band-Datei nicht vorhanden, nicht einrichten, nur Fehlermeldung ausgeben.

= + Falls Band-Datei nicht vorhanden, einrichten ohne vorher nachzufragen.

## Leistung

Mit dem Makro \*MBUPDATE können Dateien des angegebenen Projekts (Verzeichnisses) in eine Band-Datei gerettet bzw. Dateien einer Band-Datei in das angegebene Projekt geholt werden, oder es kann geprüft werden, ob Dateien eines angegebenen Projekts mit den aktuellen Daten in einer Band-Datei stehen.

Falls nicht alle Dateien mit einem TUSTEP-konformen Namen (andere bleiben in jedem Fall unberücksichtigt) gerettet bzw. geholt bzw. überprüft werden sollen, kann eine Dateiauswahl über folgende Kriterien erfolgen:

- alle Dateien, die TUSTEP-, Fremd- und/oder SCRATCH-Dateien sind,
- alle Dateien, in deren Namen bestimmte Zeichenfolgen vorkommen,
- alle Dateien, in deren Namen bestimmte Zeichenfolgen nicht vorkommen,
- alle Dateien, deren Name explizit angegeben ist.

Die Daten einer Datei werden nur dann in die Band-Datei gerettet, wenn die Band-Datei noch keine Datei gleichen Namens enthält oder wenn die Daten der gleichnamigen Datei ein älteres Änderungsdatum haben. Beim Retten mit dem Makro \*MBUPDATE werden die Dateien immer ans Ende der Band-Datei geschrieben; enthält die Band-Datei schon gleichnamige Dateien, so bleiben diese erhalten.

Die Daten einer Datei werden nur dann aus der Band-Datei geholt, wenn die Datei, in die sie geholt werden sollen, ein älteres Änderungsdatum hat. Enthält die Band-Datei Dateien gleichen Namens, wird jeweils nur die Datei berücksichtigt, deren Daten das jüngste Änderungsdatum haben. Wenn beim Holen noch keine gleichnamige Datei im angegebenen Projekt existiert, wird sie automatisch eingerichtet. Sollen Daten von der Band-Datei in eine Datei geholt werden, die eine neueres Änderungsdatum hat, so muss die Datei (nicht nur die Daten!) zuvor gelöscht werden.

Falls die angegebene Band-Datei auf dem angegebenen Gerät nicht vorhanden ist, wird nachgefragt, ob sie eingerichtet werden soll. Die Nachfrage bzw. das Einrichten kann mit der Spezifikation EINRICHTEN unterdrückt werden.

Für die Anfrage, ob eine Datei gerettet/geholt werden soll, gibt es fünf zulässige Antworten, die abgekürzt werden können:

JA	Daten retten/holen, weiterhin nachfragen.
NEIN	Daten nicht retten/holen, weiterhin nachfragen.
ALLE	Datei retten/holen, nicht mehr nachfragen.
KEINE	Datei nicht retten/holen, Makro beenden
VERGLEICHEN	Daten mit den Originaldaten vergleichen und die Unterschiede anzeigen, Anfrage wiederholen.

**Hinweise:**

Die Band-Datei muss entweder angemeldet sein, oder es muss zur Spezifikation GERAET eine System-Variable angegeben werden, die den Pfad zur Band-Datei enthält; im letzteren Fall ist die Angabe eines Projekts zusammen mit dem Namen der Band-Datei nicht erlaubt und es wird auch kein Projektname ergänzt.

Falls mit dem Makro `*MBUPDATE` von der Band-Datei Daten geholt werden sollen, deren Änderungsdatum älter ist als das der Daten in der Datei, so muss die Datei zuvor gelöscht werden; sie wird dann vom Makro `*MBUPDATE` wieder eingerichtet. Eine andere Möglichkeit ist, die Daten mit dem Kommando `#MBEINGABE` (siehe Seite 189) oder `#HOLE` (siehe Seite 165) aus der Band-Datei zu holen.

Eine Band-Datei kann mit den Kommandos `#MBLABEL` (siehe Seite 197) eingerichtet und gelöscht werden; außerdem kann sie auch mit dem Kommando `#DATEI` (siehe Seite 128) eingerichtet und mit dem Kommando `#LOESCHE` (siehe Seite 182) gelöscht werden.

Mit dem Kommando `#MBINFORMIERE` (siehe Seite 191) und mit dem Kommando `#LISTE` (siehe Seite 178) erhält man eine Liste der Dateien, die in der Band-Datei enthalten sind.

Sind in der Band-Datei mehrere Dateien mit gleichem Namen enthalten, so kann mit dem Makro `*MBUPDATE` nur die Datei geholt werden, deren Daten das jüngste Änderungsdatum haben; mit den Kommandos `#MBEINGABE` (siehe Seite 189) und `#HOLE` (siehe Seite 165) kann jede Datei aus der Band-Datei geholt werden.

Mit den Kommandos `#MBAUSGABE` (siehe Seite 187) und `#RETTE` (siehe Seite 216) können einzelne Dateien in eine Band-Datei gerettet werden; mit dem Kommando `#MBKOPIERE` (siehe Seite 193) können Band-Dateien kopiert werden.

**Beispiele:**

Die Band-Datei heißt `sammel` und steht unter Windows im Root-Verzeichnis des Laufwerks `D`, unter Linux und macOS in dem mit der System-Variablen `D` vorgegebenen Verzeichnis. Sie wurde mit `#MBLABEL, sammel, , , d, +` eingerichtet. Falls sie noch nicht eingerichtet ist, wird nachgefragt, ob sie eingerichtet werden soll.

```
#*MBUPDATE, sammel, d, RETTEN
```

rettet alle permanenten Dateien mit TUSTEP-konformen Namen des aktuellen Projekts, die in der Band-Datei noch nicht vorhanden sind oder ein älteres Änderungsdatum haben.

```
#*MBUPDATE, sammel, d, RETTEN, NAME=beispiel
```

rettet die permanente Datei mit dem Namen `beispiel` in die Band-Datei.

```
#*MBUPDATE, sammel, d, RETTEN, AUSWAHL=SCRATCH, NAME=beispiel
```

rettet die temporäre Datei mit dem Namen `beispiel` in die Band-Datei.

```
#*MBUPDATE, sammel, d, HOLEN
```

holt alle in der Band-Datei enthaltenen Dateien und speichert sie im aktuellen Ver-

zeichnis. Dateien, die in diesem Verzeichnis noch nicht existieren, werden eingerichtet. Dateien, die in diesem Verzeichnis schon existieren, werden nur geholt, wenn sie in der Band-Datei ein neueres Änderungsdatum haben. Enthält die Band-Datei mehrere Dateien mit dem gleichen Namen, wird jeweils nur die mit dem neuesten Änderungsdatum berücksichtigt.

```
#*MBUPDATE, sammel, d, HOLEN, NAME=beispiel
```

holt die Datei mit dem Namen `beispiel` aus der Band-Datei und speichert die Daten in die gleichnamige permanente Datei im aktuellen Projekt, falls diese Datei nicht schon Daten neueren Datums enthält.

```
#*MBUPDATE, sammel, d, HOLEN, AUSWAHL=SCRATCH, NAME=beispiel
```

holt die Datei mit dem Namen `beispiel` aus der Band-Datei und speichert die Daten in die gleichnamige temporäre Datei im aktuellen Projekt, falls diese Datei nicht schon Daten neueren Datums enthält.



# Systemumgebung

---

Allgemeines . . . . .	73
System-Variable TUSTEP_HST . . . . .	73
System-Variable TUSTEP_TTL . . . . .	73
System-Variablen TUSTEP_DSK, TUSTEP_SCR . . . . .	73
System-Variable TUSTEP_PRJ . . . . .	74
System-Variable TUSTEP_USR . . . . .	74
System-Variable TUSTEP_NAM . . . . .	74
System-Variable TUSTEP_MEM . . . . .	74
System-Variablen TUSTEP_COLS, TUSTEP_ROWS . . . . .	74
System-Variable TUSTEP_MDS . . . . .	75
System-Variable TUSTEP_INI . . . . .	75
System-Variable TUSTEP_CMD . . . . .	75
System-Variable TUSTEP_CGI . . . . .	75
System-Variable TUSTEP_PRN . . . . .	75
System-Variable TUSTEP_LIB . . . . .	75
System-Variable TUSTEP_LPR . . . . .	76
System-Variable TUSTEP_UCB . . . . .	76
System-Variable TUSTEP_PKZ . . . . .	76
System-Variable TUSTEP_RCP . . . . .	76



## Allgemeines

In vielen Fällen ist es nicht nötig, sich mit den in diesem Kapitel beschriebenen Anpassungsmöglichkeiten von TUSTEP zu befassen, da die Voreinstellungen entsprechend gewählt sind. Die Anpassung erfolgt über Variablen, die beim Definieren der TUSTEP-Sitzung mit dem Makro \*DESI definiert werden.

Diese Variablen (meist Umgebungs- oder Environment-Variablen genannt) dürfen nicht mit den innerhalb von TUSTEP definierten Variablen verwechselt werden. Damit klar ist, welche Art von Variablen jeweils gemeint ist, werden erstere »System-Variablen« und letztere »TUSTEP-Variablen« genannt.

In TUSTEP können nur die System-Variablen verwendet werden, deren Namen aus Großbuchstaben und »\_« bestehen, mit einem Buchstaben beginnen und nicht mit »\_« enden.

Wird eine TUSTEP-Sitzung unterbrochen und später wieder fortgesetzt, so dürfen die System-Variablen, die von TUSTEP ausgewertet werden, zwischendurch nicht verändert werden bzw. müssen wieder auf den gleichen Wert zurückgesetzt werden, bevor die TUSTEP-Sitzung fortgesetzt wird.

### System-Variable TUSTEP\_HST

Um in Ablaufprotokollen feststellen zu können, auf welchem Rechner gearbeitet wurde, kann TUSTEP über die System-Variable TUSTEP\_HST der Name des jeweiligen Rechners (host) mitgeteilt werden. Dieser Name wird dann in den Start- und Endmeldungen der Programme mit ausgegeben. Er kann in Makros (mit der Anweisung FETCH) abgefragt werden und zur Steuerung von Makros verwendet werden. Der Standardwert ist ein vom jeweiligen Betriebssystem vorgegebener Name.

### System-Variable TUSTEP\_TTL

Der Inhalt dieser System-Variablen wird in der Kopfzeile des TUSTEP-Fensters als Titel angezeigt.

### System-Variablen TUSTEP\_DSK, TUSTEP\_SCR

Die System-Variable TUSTEP\_DSK und TUSTEP\_SCR werden beim Einrichten (z. B. mit dem Kommando #DATEI) und Anmelden von Dateien (z. B. mit dem Kommando #ANMELDE) sowie beim Informieren über vorhandene Dateien (z. B. mit dem Kommando #LISTE) ausgewertet, wenn dabei für den Träger der Dateien nichts oder »-STD-« (Voreinstellung) angegeben wird.

Für permanente Dateien wird die System-Variable TUSTEP\_DSK, für temporäre Dateien die System-Variable TUSTEP\_SCR ausgewertet. Diese Voreinstellung für den Träger kann mit TUSTEP-Kommandos nicht verändert werden. Es muss ggf. eine andere System-Variable mit entsprechendem Inhalt angegeben werden.

**System-Variable** TUSTEP\_PRJ

Mit der System-Variablen wird der ursprüngliche Projektname (siehe »Ergänzen des Projektnamens« Seite 26) festgelegt. Er dient zur Ergänzung von Dateinamen, bis mit dem Kommando #DEFINIERE (siehe Seite 132) ein anderer Projektname definiert wird; der Inhalt der System-Variablen TUSTEP\_PRJ bleibt dabei unverändert.

**System-Variable** TUSTEP\_USR

Die Benutzeridentifikation (userid, loginid) wird TUSTEP über die System-Variable TUSTEP\_USR mitgeteilt. Sie kann u. a. als Kennzeichnung auf dem Umschlag (Deck- und Endblatt) bei Druckausgaben dienen.

**System-Variable** TUSTEP\_NAM

Eine weitere Kennzeichnung auf dem Deckblatt von Ausdrucken ist der Name. Er kann mit dem Kommando #DEFINIERE eingestellt werden. Die Voreinstellung dafür kann mit der System-Variablen TUSTEP\_NAM bestimmt werden.

**System-Variable** TUSTEP\_MEM

Für jede TUSTEP-Sitzung wird eine Scratch-Datei angelegt, in der für die jeweilige Sitzung Informationen (z. B. welche Dateien angemeldet sind) gemerkt werden. Wie alle Scratch-Dateien wird diese Datei in dem durch die System-Variablen TUSTEP\_SCR und TUSTEP\_PRJ festgelegten Verzeichnis angelegt. Da mehrere Sitzungen gleichzeitig existieren können, muss TUSTEP wissen, welche Datei die Informationen für die aktuelle Sitzung enthält. Dazu wird von TUSTEP die System-Variable TUSTEP\_MEM definiert. Sie enthält einen Verweis auf diese Datei in Form einer 8-stelligen Zahl `mmm00000`, wobei `mmm` die Nummer der Sitzung ist. Diese Zahl ist gleichzeitig Bestandteil des Namens, den diese Datei außerhalb von TUSTEP hat. Der Name solcher Dateien hat die Form `mmm00000.tsf`.

Die System-Variablen TUSTEP\_SCR, TUSTEP\_PRJ und TUSTEP\_MEM identifizieren eine TUSTEP-Sitzung, indem sie zusammen auf die Datei verweisen, die die entsprechenden Informationen enthält.

**System-Variablen** TUSTEP\_COLS, TUSTEP\_ROWS

Die Größe des TUSTEP-Fensters kann mit dem Kommando #DEFINIERE eingestellt werden. Mit der System-Variablen TUSTEP\_COLS kann die Voreinstellung für die Breite (Anzahl Spalten) bestimmt werden, mit der System-Variablen TUSTEP\_ROWS die Voreinstellung für die Höhe (Anzahl Zeilen). Diese beiden Variablen werden nur beim Initialisieren einer TUSTEP-Sitzung ausgewertet.

Unter Linux kann aus technischen Gründen die tatsächliche Größe des TUSTEP-Fensters nicht automatisch angepasst werden; sie muss zusätzlich mit der Maus von Hand angepasst werden.

**System-Variable** TUSTEP\_MDS

Die System-Variable TUSTEP\_MDS muss den Wert DIALOG haben, wenn TUSTEP im Dialog verwendet wird, den Wert REMOTE, wenn TUSTEP in einer REMOTE-Sitzung verwendet wird, und den Wert BATCH, wenn TUSTEP im Batch (z. B. beim Aufruf von einem WWW-Server) verwendet wird.

**System-Variable** TUSTEP\_INI

Die INI-Datei (siehe Seite 89) ist eine TUSTEP-Datei mit dem vorgegebenen Namen TUSTEP.INI. Soll als INI-Datei eine ASCII-Datei oder eine Datei mit einem anderen Namen verwendet werden, so muss der Name dieser Datei mit der System-Variablen TUSTEP\_INI vorgegeben werden; andernfalls sollte diese System-Variable nicht definiert werden. Falls die INI-Datei eine ASCII-Datei ist, kann sie in ASCII oder ISO-8859-1 codiert sein.

**System-Variable** TUSTEP\_CMD

Die Verwendung der System-Variablen TUSTEP\_CMD ist nur sinnvoll, wenn TUSTEP als System-Kommando im Batch-Modus aufgerufen wird. Mit ihr kann der Name des Segments in der INI-Datei (siehe Seite 89) festgelegt werden, dessen Inhalt als CMD-Makro ausgeführt werden soll (siehe »Starten von TUSTEP als System-Kommando« ab Seite 89).

**System-Variable** TUSTEP\_CGI

Die Verwendung der System-Variablen TUSTEP\_CGI ist nur sinnvoll, wenn TUSTEP über WWW durch einen WWW-Server im Batch-Modus aufgerufen wird. Mit ihr wird der Name des Segments in der INI-Datei (siehe Seite 89) festgelegt, dessen Inhalt als CGI-Makro ausgeführt werden soll. Hinter dem Segmentnamen muss zusätzlich der Code angegeben werden, in dem die vom CGI-Makro erzeugte Standard-Ausgabe codiert werden soll. Als Codes sind ISO-8859-1 und UTF-8 vorgesehen. Segmentname und Code-Angabe müssen durch einen Schrägstrich getrennt werden.

In Sonderfällen können hinter der Code-Angabe zwei Schrägstriche folgen. In diesem Fall wird die unbedingt notwendige Content-type-Information nicht automatisch von TUSTEP ausgegeben, sondern muss vom CGI-Makro ausgegeben werden.

**System-Variable** TUSTEP\_PRN

Mit dieser System-Variablen kann der Name des Druckers festgelegt werden, auf dem ausgedruckt werden soll, wenn bei den Kommandos #DRUCKE und #PROTOKOLL zur Spezifikation GERAET nichts oder -STD- (Voreinstellung) angegeben ist.

**System-Variable** TUSTEP\_LIB

Die System-Variable TUSTEP\_LIB muss die Trägerangabe für die zu TUSTEP gehörenden Dateien und Programme enthalten. Als Projekt wird für diese Dateien und Programme immer tustep angenommen. Wenn sie sich also z. B. im Verzeichnis

C:\PROGRAMME\TUSTEP bzw. im Verzeichnis /opt/tustep befinden, muss TUSTEP\_LIB den Wert C:\PROGRAMME bzw. /opt haben.

### System-Variable TUSTEP\_LPR

Die System-Variable TUSTEP\_LPR wird nur unter Linux und macOS verwendet. Sie definiert das System-Kommando zum Drucken mit den Kommandos #DRUCKE und #PROTOKOLL. Vor der Ausführung des System-Kommandos werden die darin enthaltenen Platzhalter von TUSTEP durch entsprechende Werte ersetzt. Es gibt folgende Platzhalter:

<GERAET> Angabe zur Spezifikation GERAET  
<ANZAHL> Angabe zur Spezifikation ANZAHL  
<NAME> Mit dem Kommando #DEFINIERE eingestellter Name  
<ZUSATZ> Inhalt der zur Spezifikation ZUSATZ angegebenen System-Variablen  
<DATEI> Name der intern erzeugten Datei mit den Codes zur Druckersteuerung, die zum Drucker geschickt werden soll; falls dieser Platzhalter nicht im System-Kommandos vorhanden ist, wird der Name der Datei am Ende des System-Kommandos eingefügt.

### System-Variable TUSTEP\_UCB

Die System-Variable TUSTEP\_UCB wird nur unter Linux und macOS verwendet. Mit ihr kann eine Datei vorgegeben werden, die als Zwischenablage dient. Damit kann im TUSTEP-Editor und in Makros auch unter Linux und macOS eine Zwischenablage wie unter Windows benutzt werden. Die Variable muss den vollständigen Pfad einschließlich des Namens der Datei, wie er auf Betriebssystemebene erforderlich ist, enthalten. Falls diese Variable nicht definiert ist, wird eine interne Datei als Zwischenablage verwendet.

### System-Variable TUSTEP\_PKZ

Die System-Variable TUSTEP\_PKZ wird nur unter Linux und macOS verwendet. Mit ihr kann der für die jeweilige TUSTEP-Sitzung zuständige Prozess (das ausführende Programm) gekennzeichnet werden. Wenn diese System-Variable nicht definiert ist, wird der Prozess nur mit der Sitzungsnummer (siehe System-Variable TUSTEP\_MEM Seite 74) gekennzeichnet.

### System-Variable TUSTEP\_RCP

Die System-Variable TUSTEP\_RCP wird nur unter Windows in REMOTE-Sitzungen verwendet und von TUSTEP automatisch definiert. Sie enthält die Nummer der Code-Page, die von Windows in TUSTEP-Fenstern verwendet wird.

# TUSTEP-Aufruf

---

Aufruf von TUSTEP . . . . .	79
Definieren und Starten einer TUSTEP-Sitzung . . . . .	80
Windows: TUSTEP auf einem Linux-Server . . . . .	82
Linux: TUSTEP auf einem Linux-Server . . . . .	84
macOS: TUSTEP auf einem Linux-Server . . . . .	85
Beginnen einer TUSTEP-Sitzung . . . . .	87
Unterbrechen einer TUSTEP-Sitzung . . . . .	87
Fortsetzen einer TUSTEP-Sitzung . . . . .	88
Beenden einer TUSTEP-Sitzung . . . . .	88
INI-Datei . . . . .	89
Starten von TUSTEP als System-Kommando . . . . .	89
Starten von TUSTEP durch einen WWW-Server . . . . .	98
Starten des TUSTEP-Editors unter Windows . . . . .	103
Starten des TUSTEP-Editors unter Linux . . . . .	104
Starten des TUSTEP-Editors unter macOS . . . . .	105

## Aufruf von TUSTEP

### Windows 7:

Start → Programme → TUSTEP

### Windows 8:

Auf dem Start-Bildschirm unten links auf den in einem weißen Kreis nach unten zeigenden Pfeil klicken (der Pfeil wird angezeigt, sobald die Maus bewegt wird). Danach wird eine Liste mit allen Programmen angezeigt. Nun entweder TUSTEP in der Liste anklicken (die Liste kann mit dem Mausrad verschoben werden) oder »tustep« eintippen (die Eingabemarke springt automatisch ins Suchfeld) und mit der Eingabetaste bestätigen.

### Windows 10:

Start-Menü aufrufen (mit Windows-Taste oder mit einem Klick auf das Windows-Symbol unten links in der Taskleiste). Nun entweder TUSTEP in der Liste anklicken (die Liste kann mit dem Mausrad verschoben werden) oder »tustep« eintippen (die Eingabemarke springt automatisch ins Suchfeld) und »TUSTEP Desktop.App« mit der Eingabetaste bestätigen.

### Linux:

Je nach graphischer Benutzeroberfläche auf unterschiedliche Weise (meist über ein Menü) oder in einem Terminal-Fenster mit der Anweisung

```
/opt/tustep/start
```

falls TUSTEP im Verzeichnis `/opt/tustep` installiert ist.

### macOS:

Im Launchpad durch einen Klick auf TUSTEP oder in der Programmliste im Finder durch einen Doppelklick auf TUSTEP.app.

## Definieren und Starten einer TUSTEP-Sitzung

Um mit TUSTEP arbeiten zu können, müssen eine oder mehrere TUSTEP-Sitzungen (z. B. für jedes Projekt oder jedes Aufgabengebiet eine Sitzung) definiert werden.

Wird TUSTEP wie oben angegeben gestartet, so wird automatisch das Makro \*DESI aufgerufen. Nach einer kurzen Anleitung wird folgende Eingabemaske angezeigt: –

The screenshot shows a dialog box titled "TUSTEP-Sitzung". At the top, there are four buttons: "TUSTEP" (highlighted in yellow), "REMOTE", "Standard", and "Starten". Below these are several input fields for defining session variables. The first field is for the session name, with "Projekte" entered. Other fields include TUSTEP\_TTL, TUSTEP\_DSK, TUSTEP\_SCR, TUSTEP\_PRJ, TUSTEP\_USR, TUSTEP\_NAM, TUSTEP\_MEM, TUSTEP\_COLS, and TUSTEP\_ROWS. At the bottom, there are buttons for "Desktop-Icon", "Einrichten", "Ändern", "Löschen", and "Beenden". A text box at the very bottom says "Bitte den Namen der Sitzung eintragen".

Diese Eingabemaske kann auch in jeder TUSTEP-Sitzung mit dem Kommando #\*DESI aufgerufen werden; weitere Informationen über das Makro werden mit dem Kommando #INFORMIERE, \*DESI ausgegeben. Falls jedoch noch keine Sitzung definiert ist, kann diese Maske nur wie auf Seite 79 angegeben aufgerufen werden.

Im ersten Feld der Eingabemaske muss ein Name für die Sitzung angegeben werden. Er darf aus 1 bis 12 Zeichen (Buchstaben a bis z, Ziffern, Minuszeichen und »\_«) bestehen, muss mit einem Buchstaben beginnen und darf nicht mit Minuszeichen oder »\_« enden.

In den folgenden Feldern werden die System-Variablen für die jeweilige Sitzung definiert. Welche Bedeutung die vorgegebenen Variablen haben, ist im Kapitel »Systemumgebung« ab Seite 71 beschrieben. In der unteren Hälfte können eigene System-Variablen (z. B. für Trägerangaben und Druckernamen) definiert werden.

Wird nur eine TUSTEP-Sitzung definiert, genügen in der Regel die automatisch eingestellten Werte; leere Felder können leer gelassen werden. Werden Sitzungen für verschiedene Projekte definiert, so empfiehlt es sich, zur System-Variablen TUSTEP\_PRJ den jeweiligen Projektnamen anzugeben.

Soll unter Windows mit TUSTEP auf einem anderen als dem vom Betriebssystem voreingestellten Drucker ausgegeben werden, sollte dafür noch eine eigene System-Variable (z. B. mit dem Namen »DJ«) definiert werden, die den vom Betriebssystem verwendeten Druckernamen (z. B. »HP DeskJet Printer«) enthält. Der Name dieser System-Variablen muss beim Drucken (z. B. mit dem Kommando #DRUCKE) als Gerät angegeben werden.



Wird mehr als eine TUSTEP-Sitzung definiert, so ist darauf zu achten, dass sich die Sitzungen nicht nur im Namen, sondern auch in mindestens einer der Angaben zu den System-Variablen `TUSTEP_SCR`, `TUSTEP_PRJ` und `TUSTEP_MEM` unterscheiden; andernfalls lässt sich die Sitzung nicht einrichten.

In der Spalte am rechten Rand werden die schon definierten TUSTEP-Sitzungen aufgelistet. Um die Definition einer solchen Sitzung in der Eingabemaske anzuzeigen, kann der Name dieser Sitzung entweder in dieser Spalte mit der linken Maustaste angeklickt oder im ersten Feld eingetragen und mit der Return-Taste bestätigt werden.

In der untersten Zeile der Eingabemaske wird jeweils ein kurzer Hinweis gegeben, was in das Feld, in dem der Cursor gerade steht, eingetragen werden soll.

In der obersten und in der vorletzten Zeile befinden sich Schaltflächen. Eine Schaltfläche kann aktiviert werden, indem sie entweder mit der linken Maustaste angeklickt wird oder indem der Cursor mit der Tabulatortaste auf die Schaltfläche positioniert und dann die Return-Taste gedrückt wird.

Nachdem alle erforderlichen Werte in die Eingabemaske eingetragen sind, kann die Sitzung durch Aktivieren der Schaltfläche »Einrichten« eingerichtet werden; d. h. die Angaben in der Eingabemaske werden gespeichert.

Danach kann die Sitzung durch Aktivieren der Schaltfläche »Starten« gestartet werden. Um eine TUSTEP-Sitzung bequemer starten zu können, kann durch Aktivieren der Schaltfläche »Desktop-Icon« ein Icon auf dem Desktop zum Aufruf der Sitzung angelegt werden.

Die Schaltfläche »REMOTE« wird nur unter Windows angezeigt und zum Einrichten von REMOTE-Sitzungen benötigt. Dies ist im nachfolgenden Kapitel beschrieben.

Mit der Schaltfläche »Standard« können die Eingabefelder mit Standard-Werten belegt werden.

Soll die Definition einer Sitzung geändert werden, müssen die zuletzt gespeicherten Werte wie oben angegeben in die Eingabemaske geholt werden. Die Werte für die System-Variablen `TUSTEP_SCR`, `TUSTEP_PRJ` und `TUSTEP_MEM` dürfen nur geändert werden, wenn die Sitzung zuvor (mit dem Kommando `#BEEENDE`) beendet wurde. Die Änderungen werden erst beim Aktivieren der Schaltfläche »Ändern« abgespeichert. Die geänderten Werte gelten nur für nachfolgende Aufrufe dieser Sitzung; sie gelten also nicht sofort, falls diese Sitzung gerade aktiv ist.

Soll eine Sitzung gelöscht werden, müssen die zuletzt gespeicherten Werte wie oben angegeben in die Eingabemaske geholt und die Schaltfläche »Löschen« aktiviert werden. Bevor eine Sitzung gelöscht werden kann, muss sie (z. B. mit dem Kommando `#BEEENDE`) beendet worden sein.

Soll der Name einer Sitzung geändert werden, müssen die zuletzt gespeicherten Werte wie oben angegeben in die Eingabemaske geholt und die Sitzung durch Aktivieren der Schaltfläche »Löschen« gelöscht werden; der Inhalt der Felder bleibt dabei erhalten. Danach kann der Name der Sitzung im ersten Feld geändert und die Sitzung durch Aktivieren der Schaltfläche »Einrichten« wieder eingerichtet werden.

Durch Aktivieren der Schaltfläche »Beenden« wird das Makro `*DESI` beendet.

Nach dem Starten einer TUSTEP-Sitzung wird ein TUSTEP-Fenster geöffnet, und es erfolgt eine Aufforderung, Kommandos einzugeben, mit denen TUSTEP mitgeteilt wird, was zu tun ist. Um z. B. weitere Informationen zu erhalten, kann mit dem Kommando `#*ZEBE` (ZEige BEschreibung) das TUSTEP-Handbuch angezeigt werden.

Eine TUSTEP-Sitzung kann durch Drücken der Escape-Taste unterbrochen oder durch Eingeben des Kommandos `#BEENDE` beendet werden.

Windows: Zum Unterbrechen einer TUSTEP-Sitzung kann auch die kleine Schaltfläche mit dem X in der rechten oberen Ecke des Fensters angeklickt werden.

Linux: Zum Unterbrechen einer TUSTEP-Sitzung sollte keinesfalls die kleine Schaltfläche mit dem X in der rechten oberen Ecke des Fensters angeklickt werden. Dadurch würde die TUSTEP-Sitzung abgebrochen und das Fenster geschlossen, aber offene Dateien würden dabei nicht korrekt abgeschlossen und Daten gingen möglicherweise verloren.

macOS: Zum Unterbrechen einer TUSTEP-Sitzung sollte keinesfalls der rote Punkt in der linken oberen Ecke des Fensters angeklickt werden. Dadurch würde die TUSTEP-Sitzung abgebrochen und das Fenster geschlossen, aber offene Dateien würden dabei nicht korrekt abgeschlossen und Daten gingen möglicherweise verloren.

## Windows: TUSTEP auf einem Linux-Server

Im Folgenden wird davon ausgegangen, dass TUSTEP auf dem Linux-Server im Verzeichnis `/opt/tustep` installiert ist.

Um von einem Windows-Rechner aus in einer Remote-Sitzung auf einem Linux-Server mit TUSTEP arbeiten zu können, muss auf dem Windows-Rechner mit dem Makro `*DESI` eine Remote-Sitzung definiert werden. Dann können Tastatur und Maus in gleicher Weise wie bei einer normalen (lokalen) TUSTEP-Sitzung unter Windows verwendet werden.

Definition und Aufruf einer REMOTE-Sitzung geschieht in gleicher Weise wie bei einer lokalen TUSTEP-Sitzung unter Windows; dies ist im vorhergehenden Kapitel beschrieben. Bei der Definition sind jedoch einige zusätzliche Angaben notwendig.

Nachdem im ersten Feld der Eingabemaske der Name der TUSTEP-Sitzung eingetragen wurde, muss als nächstes die Schaltfläche »REMOTE« aktiviert werden. Daraufhin wird folgende Eingabemaske angezeigt:

Rechts oben kann ausgewählt werden, ob die Kommunikation zwischen Windows- und Linux-Server verschlüsselt (SSH) oder unverschlüsselt (REXEC) erfolgen soll. In den folgenden Feldern können Werte eingetragen werden. Für das Feld `HOST` ist eine Angabe obligat. Für die Felder `USERID` und `PASSWD` ist die Angabe optional; fehlende Werte werden beim Aufruf der REMOTE-Sitzung nachgefordert.

- `HOST` Name oder Adresse des Linux-Servers (obligat).
- `PORT` Portnummer, falls nicht der Standard-Port für SSL-Verbindungen des Linux-Servers verwendet werden soll.
- `USERID` Benutzerkennung auf dem Linux-Server (optional).

TUSTEP-Sitzung	<input type="text"/>	<input type="text"/>	Verbindungsart	<input checked="" type="radio"/> SSH	<input type="radio"/> REXEC
HOST	=	<input type="text"/>	PORT	=	<input type="text"/>
USERID	=	<input type="text"/>	SHELL	=	bash ?
PASSWD	=	<input type="text"/>	TIMEOUT	=	<input type="text"/>
			DELAY	=	5
COMMAND	=	<input type="text"/>			
TUSTEP_LIB	=	/opt			
TUSTEP_LPR	=	lpr -P<GERAET> -#<ANZAHL> -C<NAME> -r <ZUSATZ>			
TUSTEP_UCB	=	<input type="text"/>	Zurück		
Bitte Name oder Adresse des Rechners eintragen					

- PASSWD Passwort für die Benutzererkennung (optional).
- TIMEOUT Wartezeit (in Minuten) nach der letzten Eingabe, bevor die Sitzung automatisch unterbrochen wird (optional).
- SHELL Name der Login-Shell, die auf dem Linux-Server für die jeweilige Benutzererkennung eingestellt ist.
- DELAY Wartezeit (in Sekunden) zur Synchronisation der Verbindung, bevor TUSTEP auf dem Linux-Server gestartet wird; bei fehlender Angabe wird 5 Sekunden gewartet.
- COMMAND Zusätzliche Linux-Anweisung vor dem Aufruf von TUSTEP (optional).

Mit den nächsten drei Feldern werden System-Variablen für die jeweilige Sitzung auf dem Linux-Server definiert. Der Wert für TUSTEP\_LIB muss in jedem Fall eingetragen werden, der Wert für TUSTEP\_LPR kann in der Regel unverändert übernommen werden, der Wert für TUSTEP\_UCB ist optional. Welche Bedeutung diese System-Variablen haben, ist im Kapitel »Systemumgebung« ab Seite 71 beschrieben.

In der letzten Zeile der Eingabemaske wird jeweils ein Hinweis gegeben, was in das Feld, in dem der Cursor gerade steht, eingetragen werden soll.

Nach dem Aktivieren der Schaltfläche »Zurück« unten rechts wird wieder die vorherige Eingabemaske angezeigt.

In dieser Eingabemaske kann bei REMOTE-Sitzungen in den Feldern für die System-Variablen TUSTEP\_DSK, TUSTEP\_SCR und TUSTEP\_PRJ statt eines realen Wertes auch »-STD-« angegeben werden. Dies bewirkt, dass der Wert für die jeweilige System-Variable aus dem Inhalt der System-Variablen HOME des Linux-Servers übernommen wird. Die System-Variable HOME enthält unter Linux üblicherweise den Namen für das »home-directory«. Enthält sie z. B. den Wert »/home/userid«, so wird den System-Variablen TUSTEP\_DSK bzw. TUSTEP\_SCR der Wert »/home« und der System-Variablen TUSTEP\_PRJ wird der Wert »userid« zugewiesen.

Weitere Einzelheiten zu dieser Eingabemaske sind im vorhergehenden Kapitel beschrieben.

Nachdem alle erforderlichen Werte in die Eingabemaske eingetragen sind, kann die Sitzung durch Aktivieren der Schaltfläche »Einrichten« eingerichtet werden; d. h. die Angaben in der Eingabemaske werden gespeichert.

Danach kann die Sitzung durch Aktivieren der Schaltfläche »Starten« gestartet werden. Um eine TUSTEP-Sitzung bequemer starten zu können, kann durch Aktivieren der Schaltfläche »Desktop-Icon« ein Icon auf dem Desktop zum Aufruf der Sitzung angelegt werden.

## Linux: TUSTEP auf einem Linux-Server

Im Folgenden wird davon ausgegangen, dass TUSTEP auf dem Linux-Server im Verzeichnis `/opt/tustep` installiert ist.

Um von einem Linux-Rechner aus in einer Remote-Sitzung auf einem Linux-Server mit TUSTEP arbeiten zu können, empfiehlt es sich, dafür mit dem Makro `*DERI` ein Icon auf dem Desktop zu erstellen und damit die Sitzung aufzurufen. Dann können Tastatur und Maus in gleicher Weise wie bei einer normalen (lokalen) TUSTEP-Sitzung unter Linux verwendet werden.

Erstellen eines Icons für Remote-Sitzungen

---

REMOTE-Icon

Titeltext:

Remote-Login:

Erstellen
Ändern
Löschen
Beenden

Bitte den Namen des Icons eintragen

Im ersten Feld der Eingabemaske muss ein Name für das Icon angegeben werden. Er darf aus 1 bis 12 Zeichen (Buchstaben `a` bis `z`, Ziffern, Minuszeichen und `»_«`) bestehen, muss mit einem Buchstaben beginnen und darf nicht mit Minuszeichen oder `»_«` enden.

Im zweiten Feld kann ein Text für die Titelzeile des TUSTEP-Fensters eingetragen werden.

Im dritten Feld muss die Anweisung `ssh` eingetragen werden, mit der die Verbindung zum Linux-Server hergestellt wird. `userid` und `host` müssen durch die für die Sitzung gültigen Werte ersetzt werden.

In der Spalte am rechten Rand werden die schon definierten »Remote-Icons« aufgelistet. Um die Definition eines solchen Icons in der Eingabemaske anzuzeigen, kann der Name dieser Sitzung entweder in dieser Spalte mit der Maus angeklickt oder im ersten Feld eingetragen und mit der Return-Taste bestätigt werden.

In der untersten Zeile der Eingabemaske wird jeweils ein kurzer Hinweis darauf gegeben, was in das Feld, in dem der Cursor gerade steht, eingetragen werden soll.

In der vorletzten Zeile befinden sich Schaltflächen. Eine Schaltfläche kann aktiviert werden, indem sie entweder mit der Maus angeklickt wird oder indem der Cursor mit der Tabulatortaste auf die Schaltfläche positioniert und dann die Return-Taste gedrückt wird.

Nachdem alle erforderlichen Werte in die Eingabemaske eingetragen sind, kann das Icon durch Aktivieren der Schaltfläche »Erstellen« erstellt werden; die Angaben in der Eingabemaske werden zusätzlich gespeichert.

Durch Aktivieren der Schaltfläche »Beenden« wird das Makro beendet.

Wird das so erstellte Icon angeklickt, wird eine Verbindung zum Linux-Server hergestellt. Danach kann die TUSTEP-Sitzung mit der Anweisung

```
/opt/tustep/start name
```

gestartet werden; dabei ist für *name* der Name der TUSTEP-Sitzung anzugeben.

Falls noch keine TUSTEP-Sitzung auf dem Linux-Server definiert ist, kann TUSTEP mit der Anweisung

```
/opt/tustep/start
```

gestartet werden. In TUSTEP wird in diesem Fall automatisch das Makro \*DESI aufgerufen. Nach einer kurzen Anleitung wird eine Eingabemaske angezeigt, mit der eine Sitzung definiert werden kann (siehe »Definieren und Starten einer TUSTEP-Sitzung« ab Seite 80); auf dem Linux-Server kann jedoch kein Icon für die Sitzung angelegt werden. Danach kann die Sitzung wie oben angegeben aufgerufen werden.

Weitere TUSTEP-Sitzungen können dann auch in jeder TUSTEP-Sitzung auf dem Linux-Server mit dem Makro \*DESI definiert werden.

## macOS: TUSTEP auf einem Linux-Server

Im Folgenden wird davon ausgegangen, dass TUSTEP auf dem Linux-Server im Verzeichnis `/opt/tustep` installiert ist.

Um von einem macOS-Rechner aus in einer Remote-Sitzung auf einem Linux-Server mit TUSTEP arbeiten zu können, empfiehlt es sich, dafür mit dem Makro \*DERI ein Icon auf dem Desktop zu erstellen und damit die Sitzung aufzurufen. Dann können Tastatur und Maus in gleicher Weise wie bei einer normalen (lokalen) TUSTEP-Sitzung unter macOS verwendet werden.

Im ersten Feld der Eingabemaske muss ein Name für das Icon angegeben werden. Er darf aus 1 bis 12 Zeichen (Buchstaben a bis z, Ziffern, Minuszeichen und »\_«) bestehen, muss mit einem Buchstaben beginnen und darf nicht mit Minuszeichen oder »\_« enden.

Im zweiten Feld kann ein Kommentar eingetragen werden.

Im dritten Feld muss die Anweisung `ssh` eingetragen werden, mit der die Verbindung zum Linux-Server hergestellt wird. `userid` und `host` müssen durch reale Werte ersetzt werden.

Erstellen eines Icons für Remote-Sitzungen

REMOTE-Icon

Kommentar:

Remote-Login:

TUSTEP-Start:

Erstellen
Ändern
Löschen
Beenden

Bitte den Namen des Icons eintragen

Das vierte Feld kann/sollte leer gelassen werden (s. u.).

In der Spalte am rechten Rand werden die schon definierten »Remote-Icons« aufgelistet. Um die Definition eines solchen Icons in der Eingabemaske anzuzeigen, kann der Name dieser Sitzung entweder in dieser Spalte mit der Maus angeklickt oder im ersten Feld eingetragen und mit der Return-Taste bestätigt werden.

In der untersten Zeile der Eingabemaske wird jeweils ein kurzer Hinweis darauf gegeben, was in das Feld, in dem der Cursor gerade steht, eingetragen werden soll.

In der vorletzten Zeile befinden sich Schaltflächen. Eine Schaltfläche kann aktiviert werden, indem sie entweder mit der Maus angeklickt wird oder indem der Cursor mit der Tabulatortaste auf die Schaltfläche positioniert und dann die Return-Taste gedrückt wird.

Nachdem alle erforderlichen Werte in die Eingabemaske eingetragen sind, kann das Icon durch Aktivieren der Schaltfläche »Erstellen« erstellt werden; die Angaben in der Eingabemaske werden zusätzlich gespeichert.

Durch Aktivieren der Schaltfläche »Beenden« wird das Makro beendet.

Wird das so erstellte Icon angeklickt, wird eine Verbindung zum Linux-Server hergestellt. Danach kann die TUSTEP-Sitzung mit der Anweisung

```
/opt/tustep/start name
```

gestartet werden; dabei ist für *name* der Name der TUSTEP-Sitzung anzugeben.

Falls noch keine TUSTEP-Sitzung auf dem Linux-Server definiert ist, kann TUSTEP mit der Anweisung

```
/opt/tustep/start
```

gestartet werden. In TUSTEP wird in diesem Fall automatisch das Makro \*DESI aufgerufen. Nach einer kurzen Anleitung wird eine Eingabemaske angezeigt, mit der

eine Sitzung definiert werden kann (siehe »Definieren und Starten einer TUSTEP-Sitzung« ab Seite 80); auf dem Linux-Server kann jedoch kein Icon für die Sitzung angelegt werden. Danach kann die Sitzung wie oben angegeben aufgerufen werden.

Weitere TUSTEP-Sitzungen können dann auch in jeder TUSTEP-Sitzung auf dem Linux-Server mit dem Makro \*DESI definiert werden.

Falls `ssh` so konfiguriert ist, dass das Einloggen ohne Eingabe des Passworts möglich ist, kann der TUSTEP-Aufruf wie oben angegeben in das vierte Feld eingetragen werden.

Die TUSTEP-Sitzung wird dann nach dem Anklicken des Icons automatisch aufgerufen und nach dem Beenden von TUSTEP wird das Fenster wieder automatisch geschlossen.

Falls `ssh` nicht entsprechend konfiguriert ist, sollte in das vierte Feld nichts eingetragen werden!

## Beginnen einer TUSTEP-Sitzung

Zu Beginn jeder TUSTEP-Sitzung wird TUSTEP initialisiert. Dabei werden permanente Dateien angemeldet und temporäre Dateien eingerichtet, die von TUSTEP für interne Zwecke benötigt werden.

Falls eine INI-Datei (siehe Seite 89) existiert und diese keine Segment-Datei ist, werden die darin enthaltenen Kommandos (andere Daten sind in diesem Fall nicht erlaubt) ausgeführt; falls sie eine Segment-Datei ist und ein Segment mit dem Namen `INIT` enthält, wird der Inhalt dieses Segments als Kommando-Makro interpretiert ausgeführt.

## Unterbrechen einer TUSTEP-Sitzung

Soll mit einer TUSTEP-Sitzung später weitergearbeitet werden, so kann sie durch Drücken der Escape-Taste oder durch Eingabe von \*EOF unterbrochen werden.

Falls eine INI-Datei (siehe Seite 89) existiert, diese eine Segment-Datei ist und ein Segment mit dem Namen `EXIT` enthält, wird der Inhalt dieses Segments als Kommando-Makro interpretiert ausgeführt.

Windows: Zum Unterbrechen einer TUSTEP-Sitzung kann auch die kleine Schaltfläche mit dem X in der rechten oberen Ecke des Fensters angeklickt werden.

Linux: Zum Unterbrechen einer TUSTEP-Sitzung sollte keinesfalls die kleine Schaltfläche mit dem X in der rechten oberen Ecke des Fensters angeklickt werden. Dadurch würde die TUSTEP-Sitzung abgebrochen und das Fenster geschlossen, aber offene Dateien würden dabei nicht korrekt abgeschlossen und Daten gingen möglicherweise verloren.

macOS: Zum Unterbrechen einer TUSTEP-Sitzung sollte keinesfalls der rote Punkt in der linken oberen Ecke des Fensters angeklickt werden. Dadurch würde die TUSTEP-Sitzung abgebrochen und das Fenster geschlossen, aber offene Dateien würden dabei nicht korrekt abgeschlossen und Daten gingen möglicherweise verloren.

## Fortsetzen einer TUSTEP-Sitzung

Eine unterbrochene TUSTEP-Sitzung kann fortgesetzt werden, indem TUSTEP wie zu Beginn der Sitzung gestartet wird.

Falls eine INI-Datei (siehe Seite 89) existiert, diese eine Segment-Datei ist und ein Segment mit dem Namen `CONT` enthält, wird der Inhalt dieses Segments als Kommando-Makro interpretiert ausgeführt.

Falls die Sitzung jedoch nicht wie oben beschrieben unterbrochen, sondern sonst irgendwie abgebrochen wurde (z. B. durch Programmabsturz oder Stromausfall), wird beim Fortsetzen der Sitzung statt des Makros im Segment `CONT`, das Makro im Segment `RECO` ausgeführt, falls die INI-Datei ein Segment mit diesem Namen enthält.

## Beenden einer TUSTEP-Sitzung

Eine TUSTEP-Sitzung wird mit dem Kommando `#BEEENDE` (siehe Seite 127) oder `#NORMIERE` (siehe Seite 205) beendet; dabei werden alle Scratch-Dateien gelöscht.

Falls eine INI-Datei (siehe Seite 89) existiert, diese eine Segment-Datei ist und ein Segment mit dem Namen `TERM` enthält, wird der Inhalt dieses Segments zuvor als Kommando-Makro interpretiert und ausgeführt. Dieses Makro darf keine Kommandofolge zusammenstellen, die anschließend noch ausgeführt werden soll; ggf. können Kommandos mit der Makroanweisung `EXECUTE` ausgeführt werden.



## INI-Datei

Soll nach dem Starten von TUSTEP automatisch eine Kommandofolge oder ein Makro ausgeführt werden, muss eine INI-Datei eingerichtet werden:

```
#DATEI, *TUSTEP.INI, RAN-P
```

In die INI-Datei können diejenigen Kommandos eingetragen werden, die bei Beginn einer TUSTEP-Sitzung automatisch ausgeführt werden sollen. Sie darf (wenn es keine Segment-Datei ist) außer diesen Kommandos keine anderen Daten enthalten.

Wenn die INI-Datei eine Segment-Datei ist, gilt folgendes:

- Enthält die Datei ein Segment mit dem Namen `INIT`, wird es bei Beginn einer Sitzung als Makro ausgeführt;
- enthält die Datei ein Segment mit dem Namen `EXIT`, wird es beim Unterbrechen einer Sitzung als Makro ausgeführt;
- enthält die Datei ein Segment mit dem Namen `CONT`, wird es beim Fortsetzen einer regulär unterbrochenen Sitzung (siehe »Unterbrechen einer TUSTEP-Sitzung« Seite 87) als Makro ausgeführt;
- enthält die Datei ein Segment mit dem Namen `RECO`, wird es beim Fortsetzen einer abgebrochenen Sitzung (siehe »Fortsetzen einer TUSTEP-Sitzung« Seite 88) als Makro ausgeführt;
- enthält die Datei ein Segment mit dem Namen `TERM`, wird es beim Beenden einer Sitzung als Makro ausgeführt;
- enthält die Datei ein Segment mit dem Namen `EDIT`, darf es nur Editor-Definitionen enthalten; sie werden beim ersten Aufruf des Editors in einer Sitzung automatisch interpretiert; außerdem werden sie interpretiert, wenn im Aufruf des Editors die Spezifikation `DEFINITIONEN=+` angegeben wird.

Die INI-Datei ist eine TUSTEP-Datei mit dem vorgegebenen Namen `TUSTEP.INI`; sie muss im Verzeichnis stehen, dessen Namen sich aus den System-Variablen `TUSTEP_DSK` und `TUSTEP_PRJ` ergibt.

Soll als INI-Datei eine ASCII-Datei oder eine Datei mit einem anderen Namen verwendet werden, so muss der Name dieser Datei mit der System-Variablen `TUSTEP_INI` vorgegeben werden. Diese Datei wird automatisch in eine interne TUSTEP-Datei mit dem Namen `*TUSTEP.INI` kopiert. Beginnt die Datei mit einer Zeile der Form `»#=: name«`, so wird eine Segment-Datei erstellt, wobei jede Zeile der Form `»#=: name«` als Beginn eines neuen Segments mit dem angegebenen Namen interpretiert wird. Achtung: Diese interne TUSTEP-Datei ist eine SCRATCH-Datei; werden in ihr Änderungen vorgenommen, werden diese nicht automatisch in die Datei übertragen, die mit der System-Variablen `TUSTEP_INI` vorgegeben ist.

## Starten von TUSTEP als System-Kommando

TUSTEP kann auch als Programm für ein System-Kommando verwendet werden, bei der keine Interaktion mit dem Benutzer stattfindet. Dabei wird automatisch ein CMD-Makro aus der INI-Datei (siehe Seite 89) ausgeführt und danach TUSTEP wieder beendet.

Welches Makro ausgeführt wird und wo sich die INI-Datei befindet, wird durch System-Variablen festgelegt. Die System-Variablen werden mit einem »Script« definiert. Das Script muss in einer ASCII-Datei vorliegen.

Zum Einrichten einer Datei mit einem solchen Script kann das Makro \*CMD verwendet werden. Nach dem Aufruf dieses Makros wird folgende Eingabemaske angezeigt:

Einrichten eines Scripts für ein System-Kommando

Pfad zum Script	<input type="text"/>	Auswahl	
Name des Scripts	<input type="text"/>		
Pfad zur LOG-Datei	<input type="text"/>	Auswahl	
Pfad TUSTEP_DSK	<input type="text"/>	Auswahl	
Pfad TUSTEP_SCR	<input type="text"/>	Auswahl	
Name TUSTEP_PRJ	<input type="text"/>	Nummer TUSTEP_MEM	<input type="text"/>
Name der INI-Datei	<input type="text"/>		
<input type="button" value="Einrichten"/>		<input type="button" value="Beenden"/>	

Im ersten Feld der Eingabemaske muss das Verzeichnis angegeben werden, in dem das Script eingerichtet wird.

Im zweiten Feld muss der Name des Scripts angegeben werden. er ist zugleich der Name des System-Kommandos und der Name des CMD-Makros in der INI-Datei, das ausgeführt werden soll. Da die Makros mit den Namen INIT, CONT, RECO, EXIT, TERM in der INI-Datei Sonderfunktionen haben, können diese Namen nicht als Scriptname verwendet werden.

In den folgenden Feldern werden die System-Variablen für die jeweilige Sitzung definiert. Welche Bedeutung die vorgegebenen Variablen haben, ist im Kapitel »Systemumgebung« ab Seite 71 beschrieben.

Nach Aktivieren einer Schaltfläche »Auswahl« wird eine Liste mit Namen von System-Variablen und den jeweiligen Inhalten angezeigt. Die Liste enthält jedoch nur die System-Variablen, deren Inhalt den vollständigen Pfad eines Verzeichnisses angibt, auf das zugegriffen werden kann. Wird eine System-Variable ausgewählt, so wird deren Inhalt in das jeweils vor der Schaltfläche stehende Eingabefeld eingetragen. In diesem Feld kann der Pfad ggf. noch geändert und ergänzt werden.

Als INI-Datei muss eine Datei verwendet werden, die nur für CMD-Makros verwendet wird, da die Segmente mit dem Namen CONT und RECO zweckmäßigerweise das Kommando #NORMIERE enthalten (siehe unten), das in allen anderen Fällen in der Regel nicht sinnvoll ist. Soll als INI-Datei die Datei »\*TUSTEP . INI« verwendet werden, so kann das Feld leer gelassen werden.

Durch Aktivieren der Schaltfläche »Einrichten« wird das Script eingerichtet; durch Anklicken der Schaltfläche »Beenden« wird das Makro \*CMD beendet.

Das Einrichten des Scripts beinhaltet folgendes:

- Einrichten der ASCII-Datei, die das Script enthält.

Unter Linux und macOS ist der Name der Datei identisch mit dem Namen des Scripts. Die Datei hat etwa folgenden Inhalt:

```

#!/bin/bash
#=                               Script mit bash abarbeiten
export TUSTEP_MDS=BATCH
#=                               TUSTEP muss im Batch-Modus laufen
export TUSTEP_LIB=/opt
#=                               Pfad zum TUSTEP-Directory tustep
export TUSTEP_DSK=~/Projekte/batch
#=                               Pfad zur INI-Datei
export TUSTEP_SCR=~/Projekte/batch
#=                               Pfad zu den Scratch-Dateien
export TUSTEP_PRJ=-
#=                               Subdir. für TUSTEP_DSK + TUSTEP_SCR
export TUSTEP_MEM=00100000
#=                               Eindeutige Sitzungsnummer
export TUSTEP_INI=
#=                               nichts oder Name der INI-Datei
export TUSTEP_CMD=scriptname
#=                               Name des Makros in der INI-Datei
export TUSTEP_1=$1
export TUSTEP_2=$2
export TUSTEP_3=$3
export TUSTEP_4=$4
export TUSTEP_5=$5
export TUSTEP_6=$6
export TUSTEP_7=$7
export TUSTEP_8=$8
export TUSTEP_9=$9
#=                               Aufruf-Argumente für Makro merken
$TUSTEP_LIB/tustep/command > ~/Projekte/batch/scriptname.log
#=                               TUSTEP-Makro ausführen
#=                               Ablaufprotokoll --> scriptname.log

```

Unter Windows entspricht der Name der Datei dem Namen des Scripts mit der Namenserweiterung (Extension) »bat«. Die Datei hat etwa folgenden Inhalt:

```
@ECHO OFF
REM                               Anweisungen nicht protokollieren
SET TUSTEP_MDS=BATCH
REM                               TUSTEP muss im Batch-Modus laufen
SET TUSTEP_LIB=C:\PROGRAMME
REM                               Pfad zum TUSTEP-Directory tustep
SET TUSTEP_DSK=~\Projekte\batch
REM                               Pfad zur INI-Datei
SET TUSTEP_SCR=~\Projekte\batch
REM                               Pfad zu den Scratch-Dateien
SET TUSTEP_PRJ=-
REM                               Subdir. für TUSTEP_DSK+TUSTEP_SCR
SET TUSTEP_MEM=00100000
REM                               Eindeutige Sitzungsnummer
SET TUSTEP_INI=
REM                               nichts oder Name der INI-Datei
SET TUSTEP_CMD=scriptname
REM                               Name des CMD-Makros in der INI-Datei
SET TUSTEP_1=%1
SET TUSTEP_2=%2
SET TUSTEP_3=%3
SET TUSTEP_4=%4
SET TUSTEP_5=%5
SET TUSTEP_6=%6
SET TUSTEP_7=%7
SET TUSTEP_8=%8
SET TUSTEP_9=%9
REM                               Aufruf-Argumente für Makro merken
%TUSTEP_LIB%\tustep\command > ~\Projekte\batch\scriptname.log
REM                               TUSTEP-Makro ausführen
REM                               Ablaufprotokoll --> scriptname.log
```

- Einrichten der INI-Datei, falls sie im angegebenen Verzeichnis noch nicht existiert.
- Einrichten eines Segments mit dem Namen INIT in der INI-Datei, falls noch kein Segment mit diesem Namen existiert. Das Segment enthält die Makroanweisungen

```
$$ EXECUTE/QUIET #definiere, code=ISO
```

```
$$ EXECUTE/QUIET #fehlerhalt, loesche
```

und sorgt dafür, dass beim Initialisieren der TUSTEP-Sitzungen der angegebene Code für das Ablaufprotokoll eingestellt wird, und nachfolgende Kommandos nicht mehr ausgeführt werden, falls das Makro Kommandos erzeugt und bei der Ausführung dieser Kommandos ein Fehler auftritt.

- Einrichten von Segmenten mit dem Namen CONT und RECO in der INI-Datei, falls

noch kein Segment mit dem jeweiligen Namen existiert. Die Segmente enthalten jeweils das Kommando

```
#NORMIERE
```

und sorgen dafür, dass die TUSTEP-Sitzung beendet und neu initialisiert wird, falls sie beim vorangehenden Aufruf aus irgend einem Grund nicht beendet wurde.

- Einrichten eines Segments mit dem Namen des Scripts in der INI-Datei, falls noch kein Segment mit diesem Namen existiert. Das Segment enthält ein aufrufbares CMD-Makro. Es entspricht dem unten aufgeführten Beispiel 1 und dient nur als Stellvertreter; es muss noch angepasst und ergänzt oder durch ein eigenes Makro ersetzt werden.

Nachdem das Script eingerichtet ist, kann es auf Betriebssystemebene aufgerufen werden. Dadurch wird eine neue TUSTEP-Sitzung begonnen, das Makro ausgeführt und die TUSTEP-Sitzung wieder beendet. Der Aufruf kann z. B. wie folgt aussehen:

```
scriptname quelldatei zieldatei
```

Das Ablaufprotokoll der TUSTEP-Sitzung wird in eine Datei abgelegt; sie steht in dem für die LOG-Datei angegebenen Verzeichnis und hat den gleichen Namen wie das Script, jedoch mit der Namensweiterung (Extension) »log«.

In den Script-Dateien können bei Bedarf zusätzliche System-Variablen definiert und Werte für System-Variablen geändert werden; dabei ist zu beachten, dass die zu einem Script gehörende INI-Datei immer in dem Verzeichnis stehen muss, das mit den System-Variablen TUSTEP\_DSK und TUSTEP\_PRJ vorgegeben wird.

Nachfolgend sind zwei Beispiele für ein CMD-Makro angegeben. Das erste Beispiel kopiert den Inhalt einer Fremd-Datei, deren Daten in UTF-8 (!) codiert sind, in eine andere Fremd-Datei. Quell- und Ziel-Datei müssen im Aufruf angegeben werden. Um Daten auszuwählen oder zu verändern, müssen an der angegebenen Stelle im Makro noch Makroanweisungen eingefügt werden. Beim zweiten Beispiel werden die Daten nicht über Makroanweisungen, sondern mit dem Kommando #KOPIERE ausgewählt/verändert. Die erforderlichen Parameter werden in einer Datei erwartet, die beim Aufruf als dritte anzugeben ist.

**Beispiel 1**

```
$$ MODE TUSCRIPT, {}
```

- Dateinamen des System-Kommandos übernehmen

```
FETCH argument_1 = TUSTEP_1
```

```
FETCH argument_2 = TUSTEP_2
```

- Dateinamen mit Pfad versehen (System-konforme Dateinamen)

```
SET quelle = FULL_NAME (SYSTEM, argument_1)
```

```
SET ziel = FULL_NAME (SYSTEM, argument_2)
```

- Dateinamen für TUSTEP definieren (TUSTEP-konforme Dateinamen)

```
DEFINE "quelle" = quelle
```

```
DEFINE "ziel" = ziel
```

- Kontrollausdruck der Aufrufzeit und der Dateinamen

```
SET datum = DATE (3), uhrzeit = TIME (2)
```

```
PRINT " Ausführung: ", datum, " ", uhrzeit
```

```
PRINT " Argument 1: Quelldatei -*QUELLE = ", argument_1
```

```
PRINT " Argument 2: Zieldatei -*ZIEL = ", argument_2
```

- Quell-Datei anmelden, Ziel-Datei einrichten/anmelden

```
ERROR/STOP OPEN ("quelle", READ, -)
```

```
ERROR/STOP CREATE ("ziel", FDF-O, -)
```

- Dateibearbeitung beginnen (UTF8 ggf. durch ISO ersetzen!)

```
ACCESS q: READ /RECORDS/UTF8 "quelle" num, text
```

```
ACCESS z: WRITE/ERASE/RECORDS/UTF8 "ziel" num, text
```

- Daten lesen, bearbeiten, schreiben

```
COMPILE
```

```
LOOP/999999
```

```
READ/NEXT/EXIT q
```

- Hier Anweisungen zum Auswählen/Ändern der Daten einfügen

```
WRITE/NEXT z
```

```
ENDLOOP
```

```
ENDCOMPILE
```

- Dateibearbeitung beenden

```
ENDACCESS/PRINT q  
ENDACCESS/PRINT z
```

- Dateien abmelden

```
ERROR/STOP CLOSE ("quelle")  
ERROR/STOP CLOSE ("ziel")
```

**Beispiel 2**

```
$$ MODE TUSCRIPT, {}
```

```
- Dateinamen des System-Kommandos übernehmen
```

```
FETCH argument_1 = TUSTEP_1
```

```
FETCH argument_2 = TUSTEP_2
```

```
FETCH argument_3 = TUSTEP_3
```

```
- Dateinamen mit Pfad versehen (System-konforme Dateinamen)
```

```
SET quelle      = FULL_NAME (SYSTEM, argument_1)
```

```
SET ziel        = FULL_NAME (SYSTEM, argument_2)
```

```
SET parameter   = FULL_NAME (SYSTEM, argument_3)
```

```
- Dateinamen für TUSTEP definieren (TUSTEP-konforme Dateinamen)
```

```
DEFINE "sys*quelle"    = quelle
```

```
DEFINE "sys*ziel"      = ziel
```

```
DEFINE "sys*parameter" = parameter
```

```
- Kontrollausdruck der Aufrufzeit und der Dateinamen
```

```
SET datum = DATE (3), uhrzeit = TIME (2)
```

```
PRINT "Ausführung: ", datum, " ", uhrzeit
```

```
PRINT "Argument 1: Quelldatei      SYS*QUELLE      = ", argument_1
```

```
PRINT "Argument 2: Zieldatei      SYS*ZIEL        = ", argument_2
```

```
PRINT "Argument 3: Parameterdatei SYS*PARAMETER = ", argument_3
```

```
MODE DATA
```

```
#- Quell-Datei anmelden, Ziel-Datei einrichten/anmelden
```

```
#anmelde, sys*quelle'sys*parameter, traeger==
```

```
#datei, sys*ziel, fdf-ap, traeger==
```

```
#- Hilfsdateien einrichten
```

```
#datei, scr*quelle'scr*ziel'scr*parameter
```

```
#- Dateien von UTF-8 nach TUSTEP umwandeln
```

```
#um, sys*quelle, scr*quelle, , +, code=UTF8, opt=zirkumflex
```

```
#um, sys*parameter, scr*parameter,, +, code=UTF8, opt=zirkumflex
```

```
#- Daten entsprechend der Parameter auswählen/verändern
```



```
#kopiere, scr*quelle, scr*ziel, , +, scr*parameter
```

```
- Ergebnis von TUSTEP nach UTF-8 umwandeln
```

```
#umwandle, scr*ziel, sys*ziel, , +, code=UTF8, option=zirkumflex
```

```
#- Dateien abmelden, Hilfsdateien löschen
```

```
#abmelde, sys*quelle'sys*ziel'sys*parameter
```

```
#loesche, scr*quelle'scr*ziel'scr*parameter
```

## Starten von TUSTEP durch einen WWW-Server

Wird TUSTEP durch einen WWW-Server aufgerufen, so wird automatisch ein CGI-Makro aus der INI-Datei (siehe Seite 89) ausgeführt und danach TUSTEP wieder beendet.

Welches Makro ausgeführt wird und wo sich die INI-Datei befindet, wird durch System-Variablen festgelegt. Die System-Variablen werden mit einem »Script« definiert. Das Script muss in einer ASCII-Datei vorliegen.

Zum Einrichten einer Datei mit einem solchen Script kann das Makro \*CGI verwendet werden. Nach dem Aufruf dieses Makros wird folgende Eingabemaske angezeigt:

Einrichten eines Scripts für ein CGI-Makro

---

Pfad zum Script

Name des Scripts

Pfad TUSTEP\_DSK

Pfad TUSTEP\_SCR

Projektname                       Sitzungsnummer

Code für HTML     ISO-8859-1  
                           UTF-8

Im ersten Feld der Eingabemaske muss das Verzeichnis angegeben werden, in dem die Datei mit dem CGI-Script erstellt werden soll. Das Verzeichnis ist nicht frei wählbar. Es darf nur eines angegeben werden, das vom WWW-Server dafür vorgesehen ist.

Beispiel für das Verzeichnis unter Linux:

```
/var/www/cgi-bin
```

Beispiel für das Verzeichnis unter macOS:

```
/Library/WebServer/CGI-Executables
```

Beispiel für das Verzeichnis unter Windows:

```
C:\Programme\Apache\cgi-bin
```

Falls unter Windows der TUSTEP-Server verwendet wird, ist das Verzeichnis frei wählbar und kann z. B. folgenden Namen haben:

```
C:\server
```

Der Name des Scripts ist zugleich der Name des CGI-Makros in der INI-Datei, das

ausgeführt werden soll. Da die Makros mit den Namen INIT, CONT, RECO, EXIT, TERM in der INI-Datei Sonderfunktionen haben, können diese Namen nicht als Scriptname verwendet werden.

Die Sitzungsnummer für die TUSTEP-Sitzung muss so gewählt werden, dass keine Konflikte mit anderen TUSTEP-Sitzungen auftreten können.

Werden von dem angegebenen Makro als Standard-Ausgabe Daten erzeugt, für deren Anzeige nicht-lateinische Schriften (z. B. Griechisch) erforderlich sind, muss als Code UTF-8 gewählt werden, andernfalls genügt ISO-8859-1.

Durch Aktivieren der Schaltfläche »Einrichten« wird das Script eingerichtet; durch Anklicken der Schaltfläche »Beenden« wird das Makro beendet.

Das Einrichten des Scripts beinhaltet folgendes:

- Einrichten der ASCII-Datei, die das Script enthält.

Unter Linux ist der Name der Datei identisch mit dem Namen des Scripts. Die Datei hat etwa folgenden Inhalt:

```
#!/bin/bash
#=                               Script mit bash abarbeiten
export HOME=/var/www/cgi-bin/test
#=                               Ersatz-Pfad für Tilde am Pfadanzfang
export TUSTEP_MDS=BATCH
#=                               TUSTEP muss im Batch-Modus laufen
export TUSTEP_LIB=/opt
#=                               Pfad zum TUSTEP-Directory tustep
export TUSTEP_DSK=/var/www/cgi-bin/test
#=                               Pfad zur INI-Datei und den Daten
export TUSTEP_SCR=/var/www/cgi-bin/test/tmp
#=                               Pfad zu den Scratch-Dateien
export TUSTEP_PRJ=-
#=                               Subdir. für TUSTEP_DSK+TUSTEP_SCR
export TUSTEP_MEM=00100000
#=                               Eindeutige Sitzungsnummer
export TUSTEP_CGI=scriptname/Code
#=                               Name des CGI-Makros in der INI-Datei
#=                               Code: ISO-8859-1 oder UTF-8
$TUSTEP_LIB/tustep/cgibin
#=                               TUSTEP-Makro ausführen
```

Unter Windows entspricht der Name der Datei dem Namen des Scripts mit der Namenserweiterung (Extension) »tsd«. Die Datei hat etwa folgenden Inhalt:

```
CGIBIN-Session: scriptname
=
                        Name dieser TUSTEP-Start-Datei
HOME=C:\SERVER\TEST
=
                        Ersatz-Pfad für Tilde am Pfad Anfang
TUSTEP_MDS=BATCH
=
                        TUSTEP muss im Batch-Modus laufen
TUSTEP_LIB=C:\PROGRAMME
=
                        Pfad zum TUSTEP-Directory tustep
TUSTEP_DSK=C:\SERVER\TEST
=
                        Pfad zur INI-Datei und den Daten
TUSTEP_SCR=C:\SERVER\TEST\TMP
=
                        Pfad zu den Scratch-Dateien
TUSTEP_PRJ=-
=
                        Subdir. für TUSTEP_DSK+TUSTEP_SCR
TUSTEP_MEM=00100000
=
                        Eindeutige Sitzungsnummer
TUSTEP_CGI=scriptname/code
=
                        Name des CGI-Makros in der INI-Datei
=
                        Code: ISO-8859-1 oder UTF-8
```

- Unter Windows: Einrichten eines Programms, das das Script interpretiert und TUSTEP startet. Der Name des Programms entspricht dem Namen des Scripts mit der Namenserweiterung (Extension) »exe«.

Anmerkung: Unter Linux und macOS übernimmt die Aufgabe dieses Programms der Befehlsinterpreter des Betriebssystems (die Shell).

- Einrichten der INI-Datei, falls sie in dem für das Script angegebenen Verzeichnis noch nicht existiert.
- Einrichten eines Segments mit dem Namen des Scripts in der INI-Datei, falls noch kein Segment mit diesem Namen existiert. Das Segment enthält ein aufrufbares Makro. Es dient nur als Stellvertreter und muss noch durch das eigene Makro ersetzt werden. Einfache Beispiele für CGI-Makros sind ab Seite 659 aufgeführt.

Nachdem das Script eingerichtet ist, kann es mit einem Browser aufgerufen werden. Dadurch wird TUSTEP gestartet und das gleichnamige Makro ausgeführt. Der Aufruf kann z. B. wie folgt aussehen:

```
http://servername/cgi-bin/scriptname
```

Falls die TUSTEP-Sitzung, die durch die im Script definierten System-Variablen vorgegeben ist, noch nicht existiert, wird sie initialisiert. Sie bleibt dann so lange erhalten, bis sie explizit beendet wird bzw. die zur Sitzung gehörenden Scratch-Dateien (außerhalb von TUSTEP) gelöscht werden. In der Regel wird die Sitzung bei einem Aufruf von TUSTEP fortgesetzt und nach der Ausführung des Makros unterbrochen.

In den Script-Dateien können bei Bedarf zusätzliche System-Variablen definiert und Werte für System-Variablen geändert werden; dabei ist zu beachten, dass die zu einem

Script gehörende INI-Datei immer in dem Verzeichnis stehen muss, das mit den System-Variablen TUSTEP\_DSK und TUSTEP\_PRJ vorgegeben wird.

Wenn die TUSTEP-Variable TUSTEP\_CGI »segmentname/code« enthält, so wird von TUSTEP automatisch als erstes die Zeile

```
Content-type: text/html; charset=ISO-8859-1
```

bzw.

```
Content-type: text/html; charset=UTF-8
```

und eine Leerzeile ausgegeben.

In Sonderfällen kann statt der Code-Angabe ein Minuszeichen angegeben werden. In diesem Fall wird die unbedingt notwendige Content-type-Information nicht von TUSTEP automatisch ausgegeben, sondern muss vom CGI-Makro ausgegeben werden. Dabei ist darauf zu achten, dass diese Information mit einer Leerzeile abgeschlossen werden muss.

Unter Windows enthält TUSTEP einen WWW-Server. Er kann zum Testen von CGI-Makros verwendet werden. Darüber hinaus bietet er die Möglichkeit, eine TUSTEP-Anwendung zu erstellen, bei der die Anzeige nicht in einem TUSTEP-Fensters, sondern in einem Browser-Fenster (z. B. dem Internet-Browser) erfolgt.

Zum Starten und Beenden des Servers kann das Makro \*SERVER verwendet werden. Nach dem Aufruf dieses Makros wird folgende Eingabemaske angezeigt:

WWW-Server Maske schließen

Port  Server stoppen

Zugriff  \* nur local  o auch remote

Pfad

Script

URL  Server starten

Icon

Info  Icon erstellen

---

Beenden des Server-Makros

Port 80 ist voreingestellt. Die Verwendung eines anderen Ports ist nur in Spezialfällen sinnvoll.

Um den Server zu stoppen ist keine weitere Angabe erforderlich; es genügt das Aktivieren der entsprechenden Schaltfläche.

Ob der Server nur Zugriffe vom selben Rechner (`local`) oder auch von anderen Rechnern über das Internet (`remote`) bedient, kann durch Anklicken der entsprechenden Felder bestimmt werden.

Um den Server zu starten muss mindestens der Pfad für das »home-directory« des Servers angegeben werden. Das Verzeichnis ist frei wählbar und kann z. B. folgenden Namen haben:

```
C:\Server
```

Hinweis: Beim Einrichten eines Scripts mit dem Makro \*CGI (siehe Seite 98) muss dieses Verzeichnis oder ein Verzeichnis, das diesem untergeordnet ist, angegeben werden.

Der Aufruf eines Scripts mit dem Namen `test` in diesem Verzeichnis hätte vom selben Rechner aus dann folgende Form:

```
http://localhost/test
```

Durch den Aufruf wird TUSTEP gestartet, das im Script angegebene CGI-Makro ausgeführt und TUSTEP wieder beendet. Dadurch ergibt sich, dass die Makrovariablen bei einem erneuten Aufruf wieder neu definiert werden müssen; ein Zugriff auf die Inhalte der Variablen, die sie beim letzten Aufruf zuletzt hatten, ist deshalb nicht möglich.

Wenn für eine Anwendung immer das gleiche CGI-Makro ausgeführt wird, kann in der Eingabemaske der Name des Scripts angegeben werden; falls das Script nicht im »home-directory« des Servers steht, muss der Pfad mit angegeben werden. Die Angabe bewirkt, dass das Makro beim Starten des Servers in den Server integriert wird; die Inhalte der Makrovariablen bleiben in diesem Fall von Aufruf zu Aufruf erhalten. Soll mehr als eine solche Anwendung gleichzeitig ausgeführt werden, muss für jede Anwendung ein Server gestartet werden und dabei jeweils ein anderer Port verwendet werden.

Um beim Starten des Servers automatisch auch den Browser zu starten, kann in der Eingabemaske die Adresse der gewünschten »Start-Seite« unter URL angegeben werden. In diesem Fall wird der Server auch wieder automatisch gestoppt, sobald das Browser-Fenster geschlossen wird. Außerdem ist es in diesem Fall möglich, mit der Makroanweisung `TERMINATE` (siehe Seite 460) die TUSTEP-Sitzung zu beenden, das Browser-Fenster zu schließen und den Server zu stoppen.

Damit die Eingabemaske nicht bei jedem Starten des Servers neu ausgefüllt werden muss, kann auf dem Desktop ein Icon angelegt werden, das sämtliche Angaben enthält. Obligat ist aber nur die Angabe zu Pfad; zu Info kann ein beliebiger Text angegeben werden, der angezeigt wird, wenn der Mauszeiger auf das Icon geführt wird. Nach dem Erstellen des Icons wird der Server entsprechend den Angaben automatisch gestartet. Das Icon hat eine Doppelfunktion: Durch Anklicken des Icons wird der Server gestartet, falls er nicht aktiv ist, und gestoppt, falls er gerade aktiv ist.

Der Server kann auch ohne Eingabemaske gestartet und gestoppt werden. Hierzu kann das Makro \*SERVER mit Spezifikationsangaben aufgerufen werden; weitere Informationen darüber werden mit dem Kommando `#INFORMIERE, *SERVER` ausgegeben.

## Starten des TUSTEP-Editors unter Windows

Innerhalb von TUSTEP-Sitzungen kann der Editor mit dem Kommando #EDIERE (siehe Seite 149) oder #\*E (siehe Seite 257) gestartet werden.

Außerhalb von TUSTEP-Sitzungen kann der TUSTEP-Editor über den Dateimanager (Windows-Explorer) gestartet werden. Dabei wird jeweils automatisch eine neue TUSTEP-Sitzung begonnen, der TUSTEP-Editor für die ausgewählte Datei gestartet und nach dem Beenden des Editors die TUSTEP-Sitzung wieder automatisch beendet.

Bei Dateien, deren Namen die Namenserweiterung (Extension) `.tf` oder `.tstp` haben, genügt es, den Dateinamen mit der linken Maustaste anzuklicken, um den Editor zu starten.

Für Dateien, die eine andere oder keine Namenserweiterung haben, gibt es zwei Möglichkeiten:

- Mit der rechten Maustaste auf den Dateinamen klicken und »Öffnen mit dem TUSTEP-Editor« auswählen. Falls zuvor mehrere Dateien markiert wurden, wird sofort für jede Datei eine eigene Sitzung begonnen und die jeweilige Datei angezeigt.
- Mit der rechten Maustaste auf den Dateinamen klicken, »Senden an« auswählen, dann »TUSTEP-Editor« auswählen. Falls zuvor mehrere Dateien markiert wurden, wird trotzdem nur eine Sitzung begonnen und die erste Datei angezeigt; nach dem Beenden des Editors wird jeweils die nächste Datei angezeigt.

Hinweis: Falls nach der Auswahl von »Senden an« der Eintrag »TUSTEP-Editor« nicht im Menü enthalten ist, kann dieser Eintrag mit dem Makro `*SEND2TUSTEP` ergänzt werden. Falls nach dem Installieren einer neueren TUSTEP-Version der Aufruf des Editors über diesen Eintrag nicht mehr möglich ist, muss der Eintrag mit dem selben Makro aktualisiert werden.

Wenn die betreffende Datei keine TUSTEP-Datei ist, werden die Daten umgewandelt, in eine interne TUSTEP-Datei geschrieben und der TUSTEP-Editor mit dieser internen Datei gestartet. Falls die Daten geändert werden, wird nach dem Beenden des TUSTEP-Editors gefragt, ob die Daten in die ursprüngliche Datei zurückgeschrieben werden sollen.

Wenn eine Datei binäre Daten (im Gegensatz zu Text-Daten) enthält, ist das Anzeigen der Daten mit dem TUSTEP-Editor in der Regel nicht sinnvoll; Änderungen sollten in solchen Dateien mit dem TUSTEP-Editor grundsätzlich nicht vorgenommen werden.

Sollen beim Aufruf des Editors über den Dateimanager persönliche Definitionen (z. B. für die Colorierung) übernommen werden, kann mit dem Makro `*EDIDEF` eine Datei oder ein Segment einer Datei mit diesen Definitionen vorgegeben werden.

Im ersten Feld muss der Pfad mit Dateiname der Datei mit den Editor-Definitionen angegeben werden. Falls diese Datei eine Segmentdatei ist, muss im zweiten Feld der Name des Segments angegeben werden, das die Editor-Definitionen enthält; andernfalls muss dieses Feld leer sein.

In den Feldern »Zeilen« und »Spalten« kann die Größe des Editorfensters eingetragen

Editor-Definitionen für automatischen Editor-Start

---

Pfad der Datei

Name des segments     zeilen     spalten

Parameterart

werden. Unter Windows und macOS können statt einer Zahl zwei Pluszeichen eingetragen werden; in diesem Fall wird das Editorfenster auf die maximal mögliche Höhe bzw. Breite eingestellt.

Durch Aktivieren der Schaltfläche »Speichern« werden die Angaben gespeichert; mit der Schaltfläche »Löschen« können gespeicherte Werte wieder gelöscht werden; durch Anklicken der Schaltfläche »Beenden« wird das Makro \*EDIDEF beendet.

### Starten des TUSTEP-Editors unter Linux

Innerhalb von TUSTEP-Sitzungen kann der Editor mit dem Kommando #EDIERE (siehe Seite 149) oder #\*E (siehe Seite 257) gestartet werden.

Außerhalb von TUSTEP-Sitzungen kann der TUSTEP-Editor über den Dateimanager gestartet werden. Dabei wird jeweils automatisch eine neue TUSTEP-Sitzung begonnen, der TUSTEP-Editor für die ausgewählte Datei gestartet und nach dem Beenden des Editors die TUSTEP-Sitzung wieder automatisch beendet.

Bei TUSTEP-Dateien und bei Dateien, deren Namen die Namensweiterung (Extension) `tf` oder `tstp` haben, genügt es, den Dateinamen mit der linken Maustaste anzuklicken, um den Editor zu starten.

In jedem Fall kann der Dateiname mit der rechten Maustaste angeklickt werden. Im Idealfall gibt es den Menü-Punkt »Mit TUSTEP öffnen«. Andernfalls muss der Menü-Punkt »Öffnen mit ...« bzw. »Mit anderer Anwendung öffnen« ausgewählt werden. Die weiteren Schritte, um TUSTEP in den Menüs zu finden, sind von der jeweiligen Linux-Version bzw. vom Dateimanager abhängig. Meist ist TUSTEP unter »Büroprogramme«, »Dienstprogramme« oder »weitere Programme« zu finden.

Wenn die betreffende Datei keine TUSTEP-Datei ist, werden die Daten umgewandelt, in eine interne TUSTEP-Datei geschrieben und der TUSTEP-Editor mit dieser internen Datei gestartet. Falls die Daten geändert werden, wird nach dem Beenden des TUSTEP-Editors gefragt, ob die Daten in die ursprüngliche Datei zurückgeschrieben werden sollen.

Wenn eine Datei binäre Daten (im Gegensatz zu Text-Daten) enthält, ist das Anzeigen der Daten mit dem TUSTEP-Editor in der Regel nicht sinnvoll; Änderungen



sollten in solchen Dateien mit dem TUSTEP-Editor grundsätzlich nicht vorgenommen werden.

Sollen beim Aufruf des Editors über den Dateimanager persönliche Definitionen (z. B. für die Colorierung) übernommen werden, kann unter Linux in gleicher Weise wie unter Windows mit dem Makro `*EDIDEF` eine Datei oder ein Segment einer Datei mit diesen Definitionen vorgegeben werden (siehe Seite 103).

## Starten des TUSTEP-Editors unter macOS

Innerhalb von TUSTEP-Sitzungen kann der Editor mit dem Kommando `#EDIERE` (siehe Seite 149) oder `#*E` (siehe Seite 257) gestartet werden.

Außerhalb von TUSTEP-Sitzungen kann der TUSTEP-Editor über den Dateimanager (Finder) gestartet werden. Dabei wird jeweils automatisch eine neue TUSTEP-Sitzung begonnen, der TUSTEP-Editor für die ausgewählte Datei gestartet und nach dem Beenden des Editors die TUSTEP-Sitzung wieder automatisch beendet.

Bei Dateien, deren Namen die Namenserweiterung (Extension) `tf` oder `tstp` haben, genügt es, den Dateinamen mit der linken Maustaste anzuklicken, um den Editor zu starten.

Für Dateien, die eine andere oder keine Namenserweiterung haben, gibt es zwei Möglichkeiten:

- Dateinamen mit der linken Maustaste anklicken und Maustaste gedrückt halten.
- Dateinamen ins Dock auf das TUSTEP-Icon ziehen.
- Maustaste wieder loslassen.

Oder

- Dateinamen mit der rechten Maustaste anklicken.
- Menü-Punkt »Öffnen mit« auswählen.
- »TUSTEP.app« anklicken, falls dies angezeigt wird; andernfalls:
- Menü-Punkt »Anderem Programm ...« auswählen.
- Zu »TUSTEP.app« scrollen und anklicken.

Wenn die betreffende Datei keine TUSTEP-Datei ist, werden die Daten umgewandelt, in eine interne TUSTEP-Datei geschrieben und der TUSTEP-Editor mit dieser internen Datei gestartet. Falls die Daten geändert werden, wird nach dem Beenden des TUSTEP-Editors gefragt, ob die Daten in die ursprüngliche Datei zurückgeschrieben werden sollen.

Wenn eine Datei binäre Daten (im Gegensatz zu Text-Daten) enthält, ist das Anzeigen der Daten mit dem TUSTEP-Editor in der Regel nicht sinnvoll; Änderungen sollten in solchen Dateien mit dem TUSTEP-Editor grundsätzlich nicht vorgenommen werden.

Sollen beim Aufruf des Editors über den Dateimanager persönliche Definitionen (z. B. für die Colorierung) übernommen werden, kann unter macOS in gleicher Weise wie unter Windows mit dem Makro `*EDIDEF` eine Datei oder ein Segment einer Datei mit diesen Definitionen vorgegeben werden (siehe Seite 103).



# Kommandos

Allgemeines . . . . .	110
Eingabe . . . . .	113
ABMELDE	Abmelden von Dateien . . . . . 119
AENDERE	Ändern von Dateinamen . . . . . 122
ANMELDE	Anmelden von Dateien . . . . . 124
BEENDE	Beenden/Unterbrechen einer TUSTEP-Sitzung . . . . . 127
DATEI	Einrichten von Dateien und Projekten . . . . . 128
DEFINIERE	Definieren einer Makro-Datei, Variablen u. a. . . . . 132
DRUCKE	Druckausgabe . . . . . 142
DUMPE	Dateien analysieren . . . . . 147
DVORBEREITE	Daten zum Drucken vorbereiten . . . . . 148
EDIERE	Edieren von Daten/Dateien . . . . . 149
EINFUEGE	Einfügen von Textteilen . . . . . 156
FAUFBEREITE	Formulare aufbereiten zum Drucken . . . . . 157
FEHLERHALT	Fehlerhalt ein/ausschalten . . . . . 158
FORMATIERE	Formatieren (autom. Umbruch) von Texten . . . . . 160
HALT	Halt vor Ausführung noch anstehender Kommandos . . . . . 161
HILFE	Online-Hilfe . . . . . 162
HOLE	Holen von Daten . . . . . 165
INFORMIERE	Informieren über Makro-Datei, Variablen u. a. . . . . 167
KAUSFUEHRE	Korrekturanweisungen ausführen . . . . . 171
KOPIERE	Kopieren, Auswählen, Modifizieren . . . . . 172
LISTE	Listen von Dateinamen u. a. . . . . 173
LOESCHE	Löschen von Daten/Dateien und Projekten . . . . . 182
MAKRO	Zusammenstellen einer Kommandofolge . . . . . 186
MBAUSGABE	Ausgabe in eine Band-Datei . . . . . 187
MBEINGABE	Eingabe von einer Band-Datei . . . . . 189
MBINFORMIERE	Informieren über eine Band-Datei . . . . . 191
MBKOPIERE	Kopieren einer Band-Datei . . . . . 193
MBLABEL	Labeln einer Band-Datei . . . . . 197
MBTEST	Testen einer Band-Datei . . . . . 199
MISCHE	Mischen von Daten/Dateien . . . . . 201
MODI	Modi merken/zurücksetzen . . . . . 203
NORMIERE	Normieren/Beenden von TUSTEP . . . . . 205
NUMMERIERE	Neu-Nummerieren von Verweisen . . . . . 206
PARAMETER	Parameter-Modi einstellen . . . . . 207
PROTOKOLL	Zweitprotokoll ein/ausschalten u. a. . . . . 209
RAUFBEREITE	Register nach dem Sortieren aufbereiten . . . . . 215
RETTE	Retten von Daten . . . . . 216
RVORBEREITE	Register zum Sortieren vorbereiten . . . . . 220
SATZ	Satzherstellung . . . . . 221
SIGNAL	Ausgeben von Signaltönen . . . . . 222
SORTIERE	Sortieren von Daten/Dateien . . . . . 224
SPRUEFE	Sortierschlüssel prüfen . . . . . 227
SUCHE	Durchsuchen von Texten . . . . . 228
SVORBEREITE	Daten zum Sortieren vorbereiten . . . . . 229
TESTE	Testen/Vergleichen einer Datei . . . . . 230

---

TITEL	Titel einer Datei definieren/übertragen . . . . .	231
TUE	Ausführen einer Kommandofolge . . . . .	233
UMWANDLE	Umwandeln/Verschlüsseln von Daten/Dateien . . . . .	237
VAUFBEREITE	Vergleichsergebnisse aufbereiten . . . . .	247
VERGLEICHE	Vergleichen fortlaufender Texte . . . . .	248
WAHLSCHALTER	Wahlschalter setzen/löschen . . . . .	249
WISCHEN	Wischen (Überschreiben) von Daten . . . . .	250
ZEIT	Zeitabfrage . . . . .	251

## Allgemeines

Jedes Kommando beginnt mit einem Nummernzeichen (#). Danach folgt der Kommandoname, der nach dem zweiten Zeichen (mit oder ohne Punkt) an beliebiger Stelle abgekürzt werden kann. Eine Ausnahme bilden die Kommandos `EDIERE` und `TUE`, die mit `E` bzw. `T` abgekürzt werden können, sowie die mit `MB` beginnenden Kommandos, die erst nach dem dritten Zeichen abgekürzt werden können.

In der Regel folgen anschließend noch Spezifikationswerte für die einzelnen Spezifikationen. Jede Spezifikation ist durch einen Spezifikationsnamen benannt. Die Spezifikationen werden voneinander und vom Kommandonamen durch ein Komma getrennt. Für manche Spezifikationen sind mehrere Spezifikationswerte zugelassen, die durch Apostroph getrennt werden.

Die Spezifikationsangabe kann die Form einer Wertzuweisung haben. Dabei steht links vom Gleichheitszeichen der Spezifikationsname und rechts davon der Spezifikationswert. Spezifikationsnamen können ebenfalls abgekürzt werden, müssen aber für das jeweilige Kommando noch eindeutig sein.

Die Reihenfolge der Spezifikationen ist für jedes Kommando festgelegt und entspricht der Reihenfolge in der jeweiligen Beschreibung. Der Spezifikationsname und das Gleichheitszeichen können weggelassen werden, falls diese Reihenfolge eingehalten wird. Wenn der Spezifikationsname angegeben wird, muss die Reihenfolge nicht eingehalten werden. Innerhalb eines Kommandos können beide Möglichkeiten gemischt auftreten.

Wird zu einer Spezifikation keine Angabe gemacht, so gilt für diese Spezifikation der voreingestellte Wert. Dieser ist in der jeweiligen Beschreibung der einzelnen Kommandos angegeben und mit einem Stern (\*) gekennzeichnet. Die Voreinstellungen können nicht umdefiniert werden, jedoch können eigene Kommandos (Makros) mit eigenen Voreinstellungen definiert werden.

An Stelle eines Spezifikationswertes kann auch ein in spitze Klammern eingeschlossener Name einer zuvor definierten TUSTEP-Variablen (siehe Kommando `#DEFINIERE` Seite 132) angegeben werden. In diesem Fall wird der Inhalt der TUSTEP-Variablen als Spezifikationswert eingesetzt. Der Spezifikationswert darf keine Kommata enthalten.

Ein Kommando kann in einer oder in mehreren Zeilen geschrieben werden. Nach jedem Komma, Apostroph oder Gleichheitszeichen kann eine neue Zeile begonnen werden. Eine Kennzeichnung der Fortsetzungszeilen ist nicht notwendig.

Die Kommandos können in Groß- und/oder in Kleinbuchstaben geschrieben werden. Vor und nach jedem Nummernzeichen, Komma, Gleichheitszeichen und Apostroph können Leerzeichen eingefügt sein; sie sind an diesen Stellen ohne Bedeutung.

Steht nach einem Nummernzeichen, das den Beginn eines neuen Kommandos kennzeichnet, ein `»=«`, `»+«` oder `»-«` so gilt dies als Beginn eines Kommentars. Dieser endet, falls in der gleichen Zeile noch ein Nummernzeichen folgt, vor diesem; andernfalls endet der Kommentar vor der nächsten Zeile, die mit einem Nummernzeichen beginnt. Kommentare, die mit `»=«` gekennzeichnet sind, werden ins Ablaufprotokoll ausgegeben, falls die Ausgabe des Ablaufprotokolls nicht unterdrückt wird (siehe

Kommando #PROTOKOLL Seite 209). Mit »+« gekennzeichnete Kommentare werden immer ins Ablaufprotokoll ausgegeben, mit »-« gekennzeichnete werden nicht ausgegeben. Beginnt der Kommentar mit »@@@@ «, so wird er in der Farbe ausgegeben, die für Warnungen eingestellt ist; beginnt er mit »@@@@@@@@ «, so wird er in der Farbe ausgegeben, die für Fehlermeldungen eingestellt ist; andernfalls wird er in der Farbe ausgegeben, die für die Kommandoausgabe eingestellt ist. Diese Farben können mit dem Kommando #DEFINIERE (siehe Seite 132) eingestellt werden.

Daten, die von den Kommandos verarbeitet oder ausgewertet werden, können bei manchen Kommandos nicht nur in Dateien, sondern auch unmittelbar nach dem Kommando stehen. Zur entsprechenden Spezifikation muss in diesem Fall statt eines Dateinamens ein Stern angegeben werden. Die nach dem Kommando stehenden Daten müssen mit einer Zeile abgeschlossen werden, in der nur linksbündig \*EOF steht.

Bei einigen Kommandos kann es vorkommen, dass Daten, die hinter dem Kommando folgen, Zeilen mit \*EOF enthalten. Damit eine solche Zeile nicht als Datenabschluss gewertet wird, muss sie mit einem Stern markiert werden, d. h. an Stelle von \*EOF muss \*EOF\* stehen. Beim Einlesen der Daten wird der Markierungsstern automatisch entfernt. Bei entsprechender Kommandoverschachtelung kann es notwendig sein, ein \*EOF mehrfach zu markieren. Beim Einlesen wird dann jeweils ein Markierungsstern entfernt, bis schließlich ein unmarkiertes \*EOF übrigbleibt, das als Datenabschluss gewertet wird.

Im folgenden Beispiel wird eine Datei mit dem Namen txt eingerichtet. Danach wird der Editor gerufen, mit dem die Datei txt bearbeitet (in diesem Fall Text eingetragen) wird. Anschließend wird der Inhalt der Datei txt formatiert. Dabei soll der Zeichenvorrat des HP LaserJet zur Verfügung gestellt (DRT = Druckertyp) und oben auf jeder Seite in der Mitte zwischen zwei Minuszeichen eine Seitennummer eingesetzt werden (KT = Kopftext). Als letztes werden die formatierten Daten auf dem HP LaserJet ausgedruckt.

```
#DATEI, txt, SEQ-P #EDIERE, txt, T
#FORMATIERE, txt, LOESCHEN=+, PARAMETER=*
DRT      HPLJ
KT       / @z - &#3 - /
*EOF
#DRUCKE, , HPLJ
```

Soll nun obige Kommandofolge z. B. mit dem Kommando #UMWANDLE in eine Datei mit dem Namen cmd eingetragen werden, so kann dies wie folgt geschehen:

```
#UMWANDLE, *, cmd, 0, +
#DATEI, txt, SEQ-P #EDIERE, txt, T
#FORMATIERE, txt, LOESCHEN=+, PARAMETER=*
DRT      HPLJ
KT       / @z - &#3 - /
*EOF*
#DRUCKE, , HPLJ
*EOF
```

Dazu muss in der Zeile \*EOF (vor #DRUCKE), die die Parameter für das Kommando #FORMATIERE abschließt, ein Stern ergänzt werden, damit #UMWANDLE dieses \*EOF nicht als Datenabschluss wertet. Dieser Markierungsstern wird von #UMWANDLE entfernt, d. h. in der Datei cmd steht dann \*EOF. Die Zeile mit \*EOF (nach #DRUCKE) schließt die Daten für das Kommando #UMWANDLE ab; sie wird nicht in die Datei cmd übertragen.



## Eingabe

Mit der Eingabeaufforderung

```
Gib Kommando >
```

werden Kommandos angefordert, durch die TUSTEP mitgeteilt wird, was es tun soll. Nachdem ein Kommando geschrieben ist, muss es mit der Return-Taste abgeschickt werden. Erst dann wird es interpretiert und ausgeführt.

Endet eine Kommandozeile mit einem Komma, einem Apostroph oder einem Gleichheitszeichen, so wird angenommen, dass das Kommando noch unvollständig ist, und die Fortsetzung des Kommandos mit

```
Gib Spezifikationen >
```

angefordert. Soll das Kommando nicht weiter ergänzt werden, so kann eine leere Eingabezeile mit der Return-Taste abgeschickt werden.

Manche Programme erwarten noch Daten (z. B. Parameter) unmittelbar nach dem Kommando, mit dem sie aufgerufen werden. Diese Daten werden je nach ihrem Zweck mit verschiedenen lautenden Eingabeaufforderungen verlangt. Jede Zeile muss einzeln mit der Return-Taste abgeschickt werden. Das Ende solcher Daten kann dem Programm durch Eingabe einer Zeile, in der nur linksbündig \*EOF steht, mitgeteilt werden.

Die jeweils letzten 80 eingegebenen Zeilen werden gemerkt. Sie können zur Information aufgelistet werden oder einzeln in die Kommandozeile ausgegeben und (ggf. nach einer Korrektur) wieder als neue Eingabe abgeschickt werden.

Um die Eingabe zu erleichtern, gibt es einige Tastenkombinationen. Dies sind weiter unten im Abschnitt »Tastenkombinationen« beschrieben.

Für Kommandos, die wiederholt ausgeführt werden sollen, empfiehlt es sich, diese mit dem Editor in eine Datei einzutragen. Dann können sie bei Bedarf mit dem Kommando #TUE (siehe Seite 233) ausgeführt werden, ohne dass sie jedesmal neu eingegeben werden müssen. Beim Kommando #TUE muss in diesem Fall nur der Name dieser Datei angegeben werden.

Das gleiche gilt für Kommandos, die anschließend noch die Eingabe von Daten wie z. B. Parameter erwarten. Stehen sie in einer Datei, dann müssen bei einem möglichen Tippfehler, der die Wiederholung des Kommandos erfordert, nicht alle Parameter nochmals eingegeben werden, sondern es genügt die Korrektur des Tippfehlers in der Datei und die Wiederholung des Kommandos #TUE.

Sollen solche Kommandofolgen aufbewahrt werden, so empfiehlt es sich, nicht jede Kommandofolge in einer eigenen Datei zu speichern, sondern als ein Segment in einer Segment-Datei zu speichern. Dazu wird die Kommandofolge, nachdem sie ausgetestet ist, im Editor mit der Rette-Anweisung (siehe Seite 282) als Segment in eine Segment-Datei gespeichert. Soll ein Segment korrigiert werden, so kann es im Editor mit der Hole-Anweisung (siehe Seite 281) in eine Hilfsdatei kopiert und nach der Korrektur wieder mit der Rette-Anweisung in der Segment-Datei abgespeichert werden. Die Ausführung der in einem Segment stehenden Kommandofolge kann ebenfalls mit dem Kommando #TUE erfolgen. Dabei muss nach dem Namen der Segment-Datei noch der Name des jeweiligen Segments angegeben werden.

Soll eine Kommandofolge jeweils geringfügig modifiziert ausgeführt werden, so kann sie Platzhalter enthalten, die dann beim Aufruf mit #TUE durch aktuelle Angaben ersetzt werden. Fortgeschrittenen wird jedoch die Verwendung von Makros (siehe Kapitel »Makros« ab Seite 403) empfohlen. Mit ihnen kann eine Kommandofolge in vielfältiger Weise modifiziert werden, nicht nur auf Grund der Angaben beim Aufruf des Makros, sondern z. B. auch in Abhängigkeit von den Antworten auf die im Makro enthaltenen Fragen.

## Tastenbezeichnungen

Shift	»Umschalttaste« für Großbuchstaben.
Ctrl	»Control-Taste« links im Buchstabenblock unter der Shift-Taste; auf deutschen Tastaturen meist mit »Strg« beschriftet.
Return	»Eingabetaste« rechts im Buchstabenblock über der Shift-Taste.
Enter	»Eingabetaste« rechts unten im Ziffernblock. Alternativ kann in TUSTEP auch Shift+Return oder Ctrl+E verwendet werden.
Backspace	»Rücktaste« rechts im Buchstabenblock über der »Eingabetaste«.
Escape	»Abbruchtaste« ganz oben links, sie ist mit »Esc« beschriftet.

Die folgenden Tasten befinden sich im Mittelblock zwischen Buchstabenblock und Ziffernblock. Gleichbeschriftete Tasten im Ziffernblock können eventuell alternativ verwendet werden. Bei Tastaturen ohne Ziffernblock sind die folgenden Tasten in der rechten oberen und rechten unteren Ecke der Tastatur und erfordern eventuell ein zusätzliches Drücken der Fn-Taste.

Insert	»Einfügetaste«, sie ist mit »Einfg«, »Insert« oder »Ins« beschriftet. Auf MAC-Tastaturen fehlt diese Taste.
Delete	»Löschtaste«, sie ist mit »Entf« oder »Del« beschriftet. Alternativ kann in TUSTEP auch Shift+Backspace verwendet werden.
PfO	»Pfeil-nach-oben-Taste«
PfU	»Pfeil-nach-unten-Taste«
PfL	»Pfeil-nach-links-Taste«
PfR	»Pfeil-nach-rechts-Taste«
Pos1	»Anfangtaste«, sie ist mit »Pos1«, »Home« oder einem Pfeil nach links oben beschriftet. Unter macOS kann alternativ fn+PfL verwendet werden.
End	»Endetaste«, sie ist mit »Ende«, »End« oder einem Pfeil nach rechts unten beschriftet. Unter macOS kann alternativ fn+PfR verwendet werden.

PgUp	»Bild-rauf-Taste«, sie ist mit »Page Up«, »PgUp« oder mit »Bild« und einem Pfeil nach oben beschriftet. Unter macOS kann alternativ <code>fn+PfO</code> verwendet werden.
PgDn	»Bild-runter-Taste«, sie ist mit »Page Down«, »PgDn« oder mit »Bild« und einem Pfeil nach unten beschriftet. Unter macOS kann alternativ <code>fn+PfU</code> verwendet werden.

## Tastenkombinationen

Außer den im Folgenden aufgeführten Tastenkombinationen können auch noch Funktionstasten verwendet werden. Welche Voreinstellungen es gibt und wie sie definiert werden können, ist beim Kommando `#DEFINIERE` (siehe Seite 132) beschrieben.

Die im Folgenden aufgeführten Tastenkombinationen wirken nur dann alle wie nachfolgend beschrieben, wenn der Protokoll-Modus auf `PORTIONIERT` oder `FREILAUFEND` eingestellt ist. Wurde er mit dem Kommando `#PROTOKOLL` (siehe Seite 209) auf `SYSTEM` gestellt, so haben sie die vom jeweiligen Betriebssystem vorgegebene Wirkung.

Welche Tasten mit der in der linken Spalte angegebenen Tastenbezeichnungen gemeint sind, ist im vorangehenden Kapitel »Tastenbezeichnungen« auf Seite 114 beschrieben.

<code>Ctrl+R</code>	Listet die letzten 22 eingegebenen Zeilen (bzw. die mit dem Kommando <code>#DEFINIERE</code> eingestellte Anzahl Zeilen abzüglich 3 Zeilen) auf; wird der Steuerbefehl wiederholt, werden die restlichen gemerkten Zeilen aufgelistet.
<code>Ctrl+G</code>	Zeigt in einem Fenster die letzten eingegebenen Zeilen an. Davon kann ggf. eine Zeile ausgewählt werden um sie verändert oder unverändert nochmals einzugeben (siehe Kommando-Manager Seite 117).
<code>PfO</code>	Zeigt die jeweils davor eingegebene Zeile an; dies ermöglicht ein Zurückblättern innerhalb der letzten 80 eingegebenen Zeilen.
<code>PfU</code>	Zeigt jeweils die danach eingegebene Zeile an; dies ermöglicht (nach Zurückblättern mit <code>PfO</code> oder <code>PgUp</code> ) ein Vorblättern innerhalb der letzten 80 eingegebenen Zeilen.
<code>PgUp</code>	Zeigt die jeweils davor eingegebene Zeile an, die mit der links vom Cursor stehenden Zeichenfolge beginnt; dies ermöglicht ein Zurückblättern innerhalb der letzten 80 eingegebenen Zeilen.
<code>Ctrl+PfO</code>	Wirkt gleich wie <code>PgUp</code>
<code>PgDn</code>	Zeigt die jeweils danach eingegebene Zeile an, die mit der links vom Cursor stehenden Zeichenfolge beginnt; dies ermöglicht (nach Zurückblättern mit <code>PfO</code> oder <code>PgUp</code> ) ein Vorblättern innerhalb der letzten 80 eingegebenen Zeilen.

---

Ctrl+PfU	Wirkt gleich wie PgDn
PfR	Cursor springt um ein Zeichen nach rechts.
PfL	Cursor springt um ein Zeichen nach links.
TAB	Wenn für ein parametergesteuertes Programm (z. B. #KOPIERE) Parameter eingegeben werden, springt der Cursor auf die nächstfolgende Tabulatorposition; Tabulatorpositionen sind 11, 21, ..., 71.  In allen anderen Fällen wird ein Dateiname, dessen Anfang unmittelbar links vom Cursor steht, vervollständigt. Dabei werden nur Namen von angemeldeten Dateien (einschließlich Scratch-Dateien) berücksichtigt. Ist der Anfang des Dateinamens nicht eindeutig, so wird beim ersten Mal der Name bis zu der Stelle ergänzt, an der er mehrdeutig wird; danach wird jeweils der nächste nach alphabetischer Reihenfolge in Frage kommende Dateiname angezeigt. Steht links vom Cursor ein Leerzeichen, Komma, Gleichheitszeichen oder Apostroph, wird »-STD-« ergänzt.
Pos1	Cursor springt an den Anfang der Kommandozeile.
End	Cursor springt an das Ende der Kommandozeile.
Ctrl+PfR	Cursor springt an den Anfang des nächsten Wortes. Steht der Cursor im letzten Wort, so springt er hinter dieses Wort auf die Position, an der das nächste Wort beginnen kann; steht der Cursor schon hinter dem letzten Wort, springt er an den Anfang der Zeile. Als »Wort« gelten Zeichenfolgen, die durch Leerzeichen und/oder Komma begrenzt sind.
Ctrl+PfL	Cursor springt an den Anfang des Wortes, in dem er steht; steht der Cursor am Anfang eines Wortes, springt er an den Anfang des vorhergehenden Wortes; steht der Cursor am Anfang des ersten Wortes oder davor, so springt er hinter das letzte Wort auf die Position, an der das nächste Wort beginnen kann. Als »Wort« gelten Zeichenfolgen, die durch Leerzeichen und/oder Komma begrenzt sind.
Ctrl+V	Fügt den Inhalt der Zwischenablage vor der aktuellen Cursor-Position ein.
CMD+V	Nur macOS: Fügt den Inhalt des Pasteboards vor der aktuellen Cursor-Position ein.
Ctrl+T	Tauscht das Zeichen auf der aktuellen Cursor-Position mit dem unmittelbar davor stehenden Zeichen aus.
Delete	Löscht das Zeichen auf der aktuellen Cursor-Position und verschiebt alle in der Zeile folgenden Zeichen um eine Position nach links.

Backspace	Löscht das Zeichen links von der aktuellen Cursor-Position und verschiebt alle in der Zeile folgenden Zeichen um eine Position nach links. Der Cursor rückt ebenfalls um eine Position nach links.
Insert	Schaltet vom Einfügemodus in den Überschreibmodus und umgekehrt. Im Überschreibmodus werden bereits vorhandene Zeichen überschrieben; im Einfügemodus werden die neu eingegebenen Zeichen an der aktuellen Cursor-Position eingeschoben, d. h. alle Zeichen von der Cursor-Position bis zum Zeilenende werden nach rechts verschoben.
Enter	Ende der Eingabe.
Return	Ende der Eingabe.
Escape	<ul style="list-style-type: none"> <li>- bei Dateneingabe: Dateneingabe wird beendet; evtl. noch in der Eingabezeile stehende Daten werden ignoriert.</li> <li>- sonst: TUSTEP-Sitzung wird unterbrochen; evtl. noch in der Eingabezeile stehende Daten werden ignoriert.</li> </ul>

## Kommando-Manager

Der Kommando-Manager kann mit der Tastenkombination `Ctrl+G` oder mit einer entsprechend definierten Funktionstaste (siehe »Belegen der Funktionstasten« Seite 139) aufgerufen werden. Er zeigt in einem Fenster die letzten eingegebenen Kommandos und Daten an, z. B.:

Kommando-Manager	Liste der eingegebenen Kommandos und Daten	Beenden
<pre>#definiere,dateiname=* dld*test.xml = ~/Downloads/test.xml *eof* #anmelde,dld*test.xml,traeger=- #datei,test.tf,seq-ap #*import,dld*test.xml,test.tf,,+,ignore=- #ediere,test.tf #datei,test.rtf,seq-ap #*export,test.tf,test.rtf,,+,anzeigen=+ #ediere #*export,test.tf,test.rtf,,+,anzeigen=+</pre>		
Auswahl mit Maus-Klick oder mit Pfeil nach oben/unten und Return/Enter		

In diesem Fenster sind folgende Aktionen möglich:

- Mit dem Mousrad und mit den Tasten `PfO/PfU` kann die Liste gescrollt werden, mit den Tasten `PgUp/PgDn` kann in der Liste geblättert werden.
- Um ein Kommando nochmals unverändert auszuführen bzw. eine Datenzeile nochmals unverändert einzugeben, kann die entsprechende Zeile mit der linken Maustaste angeklickt werden oder der Cursor mit den Tasten `PfO/PfU` in die entsprechende Zeile positioniert und die Taste `Return` gedrückt werden. In beiden Fällen wird das Fenster automatisch geschlossen.

- Um ein Kommando bzw. eine Datenzeile in die Kommandozeile auszugeben, um es/sie (ggf. nach einer Korrektur) abzuschicken, kann die entsprechende Zeile mit der rechten Maustaste angeklickt werden oder der Cursor mit den Tasten `PfO/PfU` in die entsprechende Zeile positioniert und die Taste `Enter` gedrückt werden. In beiden Fällen wird das Fenster automatisch geschlossen.
- Um das Fenster ohne Auswahl einer Zeile zu schließen, kann mit der linken Maustaste auf die Schaltfläche »Beenden« geklickt werden oder die Taste `Escape` gedrückt werden.
- Mit der Taste `Delete` können Zeilen aus der Liste entfernt werden. Sie werden jedoch erst beim Schließen des Fensters endgültig entfernt. Dies kann vermieden werden, indem das Fenster mit der Taste `Escape` geschlossen wird.

Wurde mit einem Mausklick oder einer der Tasten `Return` oder `Enter` eine der gemerkten Zeile ausgewählt, so kann unmittelbar danach mit der Taste `PfU` die jeweils nachfolgende Zeile in die Kommandozeile geholt werden.

# Abmelden von Dateien

#ABMELDE

## Kommando

#ABMELDE

DATEI	= datei	Name der abzumeldenden Datei.  Es können auch mehrere Dateinamen angegeben werden; sie müssen durch Apostroph getrennt sein.  Zu den Spezifikationen PROJEKT, TRAEGER, POSITIV und NEGATIV darf nur die Voreinstellung der jeweiligen Spezifikation angegeben werden; sie ist in diesem Fall aber bedeutungslos.
	= -	* Keine Datei abmelden.
	= +	Die mit den Spezifikationen PROJEKT/TRAEGER/POSITIV/NEGATIV ausgewählten Dateien abmelden.
PROJEKT	= name	Name des Projekts, von dem Dateien abgemeldet werden sollen.
	= +	Dateien des aktuellen Projekts abmelden.
	= -STD-	Dateien des Projekts abmelden, das beim Initialisieren der TUSTEP-Sitzung eingestellt wurde; dieses ist durch die System-Variable TUSTEP_PRJ vorgegeben.
	= -	Dateien abmelden, die keinem Projekt zugeordnet sind.
	= ?	* Dateien unabhängig davon abmelden, welchem Projekt sie zugeordnet sind.
TRAEGER	= name	Name der System-Variablen, die den Pfad für die abzumeldenden Dateien enthält.  Unter Windows kann auch direkt der Laufwerksbuchstabe angegeben werden, falls der Pfad keinen Verzeichnisnamen enthält.
	= -STD-	Dateien abmelden, die sich auf dem Standard-Träger für permanente Dateien befinden; dieser ist durch die System-Variable TUSTEP_DSK vorgegeben.
	= -	Dateien mit »definierten Dateinamen« abmelden.

	= ?	* Dateien unabhängig davon abmelden, auf welchem Träger sie sich befinden.
POSITIV	= -	* Keine Positiv-Auswahl von Dateien, deren Name einem vorgegebenen Muster entspricht.
	= . . .	Nur solche Dateien abmelden, deren Dateinamen mindestens einem der angegebenen Muster entspricht.
NEGATIV	= -	* Keine Negativ-Auswahl von Dateien, deren Name einem vorgegebenen Muster entspricht.
	= . . .	Nur solche Dateien abmelden, deren Dateinamen keinem der angegebenen Muster entspricht.
WISCHEN	= +	Bei temporären Dateien, die abgemeldet und damit gelöscht werden, zuvor die in der Datei stehenden Daten überschreiben.
	= -	Bei temporären Dateien, die abgemeldet und damit gelöscht werden, die in der Datei stehenden Daten nicht überschreiben.
FRAGEN	= -STD-	* Permanente Dateien ohne nachzufragen abmelden. Bei temporären Dateien (Scratch-Dateien) jeweils nachfragen, ob die Datei gelöscht werden darf.
	= +	Sowohl bei permanenten Dateien als auch bei temporären Dateien jeweils nachfragen, ob die Datei abgemeldet bzw. gelöscht werden darf.
	= -	Permanente Dateien ohne nachzufragen abmelden, temporäre Dateien ohne nachzufragen löschen.

## Leistung

Mit diesem Kommando können Dateien abgemeldet werden. Damit wird verhindert, dass nachfolgende TUSTEP-Programme (z. B. wegen Tippfehler) versehentlich auf diese Dateien zugreifen.

Sollen alle angemeldeten Dateien eines bestimmten Projekts abgemeldet werden, genügt es, ein »+« zur Spezifikation `DATEI` und den Namen dieses Projekt zur Spezifikation `PROJEKT` anzugeben. Entsprechend können alle angemeldeten Dateien eines Datenträgers durch eine entsprechende Angabe zur Spezifikation `TRAEGER` abgemeldet werden. Wird zu beiden Spezifikationen etwas angegeben, so werden alle angemeldeten Dateien dieses Projekts auf diesem Datenträger abgemeldet.

Sollen alle Dateien abgemeldet werden, deren Dateinamen eine bestimmte Zeichenfolge enthalten bzw. nicht enthalten, so können zur Spezifikation `POSITIV` Muster angegeben werden, von denen mindestens eines dem Dateinamen entsprechen muss, damit die Datei abgemeldet wird; zur Spezifikation `NEGATIV` können Muster angegeben werden, von denen keines dem Dateinamen entsprechen darf, damit die Datei abgemeldet wird. Die möglichen Angaben sind beim Kommando `#LISTE` auf Seite 178 beschrieben.



Werden Scratch-Dateien abgemeldet, so werden sie gelöscht. Damit steht der Speicherplatz wieder für andere Dateien zur Verfügung. Ob zuvor die in solchen Dateien stehenden Daten überschrieben (Datenschutz!) werden sollen, kann mit der Spezifikation `WISCHEN` angegeben werden. Wird nichts angegeben, so werden die Daten überschrieben, wenn nicht mit dem Kommando `#WISCHEN` (siehe Seite 250) ein anderer Modus eingestellt wurde.

Falls auf Grund der Spezifikation `FRAGEN` nachgefragt wird, ob eine Datei abgemeldet bzw. gelöscht werden soll, sind vier Antworten, die abgekürzt werden können, vorgesehen:

- 1) `JA` Die Datei abmelden bzw. löschen.
- 2) `NEIN` Die Datei nicht abmelden bzw. nicht löschen.
- 3) `ALLE` Die Datei und alle noch zum Abmelden bzw. Löschen vorgesehenen Dateien ohne nachzufragen abmelden bzw. löschen.
- 4) `KEINE` Die Datei und alle noch zum Abmelden bzw. Löschen vorgesehenen Dateien nicht abmelden bzw. nicht löschen.

Falls mit dem Kommando `#PROTOKOLL` (siehe Seite 209) der Protokoll-Modus nicht auf `SYSTEM` eingestellt wurde, kann statt `KEINE` einzugeben auch die Escape-Taste gedrückt werden.

## Beispiele

Datei mit dem Namen `arbeit` abmelden:

```
#AB, arbeit
```

Alle angemeldeten Dateien des Projekts `doku` abmelden:

```
#AB, +, PR=doku
```

Alle angemeldeten Dateien mit der Namenserweiterung `xy` abmelden:

```
#AB, +, POS=*.xy
```

Unter Windows alle angemeldeten Dateien, die auf dem USB-Speicher-Stick mit dem Laufwerksbuchstaben `E` stehen, abmelden:

```
#AB, +, TR=E
```

# Ändern von Dateinamen und Projektnamen

#AENDERE

## Kommando

#AENDERE

- DATEINAME = - \* Keine Datei umbenennen.  
= name1:name2 Alter und neuer Name der Datei, deren Name geändert werden soll.  
Es können auch mehrere Dateinamen-Paare angegeben werden; sie müssen durch Apostroph getrennt sein.
- PROJEKTNAME = - \* Kein Projekt umbenennen.  
= name1:name2 Alter und neuer Name des Projekts, dessen Name geändert werden soll.  
Es können auch mehrere Projektnamen-Paare angegeben werden; sie müssen durch Apostroph getrennt sein.
- TRAEGER = name Name der System-Variablen, die den Pfad für die Projekte enthält, deren Name geändert werden soll.  
Unter Windows kann auch direkt der Laufwerksbuchstabe angegeben werden, falls der Pfad keinen Verzeichnisnamen enthält.  
= -STD- \* Die Projekte, deren Name geändert werden soll, befinden sich auf Standard-Träger für permanente Dateien; dieser ist durch die System-Variable TUSTEP\_DSK vorgegeben.

## Leistung

Mit diesem Kommando können die Namen von Dateien und Projekten (Verzeichnissen) geändert, d. h. umbenannt werden.

Werden beim Umbenennen von Dateien der alte und neue Dateiname mit verschiedenen Projektnamen ergänzt, so wird die Datei damit vom alten ins neue Projekt verschoben (d. h. sie wird im alten Projekt gelöscht und in das neue Projekt auf dem selben Träger eingetragen); das Projekt, in das die Datei verschoben werden soll, muss schon existieren.

## Hinweis

Die Träger der Dateien bzw. der Projekte können nicht geändert werden. Eine Angabe zur Spezifikation TRAEGER gilt nur für die Projekte, die umbenannt werden.

## Beispiele

Name der Datei `arbeit` in `hausarbeit` ändern:

```
#AE, arbeit:hausarbeit
```

Datei mit dem Namen `arbeit` vom Projekt `doku` in das Projekt `archiv` verschieben:

```
#AE, doku*arbeit:archiv*arbeit
```

Name des Projekts `mail` in `briefe` ändern:

```
#AE, , mail:briefe
```

# Anmelden von Dateien

#ANMELDE

## Kommando

#ANMELDE

LESEN	= datei	Name der zum Lesen anzumeldenden Datei. Es können auch mehrere Dateinamen angegeben werden; sie müssen durch Apostroph getrennt sein. Zu den Spezifikationen PROJEKT, POSITIV und NEGATIV darf nur die Voreinstellung der jeweiligen Spezifikation angegeben werden; sie ist in diesem Fall aber bedeutungslos.
	= -	* Keine Datei zum Lesen anmelden.
	= +	Die mit den Spezifikationen PROJEKT/TRAEGER/POSITIV/NEGATIV ausgewählten Dateien zum Lesen anmelden.
SCHREIBEN	= datei	Name der zum Schreiben anzumeldenden Datei. Es können auch mehrere Dateinamen angegeben werden; sie müssen durch Apostroph getrennt sein. Zu den Spezifikationen PROJEKT, POSITIV und NEGATIV darf nur die Voreinstellung der jeweiligen Spezifikation angegeben werden; sie ist in diesem Fall aber bedeutungslos.
	= -	* Keine Datei zum Schreiben anmelden.
	= +	Die mit den Spezifikationen PROJEKT/TRAEGER/POSITIV/NEGATIV ausgewählten Dateien zum Schreiben anmelden.
SCRATCH	= -	* (nicht mehr definiert)
PROJEKT	= name	Name des Projekts, von dem Dateien angemeldet werden sollen.
	= +	* Dateien des aktuellen Projekts anmelden.
	= -STD-	Dateien des Projekt anmelden, das beim Initialisieren der TUSTEP-Sitzung eingestellt wurde; dieses ist durch die System-Variable TUSTEP_PRJ vorgegeben.
	= -	Dateien anmelden, die keinem Projekt zugeordnet sind.

TRAEGER	= name	Name der System-Variablen, die den Pfad für die anzumeldenden Dateien enthält.  Unter Windows kann auch direkt der Laufwerksbuchstabe angegeben werden, falls der Pfad keinen Verzeichnisnamen enthält.
	= -STD-	* Dateien anmelden, die sich auf dem Standard-Träger für permanente Dateien befinden; dieser ist durch die System-Variable TUSTEP_DSK vorgegeben.
	= -	Dateien mit »definierten Dateinamen« anmelden.
POSITIV	= -	* Keine Positiv-Auswahl von Dateien, deren Name einem vorgegebenen Muster entspricht.
	= . . .	Nur solche Dateien anmelden, deren Dateinamen mindestens einem der angegebenen Muster entspricht.
NEGATIV	= -	* Keine Negativ-Auswahl von Dateien, deren Name einem vorgegebenen Muster entspricht.
	= . . .	Nur solche Dateien anmelden, deren Dateinamen keinem der angegebenen Muster entspricht.
FRAGEN	= -STD-	* Gleichbedeutend mit FRAGEN=-.
	= +	Vor dem Anmelden einer Datei jeweils nachfragen, ob sie angemeldet werden soll.
	= -	Dateien ohne nachzufragen anmelden.

## Leistung

Mit diesem Kommando können permanente Dateien angemeldet werden. Dies ist notwendig, um auf solche Dateien zugreifen zu können. In Dateien, die zum Lesen angemeldet sind, kann nur gelesen werden; in solchen, die zum Schreiben angemeldet sind, kann gelesen und geschrieben werden.

Sollen alle Dateien eines bestimmten Projekts angemeldet werden, genügt es, ein »+« zur Spezifikation LESEN bzw. SCHREIBEN und den Namen dieses Projekts zur Spezifikation PROJEKT anzugeben.

Sollen alle Dateien angemeldet werden, deren Dateinamen eine bestimmte Zeichenfolge enthalten bzw. nicht enthalten, so können zur Spezifikation POSITIV Muster angegeben werden, von denen mindestens eines dem Dateinamen entsprechen muss, damit die Datei angemeldet wird; zur Spezifikation NEGATIV können Muster angegeben werden, von denen keines dem Dateinamen entsprechen darf, damit die Datei angemeldet wird. Die möglichen Angaben sind beim Kommando #LISTE auf Seite 178 beschrieben.

Falls auf Grund der Spezifikation FRAGEN nachgefragt wird, ob eine Datei angemeldet werden soll, sind vier Antworten vorgesehen, die abgekürzt werden können:

- 1) JA Die Datei anmelden.
- 2) NEIN Die Datei nicht anmelden.
- 3) ALLE Die Datei und alle noch zum Anmelden vorgesehenen Dateien ohne nachzufragen anmelden.
- 4) KEINE Die Datei und alle noch zum Anmelden vorgesehenen Dateien nicht anmelden.

Falls mit dem Kommando #PROTOKOLL (siehe Seite 209) der Protokoll-Modus nicht auf SYSTEM eingestellt wurde, kann statt KEINE einzugeben auch die Escape-Taste gedrückt werden.

## Hinweise

Werden Dateien durch Angabe von »+« zur Spezifikation LESEN oder SCHREIBEN angemeldet, so bleiben Dateien, die einen für TUSTEP unzulässigen Namen haben, in jedem Fall unberücksichtigt.

Die Spezifikation PROJEKT ist nur von Bedeutung, wenn zur Spezifikation LESEN oder SCHREIBEN ein »+« angegeben ist. Werden zu diesen beiden Spezifikationen Dateinamen angegeben, so müssen sie mit dem Projektnamen angegeben werden, falls sie nicht zum aktuellen Projekt gehören. Die Angabe des Projektnamens zur Spezifikation PROJEKT ist in diesem Fall wirkungslos.

Insgesamt können in einer TUSTEP-Sitzung etwa 4000 Dateien (ohne die von TUSTEP intern angemeldeten) gleichzeitig angemeldet sein. Dazu zählen auch Dateien, die mit dem Kommando #DATEI (siehe Seite 128) oder der Makrofunktion CREATE (siehe Seite 479) in derselben Sitzung eingerichtet werden. Ggf. müssen erst Dateien mit dem Kommando #ABMELDE (siehe Seite 119) oder der Makrofunktion CLOSE (siehe Seite 482) abgemeldet oder mit dem Kommando #LOESCHE (siehe Seite 182) oder der Makrofunktion DELETE (siehe Seite 484) gelöscht werden, bevor weitere Dateien angemeldet werden können.

## Beispiele

Datei mit dem Namen `arbeit` zum Schreiben anmelden:

```
#AN, , arbeit
```

Alle Dateien des Projekts `doku` zum Lesen anmelden:

```
#AN, +, PR=doku
```

Alle Dateien des aktuellen Projekts mit der Namenserweiterung `xy` zum Lesen anmelden:

```
#AN, +, POS=*.xy
```

Unter Windows alle Dateien, die im Hauptverzeichnis des USB-Speicher-Sticks mit dem Laufwerksbuchstaben `E` stehen, zum Lesen anmelden:

```
#AN, +, PR=-, TR=E
```

## Beenden/Unterbrechen einer TUSTEP-Sitzung

#BEENDE

### Kommando

#BEENDE

MODUS	= +	* TUSTEP-Sitzung beenden.
	= -	TUSTEP-Sitzung unterbrechen.
WISCHEN	= +	Daten in den temporären Dateien überschreiben, falls die TUSTEP-Sitzung beendet (und nicht nur unterbrochen) wird.
	= -	Daten in den temporären Dateien nicht überschreiben.

### Leistung

Mit diesem Kommando kann die TUSTEP-Sitzung beendet oder unterbrochen werden.

Wenn die TUSTEP-Sitzung beendet wird, werden

- alle Standard-Dateien gelöscht,
- alle temporären Dateien (Scratch-Dateien) gelöscht,
- alle angemeldeten Dateien abgemeldet.

Ob beim Löschen der temporären Dateien (Scratch-Dateien) die in diesen Dateien stehenden Daten überschrieben (Datenschutz!) werden sollen, kann mit der Spezifikation WISCHEN angegeben werden. Wird nichts angegeben, so werden die Daten überschrieben, wenn nicht mit dem Kommando #WISCHEN (siehe Seite 250) ein anderer Modus eingestellt wurde.

Wenn die TUSTEP-Sitzung nur unterbrochen wird, so bleiben alle Dateien und Einstellungen erhalten. Eine unterbrochene TUSTEP-Sitzung kann durch einen erneuten Aufruf von TUSTEP fortgesetzt werden.

# Einrichten von Dateien und Projekten

#DATEI

## Kommando

#DATEI

NAME = name Name der einzurichtenden Datei bzw. des einzurichtenden Projekts (Verzeichnisses).

Es können auch mehrere Datei- bzw. Projektnamen angegeben werden; sie müssen durch Apostroph getrennt sein.

TYP = typ Typ der einzurichtenden Datei.

= typ-opt Typ der einzurichtenden Datei, zusätzlich durch Optionen ergänzt.

Für typ sind folgende Angaben möglich:

\* SEQ TUSTEP-Datei mit sequentiellem Zugriff.

RAN TUSTEP-Datei mit wahlfreiem (random) Zugriff.

FDF Fremd-Datei im Fremd-Daten-Format (z. B. ASCII-Datei).

TAPE Band-Datei

Für opt sind folgende Angaben möglich:

A Datei anmelden, falls sie existiert.

\* T Temporäre Datei.

P Permanente Datei.

Ergänzende Hinweise zu typ und opt s. u. unter Dateityp bzw. Optionen.

= PROJEKT Projekt (Verzeichnis) mit dem zur Spezifikation NAME angegebenen Namen einrichten.

TRAEGER = name Name der System-Variablen, die den Pfad für die einzurichtenden Dateien bzw. das einzurichtende Projekt enthält.

Unter Windows kann auch direkt der Laufwerksbuchstabe angegeben werden, falls der Pfad keinen Verzeichnisnamen enthält.



	= -STD-	* Permanente Dateien bzw. Projekte auf dem Standard-Träger für permanente Dateien einrichten; dieser ist durch die System-Variable TUSTEP_DSK vorgegeben.  Temporäre Dateien (Scratch-Dateien) auf dem Standard-Träger für temporäre Dateien einrichten; dieser ist durch die System-Variable TUSTEP_SCR vorgegeben.
	= -	Zur Spezifikation DATEI sind »definierte Dateinamen« angegeben.
FRAGEN	= -STD-	* Im Dialog gleichbedeutend mit FRAGEN=+, im Batch gleichbedeutend mit FRAGEN=-.
	= +	Wird versucht eine Scratch-Datei einzurichten, die schon existiert, nachfragen, ob die darin enthaltenen Daten gelöscht werden dürfen.
	= -	Wird versucht eine Scratch-Datei einzurichten, die schon existiert, die darin enthaltenen Daten ohne nachzufragen löschen.

## Leistung

Mit diesem Kommando können Dateien eingerichtet werden. Sie werden dabei gleichzeitig zum Schreiben angemeldet (vergleiche Kommando #ANMELDE).

Die TUSTEP-Programme erwarten TUSTEP-Dateien (TYP=SEQ oder TYP=RAN). Beim Kommando #UMWANDLE und bei der Hole- und Rette-Anweisung im Editor dürfen auch Text-Dateien im Fremd-Daten-Format (TYP=FDF) angegeben werden. In Makros (siehe Kapitel »Makros« ab Seite 403) können TUSTEP-Dateien und Dateien im Fremd-Daten-Format bearbeitet werden. Band-Dateien (TYP=TAPE) können nur mit den mit #MB . . . beginnenden Kommandos und mit den Kommandos #RETTE und #HOLE beschrieben und gelesen werden.

Der Typ der Datei kann durch Angabe von Optionen genauer bestimmt werden. Die Optionen (einzelne Buchstaben) werden nach dem Typ angegeben und von diesem durch ein »-« getrennt (z. B.: SEQ-T).

Außerdem können mit diesem Kommando Projekte (Verzeichnisse) für Dateien eingerichtet werden. Die Aufteilung der Dateien auf verschiedene Projekte ermöglicht es, logisch zusammengehörende (z. B. zu einem Aufgabengebiet gehörende) Dateien in Gruppen zusammenzufassen und bietet so eine bessere Übersicht über die Dateien.

Falls auf Grund der Spezifikation FRAGEN nachgefragt wird, ob die Daten in einer bereits vorhandenen Scratch-Datei gelöscht werden dürfen, sind vier Antworten vorgesehen, die abgekürzt werden können:

- 1) JA Die Daten löschen.
- 2) NEIN Die Daten nicht löschen.

- 3) ALLE Die Daten löschen und bei allen noch zum Einrichten vorgesehenen Scratch-Dateien Daten ohne nachzufragen löschen.
- 4) KEINE Die Daten nicht löschen und bei allen noch zum Einrichten vorgesehenen Scratch-Dateien Daten ebenfalls nicht löschen.

Falls mit dem Kommando #PROTOKOLL (siehe Seite 209) der Protokoll-Modus nicht auf SYSTEM eingestellt wurde, kann statt KEINE einzugeben auch die Escape-Taste gedrückt werden.

## Dateityp

Für TUSTEP-Dateien sollte der Typ RAN gewählt werden, wenn die Daten im Textmodus nummeriert sind, die Satznummern alle aufsteigend und voneinander verschieden sind und die Daten nicht nur sequentiell gelesen werden, sondern auch gezielt auf Sätze mit bestimmten Satznummern zugegriffen wird. Dies trifft insbesondere auch auf Segment-Dateien zu. In allen anderen Fällen kann für TUSTEP-Dateien der Typ SEQ gewählt werden. Weitere Erläuterungen siehe Kapitel »Dateien« im Abschnitt »Zugriff auf die Sätze« auf Seite 34.

## Optionen

Die Option A darf nur als erste, von den Optionen T und P darf nur eine angegeben werden. Falls keine der Optionen T und P angegeben ist, wird die Option T angenommen. Ein Trennzeichen zwischen den Optionen ist nicht erforderlich.

- A Es wird erst versucht, die angegebene Datei zum Schreiben anzumelden. Existiert die Datei noch nicht, so wird sie entsprechend den weiteren Optionen eingerichtet.
- T Temporäre Datei (Scratch-Datei)
- P Permanente Datei

## Anmerkung

Insgesamt können in einer TUSTEP-Sitzung etwa 4000 Dateien (ohne die von TUSTEP intern angemeldeten) gleichzeitig angemeldet sein. Dazu zählen auch Dateien, die mit dem Kommando #DATEI oder der Makrofunktion CREATE (siehe Seite 479) in derselben Sitzung eingerichtet werden. Ggf. müssen erst Dateien mit dem Kommando #ABMELDE (siehe Seite 119) oder der Makrofunktion CLOSE (siehe Seite 482) abgemeldet oder mit dem Kommando #LOESCHE (siehe Seite 182) oder der Makrofunktion DELETE (siehe Seite 484) gelöscht werden, bevor neue Dateien eingerichtet werden können.

## Einschränkungen

Auf einem Träger kann nicht gleichzeitig eine Datei und ein Projekt mit dem gleichen Namen existieren.

Der Projektname »TUSTEP« ist für TUSTEP reserviert und darf nicht verwendet werden.

Datei- und Projektnamen in der Form »TUSTEP . xxx« (wobei xxx für eine beliebige Buchstaben-Ziffern-Kombination steht) sind für TUSTEP reserviert und dürfen nicht verwendet werden. Dazu gibt es eine Ausnahme:

Der Name TUSTEP . INI ist für die INI-Datei reserviert. In sie können Makros (siehe Kapitel »Makros« ab Seite 403), die beim Initialisieren und/oder beim Fortsetzen einer TUSTEP-Sitzung (siehe Seite 89) ausgeführt werden sollen, und Editor-Definitionen eingetragen werden.

In der Datei \*TUSTEP .MCR werden die Makro-Aufrufe mit ihren Spezifikationswerten gespeichert, die mit dem Makro \*M (siehe Seite 408) aufgerufen bzw. gemerkt werden.

In der Datei \*TUSTEP .CDS werden die Code-Auswahlen gespeichert, die mit dem Makro \*DECO (siehe Seite 138) definiert werden.

## Beispiele

Permanente Datei mit dem Namen `arbeit` einrichten:

```
#DA, arbeit, SEQ-P
```

Permanente Datei mit dem Namen `liste` anmelden; falls sie noch nicht vorhanden ist, Datei einrichten:

```
#DA, liste, SEQ-AP
```

Unter Windows Datei mit dem Namen `kopie` im Hauptverzeichnis des USB-Speicher-Sticks mit dem Laufwerksbuchstaben `E` einrichten:

```
#DA, -*kopie, SEQ-P, TR=E
```

## Definieren einer Makro-Datei, Variablen u. a.

**#DEFINIERE**

### Kommando

#DEFINIERE

MAKRO	= n:datei	Name der n-ten (n = 1 bis 3) Makro-Datei mit Makros. Es können auch zwei oder drei Makro-Dateien angegeben werden; die Angaben müssen durch Apostroph getrennt sein.
	= datei	Entspricht der Angabe 1:datei.
	= n:-	Definition der n-ten (n = 1 bis 3) Makro-Datei löschen. Es können auch zwei oder drei Definitionen von Makro-Dateien gleichzeitig gelöscht werden; die Angaben müssen durch Apostroph getrennt sein.
	= -	Entspricht der Angabe 1:-.
	= +	* Makro-Dateien beibehalten.
VARIABLEN	= name:wert	TUSTEP-Variable mit dem Namen name definieren und ihr den Wert wert zuweisen.
	= datei	Name der Datei mit Definitionen von TUSTEP-Variablen.
	= *	Definitionen von TUSTEP-Variablen folgen auf das #DEFINIERE-Kommando und sind mit *EOF abgeschlossen.
	= -	Alle TUSTEP-Variablen löschen.
	= +	* TUSTEP-Variablen beibehalten.
PROJEKT	= name	Dateinamen mit dem angegebenen Projektnamen ergänzen.
	= -STD-	Dateinamen mit dem Projektnamen ergänzen, der beim Initialisieren der TUSTEP-Sitzung eingestellt wurde; dieser ist durch die System-Variable TUSTEP_PRJ vorgegeben.
	= -	Dateinamen nicht automatisch mit einem Projektnamen ergänzen.
	= +	* Eingestellten Projektnamen beibehalten.

NAME	= name	Name, der bei Druckausgaben auf dem Umschlag (Deck- und Endblatt) verwendet werden soll.
	= +	* Eingestellten Namen beibehalten.
CODE	= -	Für das TUSTEP-Fenster den internationalen ASCII-Zeichensatz (siehe Tabelle Seite 815) einstellen; keine Umcodierung vornehmen.
	= -STD-	Standard-Code einstellen. Unter Linux und macOS ist dies ISO8859 und unter Windows CPnnn, wobei nnn die Nummer der vom Betriebssystem in TUSTEP-Fenstern verwendeten Code-Page ist.
	= IBMPC	Für das TUSTEP-Fenster den IBM-PC-Zeichensatz (siehe Tabelle Seite 817) einstellen.
	= ANSI	Wie CP1252.
	= CPnnn	Für das TUSTEP-Fenster die Code-Page nnn (nnn = 437, 850, 852, 1250, 1252) einstellen (siehe Tabellen ab Seite 818).
	= ISO8859	Für das TUSTEP-Fenster den Zeichensatz 1 der ISO-Norm 8859 (siehe Tabelle Seite 826) einstellen.
	= LINUX	Für das TUSTEP-Fenster den Linux-Zeichensatz (siehe Tabelle Seite 827) einstellen.
	= DECMCS	Für das TUSTEP-Fenster den »DEC Multinational Character Set« (siehe Tabelle Seite 825) einstellen.
	= ...	Unter Windows: Zeichenauswahl mit dem angegebenen Namen einstellen; Zeichenauswahlen können mit dem Makro *DECO (siehe Seite 138) definiert werden.
	= +	* Eingestellten Code für das TUSTEP-Fenster beibehalten.
FUNKTIONEN	= datei	Name der Datei mit Definitionen für die Belegung der Funktionstasten.
	= *	Definitionen für Belegung der Funktionstasten folgen auf das #DEFINIERE-Kommando (ggf. nach den Daten für die Spezifikation VARIABLEN) und sind mit *EOF abgeschlossen.
	= +	* Belegung der Funktionstasten beibehalten.
ZEILEN	= n	Höhe des TUSTEP-Fensters auf n (n = 10 bis 120) Zeilen einstellen.
	= ++	Windows und macOS: Höhe des TUSTEP-Fensters auf die Anzahl Zeilen einstellen, die maximal in einem TUSTEP-Fenster auf dem Bildschirm dargestellt werden können.

	= n;m	Höhe des TUSTEP-Fensters auf n (n = 10 bis 120) Zeilen und Scroll-Bereich des TUSTEP-Fensters auf m (m = n bis 800) Zeilen einstellen.
	= +	* Eingestellte Zeilenzahl beibehalten.
SPALTEN	= n	Breite des TUSTEP-Fensters auf n (n = 80 bis 240) Zeichen einstellen.
	= ++	Windows und macOS: Breite des TUSTEP-Fensters auf die Anzahl Spalten einstellen, die maximal in einem TUSTEP-Fenster auf dem Bildschirm dargestellt werden können.
	= +	* Eingestellte Spaltenzahl beibehalten.
SIGNAL	= ton	Ton, der unter Windows als Fehlersignal bei unerlaubter Tastatureingabe verwendet werden soll.  Welche Angabe für ton erwartet wird, ist beim Kommando #SIGNAL (siehe Seite 222) beschrieben. Hier ist jedoch nur ein Ton, keine Tonfolge zugelassen.
	= +	* Eingestelltes Fehlersignal beibehalten.
	= -	Kein Fehlersignal ausgeben.
FARBEN	= -STD-	Standard-Farben für das Ablaufprotokoll einstellen.
	= xx;...;yy	Angegebene Farben für das Ablaufprotokoll einstellen (siehe »Einstellen der Farben«).
	= +	Eingestellte Farben beibehalten.
DATEINAMEN	= name:bez	Dateiname name definieren und ihm die auf Betriebssystemebene erforderliche Dateibezeichnung bez (Pfad mit Dateiname), zuordnen.
	= name:-	Definition des Dateinamens name löschen.
	= datei	Name der Datei mit Definitionen von Dateinamen.
	= *	Definitionen von Dateinamen folgen auf das #DEFINIERE-Kommando (ggf. nach den Daten für die Spezifikationen VARIABLEN und FUNKTIONEN) und sind mit *EOF abgeschlossen.
	= -	Alle »definierten Dateinamen« löschen.
	= +	* »Definierte Dateinamen« beibehalten.
MAUS	= n	Unter Linux und macOS: Mausrad-Verzögerung auf n (n = 0 bis 9) einstellen.
	= +	* Eingestellte Mausrad-Verzögerung beibehalten.

## Leistung

Mit diesem Kommando

- kann festgelegt werden, in welchen Dateien die Makros (siehe Seite 110) stehen, und in welcher Reihenfolge diese Dateien beim Aufruf eines Makros durchsucht werden.
- können TUSTEP-Variablen definiert werden, auf die in Kommandos (siehe Seite 110) und in Makros (siehe Makrofunktion `FETCH` Seite 422) zugegriffen werden kann.
- kann der Projektname umdefiniert werden, der zur Ergänzung der Dateinamen verwendet werden soll, falls dieser nicht explizit mit dem Dateinamen angegeben wird.
- kann der Name umdefiniert werden, der bei Druckausgaben auf dem Deckblatt verwendet werden soll.
- kann festgelegt werden, in welcher Weise die Zeichen für die Eingabe vom TUSTEP-Fenster und die Ausgabe in das TUSTEP-Fenster umcodiert werden. Beim Initialisieren der TUSTEP-Sitzung wird der Standard-Code eingestellt.
- kann die Belegung der Funktionstasten für die Kommandoebene (nicht für den Editor) definiert werden.
- kann die Höhe des TUSTEP-Fensters eingestellt werden. Für das Kommando `#EDIERE` gilt diese Einstellung jedoch nur, falls mit dem Editor nicht explizit eine andere Höhe eingestellt wurde. Beim Initialisieren der TUSTEP-Sitzung werden 25 Zeilen bzw. die mit der System-Variablen `TUSTEP_ROWS` vorgegebene Zeilenzahl eingestellt.
- kann die Breite des TUSTEP-Fensters für die Kommandos `#EDIERE` und `#SUCHE` sowie für Fenster, die in Makros angezeigt werden, eingestellt werden. Für das Kommando `#EDIERE` gilt diese Einstellung jedoch nur, falls mit dem Editor nicht explizit eine andere Breite eingestellt wurde. Beim Initialisieren der TUSTEP-Sitzung werden 80 Zeichen bzw. die mit der System-Variablen `TUSTEP_COLS` vorgegebene Zeichenzahl eingestellt. Das Ablaufprotokoll wird unabhängig von dieser Definition immer in einem Fenster angezeigt, das 80 Zeichen breit ist.
- kann die Tonhöhe und Tonlänge des Fehlersignals für die Kommandos `#EDIERE` und `#SUCHE` definiert werden, das bei unerlaubter Tastatureingabe (z. B. bei einer nicht definierten Tastenkombination) ausgegeben wird.
- können die Farben für das Ablaufprotokoll im TUSTEP-Fenster definiert werden. Beim Initialisieren der TUSTEP-Sitzung werden die Standard-Farben eingestellt.
- können Dateinamen definiert werden (siehe »Definierte Dateinamen« Seite 29).
- kann unter Linux und macOS eine Mausrad-Verzögerung eingestellt werden.

## Hinweise

Die mit diesem Kommando eingestellte Spaltenzahl gilt nicht für das TUSTEP-Fenster, in dem das Ablaufprotokoll angezeigt wird, sondern nur für die TUSTEP-Fenster

der Kommandos #EDIERE und #SUCHE sowie für die Makrofenster (siehe Seite 591). Das Ablaufprotokoll wird immer 80 Spalten breit angezeigt.

Welche Einstellungen/Definitionen jeweils aktuell sind, kann mit dem Kommando #INFORMIERE (siehe Seite 167) abgefragt werden.

Beim Einrichten bzw. Anmelden einer Datei mit einem definierten Dateinamen muss als Träger ein Minuszeichen angegeben werden.

## Definieren von TUSTEP-Variablen

Der Name einer TUSTEP-Variablen kann aus 1 bis 12 Zeichen (Buchstaben, Ziffern und »\_«) bestehen, muss mit einem Buchstaben beginnen und darf nicht mit »\_« enden.

Eine TUSTEP-Variable kann mit der folgenden Anweisung definiert und mit einem Wert belegt werden bzw. mit einem neuen Wert belegt werden, falls sie schon definiert ist:

Variablenname = Variablenwert

Dabei ist der Variablenwert eine beliebige Zeichenfolge. Führende und abschließende Leerzeichen bleiben unberücksichtigt.

Jede Anweisung muss in einer eigenen Zeile stehen. Zeilen, die mit »=« beginnen, werden als Kommentar übergangen.

Soll nur eine einzige Variable definiert werden, kann die Anweisung direkt als Spezifikationswert zur Spezifikation VARIABLEN angegeben werden. Es gelten dann jedoch folgende Besonderheiten:

- Statt des Gleichheitszeichens zwischen Variablenname und Variablenwert muss aus syntaktischen Gründen ein Doppelpunkt verwendet werden.
- In der als Wert angegebenen Zeichenfolge darf aus syntaktischen Gründen keines der folgenden Zeichen vorkommen: Nummernzeichen, Komma, Gleichheitszeichen, Apostroph und Anführungszeichen.
- Alle evtl. im Variablenwert enthaltenen (nicht nur die führenden und abschließenden) Leerzeichen werden eliminiert.

## Einschränkung zum Namen

Ein mit der Spezifikation NAME definierte Name wird bei Druckausgaben auf dem Deckblatt ausgedruckt, wenn dieses von TUSTEP erstellt wird. Wenn das Deckblatt vom Betriebssystem erstellt wird, kann dieser Name auf manchen Rechnern nicht berücksichtigt werden.

## Code für Zeichendarstellung

Der in TUSTEP mit dem Kommando #DEFINIERE eingestellte Code darf nur solche Zeichen enthalten, die auch auf dem Bildschirm korrekt dargestellt werden können. Welche Zeichen dargestellt werden können, hängt von dem Code ab, der unter Win-



dows vom Betriebssystem vorgegeben bzw. unter Linux und macOS im Terminal-Programm eingestellt ist.

Ob dieser Code mit dem in TUSTEP eingestellten Code harmoniert, kann in TUSTEP mit dem Kommando `#LISTE, CODE=+` (siehe Seite 173) überprüft werden. Dabei wird die zuletzt definierte Code-Tabelle ins Ablaufprotokoll ausgegeben. Entsprechen die Zeichen in der letzten Spalte dieser Tabelle nicht der in der zweitletzten Spalte angegebenen TUSTEP-Codierung oder sind Zeichen in der letzten Spalte unsichtbar, so darf mit dieser Code-Einstellung nicht gearbeitet werden, da sonst diese Zeichen beim Edieren verlorengehen.

Über die Tastatur können nur solche Zeichen direkt eingegeben werden, die auch in dem in TUSTEP definierten Code enthalten sind; für alle anderen Zeichen müssen die im Kapitel »Zeichenvorrat« (siehe Seite 759) angegebenen TUSTEP-Codierungen verwendet werden.

#### **Linux und macOS:**

Unter Linux und macOS ist in TUSTEP der Code »ISO-8859-1« voreingestellt. Aus historischen Gründen gibt es auch noch den Code »LINUX«; er enthält aber nur einen Teil der im Code »ISO-8859-1« enthalten Zeichen. Ein weiterer möglicher Code wäre »ASCII«, der nur die ASCII-Zeichen enthält. Andere mit dem Kommando `#DEFINIERE` (siehe Seite 132) einstellbare Codes eignen sich nur für Windows.

Da die Code-Einstellung nicht automatisch an das Terminal-Programm weitergegeben werden kann, ist es erforderlich, im jeweiligen Terminal-Programm (Konsole bzw. iTerm) den Code »ISO-8859-1« ebenfalls einzustellen. Dies ist in der Installationsanleitung unter »Konfiguration von TUSTEP unter Linux« bzw. »... unter macOS« beschrieben. Kann der Code »ISO-8859-1« im Terminal-Programm nicht eingestellt werden, so muss in TUSTEP der Code »ASCII« eingestellt werden.

#### **Windows:**

Unter Windows ist in TUSTEP der Code, der der vom Betriebssystem vorgegebenen Code Page entspricht, voreingestellt. Dies ist im deutschsprachigen Raum meist die Code Page 850. Können mit dem voreingestellten Code nicht alle Zeichen korrekt dargestellt werden, so eignet sich meist der Code »IBMPC«, der außer den ASCII-Zeichen die Umlaute und »ß« enthält. Werden die genannten Zeichen nicht korrekt dargestellt, muss der Code »ASCII« eingestellt werden.

Unter Windows gibt es alternativ die Möglichkeit, aus einer vorgegebenen Liste die Zeichen (insbesondere Akzentbuchstaben) auszuwählen, die in TUSTEP zusätzlich zu den ASCII-Zeichen auf dem Bildschirm dargestellt werden können. Diese Auswahl kann mit einem frei gewählten Namen abgespeichert und dann unter Angabe dieses Namens mit dem Kommando `#DEFINIERE` eingestellt werden. Eine Auswahl mit dem Namen `WINDOWS` ist von TUSTEP schon vorgegeben und kann verwendet werden.

Wichtig: Bevor eine solche Auswahl erstellt bzw. eingestellt wird, muss unbedingt geprüft werden, ob ein geeigneter Unicode-Font eingestellt ist. Der Font »Rasterschriftart«, der meist vom Betriebssystem voreingestellt ist, eignet sich nicht! Im Zweifelsfall kann der Font »Courier New« oder der Font »Consolas« eingestellt wer-

den. Wie ein Font unter Windows eingestellt werden kann, ist in der Installationsanleitung, die mit dem Kommando #\*ZEBE, CONFIG angezeigt wird, im Kapitel »Tipps zu TUSTEP unter Windows« im Abschnitt »Schrift« beschrieben.

Um eine eigene Auswahl zu erstellen, muss das Makro \*DECO aufgerufen werden. Nach dem Aufruf dieses Makros wird folgende Eingabemaske angezeigt:



In der linken Spalte sind die Namen der schon definierten Auswahlen angegeben; in der mittleren Spalte steht jeweils der dazugehörige Kommentar. Die rechte Spalte enthält die Liste mit den Zeichen, die ausgewählt werden können. Für jedes Zeichen ist in eckigen Klammern der entsprechende Unicode als Hexadezimal-Code, das Zeichen selbst und die ASCII-Zeichenfolge, in der das Zeichen in TUSTEP eingegeben werden kann, angegeben. Die ausgewählten Zeichen sind hervorgehoben.

Um eine in der linken Spalte angegebene Auswahl in der rechten Spalte anzuzeigen, kann entweder der Name angeklickt werden oder der Name oben in das Eingabefeld eingetragen und dann unten die Schaltfläche »HOLEN« aktiviert werden. Eine Schaltfläche kann aktiviert werden, indem sie mit der Maus angeklickt wird oder indem der Cursor mit der Tabulator-Taste auf die Schaltfläche positioniert und dann die Return-Taste gedrückt wird.

Ein Zeichen kann ausgewählt werden, indem ein nicht ausgewähltes Zeichen mit der Maus angeklickt wird, oder indem der Cursor (z. B. mit den Pfeiltasten) in die entsprechende Zeile positioniert und die Return-Taste (nicht die Enter-Taste) gedrückt wird. Es dürfen nur solche Zeichen ausgewählt werden, die korrekt (nicht mit einem leeren Rechteck) dargestellt werden. Insgesamt können bis zu 128 Zeichen ausgewählt werden. Wieviel Zeichen jeweils noch ausgewählt werden können wird oben angezeigt.

Ein Zeichen kann abgewählt werden, indem ein ausgewähltes Zeichen mit der Maus angeklickt wird, oder indem der Cursor (z. B. mit den Pfeiltasten) in die entsprechende Zeile positioniert und die Delete-Taste gedrückt wird.

Um eine neue Auswahl abzuspeichern, muss oben der Name für die Auswahl eingetragen und dann unten die Schaltfläche »RETTEN« aktiviert werden. Die Auswahl wird in der Datei \*TUSTEP.CDS abgespeichert.

Nach dem Aktivieren der Schaltfläche »LÖSCHEN« wird nachgefragt, ob die in der rechten Spalte getroffene Auswahl aufgehoben werden soll (danach ist kein Zeichen mehr ausgewählt), oder ob die abgespeicherte Auswahl mit dem oben angegebenen Namen gelöscht werden soll.

Nach dem Aktivieren der Schaltflächen »EINSTELLEN« wird das Makro beendet und mit dem Kommando #DEFINIERE automatisch die Auswahl eingestellt, die zuletzt angezeigt wurde.

Soll ein bestimmter Code beim Initialisieren einer TUSTEP-Sitzung immer automatisch eingestellt werden, kann das entsprechende Kommando in die INI-Datei (siehe Seite 89) eingetragen werden.

## Belegen der Funktionstasten

Die im Folgenden beschriebene Belegung der Funktionstasten gilt nur für die Kommandoebene (für Eingabe nach »Gib Kommando«). Zur Verwendung im Editor (für Eingabe nach »Gib Anweisung«) müssen die Funktionstasten eigens belegt werden (siehe Seite 287).

Die Funktionstasten wirken nur dann alle wie nachfolgend beschrieben, wenn der Protokoll-Modus auf PORTIONIERT oder FREILAUFEND eingestellt ist. Wurde er mit dem Kommando #PROTOKOLL (siehe Seite 209) auf SYSTEM gestellt, so haben sie die vom jeweiligen Betriebssystem vorgegebene Wirkung.

Bei der Eingabe auf Kommandoebene werden die jeweils letzten 80 eingegebenen Zeilen gemerkt. Um diese Zeilen wieder ins TUSTEP-Fenster auszugeben und sie dann ggf. wieder als Eingabe zu verwenden, können Funktionstasten verwendet werden.

Die Belegung der Funktionstasten ist noch nicht beliebig. Es sind z. Z. nur acht verschiedene Belegungen möglich:

$F_n = \text{EXTEND}$  (Voreinstellung für F2)

bewirkt, dass mit der Funktionstaste  $n$  ein Dateiname, dessen Anfang unmittelbar links vom Cursor steht, vervollständigt wird. Dabei werden nur Namen von angemeldeten Dateien (einschließlich Scratch-Dateien) berücksichtigt. Ist der Anfang des Dateinamens nicht eindeutig, so wird beim ersten Drücken dieser Funktionstaste der Name bis zu der Stelle ergänzt, an der er mehrdeutig wird; bei wiederholtem Drücken dieser Funktionstaste wird jeweils der nächste nach alphabetischer Reihenfolge in Frage kommende Dateiname angezeigt. Steht links vom Cursor ein Leerzeichen, Komma, Gleichheitszeichen oder Apostroph, wird »-STD-« ergänzt.

`Fn=CANCEL` (Voreinstellung für F4)

bewirkt, dass mit der Funktionstaste `n` die Dateneingabe beendet bzw. die TUSTEP-Sitzung unterbrochen werden kann; die Funktionstaste `n` hat damit die gleiche Wirkung wie die Eingabe von `*EOF`.

`Fn=RECALL` (Voreinstellung für F5)

bewirkt, dass mit der Funktionstaste `n` die gemerkten Eingaben in einem Fenster angezeigt werden; davon kann ggf. eine ausgewählt werden, um sie verändert oder unverändert nochmals einzugeben (siehe Kommando-Manager Seite 117).

`Fn=CALL_D` (Voreinstellung für F6)

bewirkt, dass mit der Funktionstaste `n` das Makro `*D` aufgerufen wird, falls eine Kommandoingabe erwartet wird; andernfalls wird die Dateneingabe beendet.

`Fn=JMP_UP` (Voreinstellung für F7)

bewirkt, dass mit der Funktionstaste `n` bestimmte Eingaben gesucht und ausgegeben werden können. Gesucht wird jeweils eine Eingabe, deren Anfang mit den links vom Cursor stehenden Zeichen übereinstimmt. Die Suche erfolgt rückwärts und beginnt bei der letzten Eingabe bzw. vor der zuletzt ausgegebenen Eingabe.

`Fn=JMP_DN` (Voreinstellung für F8)

bewirkt, dass mit der Funktionstaste `n` bestimmte Eingaben gesucht und ausgegeben werden können. Gesucht wird jeweils eine Eingabe, deren Anfang mit den links vom Cursor stehenden Zeichen übereinstimmt. Die Suche erfolgt vorwärts und beginnt nach der zuletzt ausgegebenen Eingabe.

`Fn=CUR_UP` (Voreinstellung für F9)

bewirkt, dass mit der Funktionstaste `n` in den gemerkten Eingaben zurückgeblättert werden kann. Wird diese Funktionstaste nach einer Eingabe gedrückt, wird die letzte Eingabe ausgegeben.

`Fn=CUR_DN` (Voreinstellung für F10)

bewirkt, dass mit der Funktionstaste `n` in den gemerkten Eingaben vorgeblättert wird. Wird diese Funktionstaste nach einer Eingabe gedrückt, wird die Eingabe ausgegeben, die auf die zuletzt mit einer Funktionstaste ausgegebene folgt.

## Einstellen der Farben

Für das Ablaufprotokoll im TUSTEP-Fenster können für die Anzeige der Eingabe und für neun verschiedenen Ausgaben Farben eingestellt werden:

- Eingabeaufforderung (z. B. Gib Kommando >)
- Kommandoingabe (z. B. #anmelde,datei)
- Kommandoausgabe (z. B. #anmelde,datei)
- Parameterausgabe (z. B. AA /\$/)
- Normale Meldungen (z. B. \*\*\*\* Angemeldet: ...)
- Warnungen (z. B. ##### Datei ... enthält keine Daten)
- Fehlermeldungen (z. B. ##### Datei ... nicht gefunden)

- Fortschrittsanzeige (z. B. ... 100000 von 1234567 Sätzen = 8 %)
- Andere Ausgaben (z. B. Daten aus Dateien)

Die neun Farben sind in Hexadezimal-Codes anzugeben und durch Strichpunkt zu trennen; die möglichen Codes können der Tabelle entnommen werden, die im Editor durch die Tastenkombination `Ctrl+F` bzw. `Strg+F` angezeigt wird.

Die Anzeige erfolgt nur dann in den so definierten Farben, wenn mit dem Kommando `#PROTOKOLL` (siehe Seite 209) der Protokoll-Modus auf `PORTIONIERT` oder `FREILAUFEND` eingestellt worden ist.

## Definieren von Dateinamen

Ein Dateiname kann mit der folgenden Anweisung definiert und mit einer Datei- bezeichnung belegt werden bzw. mit einer neuen Datei- bezeichnung belegt werden, falls er schon definiert ist:

Dateiname = Datei- bezeichnung

Dabei ist die Datei- bezeichnung eine Pfadangabe einschließlich des Dateinamens, wie sie für das Betriebssystem erforderlich ist. Führende und abschließende Leerzeichen bleiben unberücksichtigt.

Jede Anweisung muss in einer eigenen Zeile stehen. Zeilen, die mit `»=«` beginnen, werden als Kommentar übergangen.

Soll nur ein einziger Dateiname definiert werden, kann die Anweisung direkt als Spezifikationswert zur Spezifikation `DATEINAMEN` angegeben werden. Es gelten dann jedoch folgende Besonderheiten:

- Statt des Gleichheitszeichens zwischen Dateiname und Datei- bezeichnung muss aus syntaktischen Gründen ein Doppelpunkt verwendet werden.
- In der als Datei- bezeichnung angegebenen Zeichenfolge darf aus syntaktischen Gründen keines der folgenden Zeichen vorkommen: Nummernzeichen, Komma, Gleichheitszeichen, Apostroph und Anführungszeichen.
- Alle evtl. in der Datei- bezeichnung enthaltenen (nicht nur die führenden und abschließenden) Leerzeichen werden eliminiert.

## Einstellen der Mausrad-Verzögerung

Auf einigen Rechnern wird beim Scrollen mit dem Mausrad und insbesondere beim Scrollen mit dem Touchpad der Text schneller als gewollt verschoben. Damit der Text langsamer verschoben wird, kann eine Verzögerung eingestellt werden. Normalerweise wird für jeden Impuls der Text um eine Zeile verschoben. Wird für die Verzögerung der Wert `n` eingestellt, so werden nach jedem Impuls, der den Text verschiebt, `n` Impulse ignoriert.

# Druckausgabe

#DRUCKE

## Kommando

#DRUCKE

PROTOKOLL	= datei	Name der Datei mit den zu druckenden Daten.
	= -STD-	* Die zu druckenden Daten stehen in der Standard-Protokoll-Datei.
TYP	= ...	Typ des Druckers, auf dem ausgedruckt werden soll. Mit dem Kommando #LISTE, DRUCKERTYPEN (siehe Seite 173) können die möglichen Druckertypen aufgelistet werden.
GERAET	= -STD-	* Die System-Variable TUSTEP_PRN enthält den Namen des Druckers, auf dem ausgedruckt werden soll; falls diese System-Variable nicht definiert ist, wird der Druckername vom Betriebssystem erfragt.
	= name	Name einer System-Variablen, die den Namen des Druckers enthält, auf dem ausgedruckt werden soll. Der Name des Druckers kann eventuell direkt angegeben werden; siehe dazu unter »Druckernamen«.
	= +	Ausgabe ins Ablaufprotokoll; bei TYP=WIN-xx unter Windows, TYP=MAC-xx unter macOS und TYP=PS-xx unter Linux Ausgabe in ein eigenes Fenster.
	= -	Keine Ausgabe auf einem Drucker. Falls zur Spezifikation DATEI keine Datei angegeben ist, werden nur die Daten der PROTOKOLL-Datei syntaktisch überprüft.
ANZAHL	= 1	* Daten 1-mal ausdrucken.
	= n	Daten n-mal ausdrucken.
	= n*m	Daten n*m-mal ausdrucken.
VORSPANN	= -	* Kein Vorspann.
	= *	Vorspann folgt auf das #DRUCKE-Kommando und ist mit *EOF abgeschlossen.
	= datei	Name der Datei mit dem Vorspann.

SEITEN	= -	* Datei vollständig ausdrucken.
	= n / n-m	Nur die Seite n bzw. nur die Seiten n bis m ausdrucken. Es können auch mehrere Seiten/Bereiche angegeben werden; die Angaben müssen durch Apostroph getrennt sein.
	= HEFT	Die einzelnen Seiten in der Reihenfolge ausdrucken, wie sie für ein Heft erforderlich ist.
	= HEFT-VS	Wie HEFT, jedoch nur die Vorderseiten ausdrucken.
	= HEFT-RS	Wie HEFT, jedoch nur Rückseiten ausdrucken.
UMSCHLAG	= -STD-	* Ob von TUSTEP bei Druckausgaben ein Umschlag (Deck- und Endblatt mit Namen) ergänzt wird, ist vom jeweiligen Rechner und vom Druckertyp abhängig.
	= -	Von TUSTEP keinen Umschlag ergänzen. Diese Angabe wird ignoriert, falls dies vom Systemverwalter für bestimmte Druckertypen nicht zugelassen ist.
	= +	Von TUSTEP einen Umschlag ergänzen.
DATEI	= -	* Druckausgabe auf dem zur Spezifikation GERAET angegebenen Drucker ausgeben.
	= datei	Name der Fremd-Datei, in die die Codes zur Steuerung des Druckers ausgegeben werden sollen. Es können bis zu zehn Dateinamen angegeben werden; sie müssen durch Apostroph getrennt sein.
LOESCHEN	= -	* Daten in den zur Spezifikation DATEI angegebenen Dateien nicht löschen.
	= +	Daten in den zur Spezifikation DATEI angegebenen Dateien zuerst löschen.
OPTIONEN	= ...	* Angaben zur Modifikation der erzeugten Steuer-codes für den Drucker.
ZUSATZ	= ...	Name einer System-Variablen, die zusätzliche Angaben für das Betriebssystem enthält (vgl. System-Variablen TUSTEP_LPR Seite 76). Welche Angaben sinnvoll oder im Einzelfall notwendig sind, ist beim Systemverwalter zu erfragen.
PORTION	= -STD-	* Ausdruck bei Zeilendruckern in Portionen zu 500 Seiten, bei Matrix- und Laserdruckern in Portionen zu 200 Seiten unterteilen.
	= n	Ausdruck in Portionen zu n Seiten unterteilen.
TITEL	= -	* Normalfall

= titel            Unter Windows: Bei TYP=WIN-xx und GERAET=+ angegebenen Titel in die Kopfzeile des Fensters einfügen, in dem die Daten angezeigt werden. Falls schon Fenster mit dem gleichen Titel vorhanden sind, diese schließen.

## Leistung

Dieses Kommando dient zum Anzeigen und Ausdrucken von Dateien, deren Daten schon zum Drucken aufbereitet sind (Protokoll-Dateien). Dies sind in der Regel Dateien, die von einem vorangehenden TUSTEP-Programm als PROTOKOLL-Datei beschrieben wurden. Soll eine Datei ausgedruckt werden, die keine Protokoll-Datei ist, so müssen die Daten zuvor entsprechend aufbereitet werden. Dies kann mit dem Kommando #FORMATIERE geschehen, wenn die Datei Formatieranweisungen dafür enthält, die interpretiert werden sollen, andernfalls mit dem kommando #DVORBEREITE.

Kann auf den Drucker, auf dem die Datei ausgedruckt werden soll, nicht direkt ausgegeben werden (z. B. weil er an einen anderen PC angeschlossen ist), so kann zur Spezifikation DATEI eine permanente Fremd-Datei (TYP=FDF) angegeben werden, in die die Codes ausgegeben werden, die für die Steuerung eines Druckers vom angegebenen Typ notwendig sind. Diese Fremd-Datei kann dann mit Kommandos, die im jeweiligen Betriebssystem zur Verfügung stehen, zum gewünschten Drucker übertragen werden. Werden zur Spezifikation DATEI mehrere Dateien angegeben, so werden jeweils so viele Seiten in eine Datei geschrieben, wie zur Spezifikation PORTION angegeben wird, und dann in der nächsten Datei weitergeschrieben.

## Hinweis

Weitere Möglichkeiten zum Anzeigen und Ausdrucken von Dateien sind im Kapitel »Drucken von Dateien« auf Seite 38 beschrieben.

## Druckernamen

Wenn der Name des Druckers, auf dem ausgedruckt werden soll, aus 1 bis 12 Zeichen (Buchstaben, Ziffern und »\_«) besteht, mit einem Buchstaben beginnt, nicht mit »\_« endet und nicht mit dem Namen einer System-Variablen übereinstimmt, kann er direkt zur Spezifikation GERAET angegeben werden. Erfüllt der Name diese Bedingungen nicht, muss eine System-Variable definiert werden, die den Druckernamen enthält, und diese zur Spezifikation GERAET angegeben werden. Die Namen der möglichen Drucker sind beim Systemverwalter zu erfragen.

## Seiten-Auswahl

Sollen nur bestimmte Seiten ausgedruckt werden, so können diese mit der Spezifikation SEITEN ausgewählt werden. Die Auswahl erfolgt anhand der Satznummern in der PROTOKOLL-Datei und nicht anhand der evtl. jeweils in der Seitenüberschrift (im Kopftext) enthaltenen Seitenzahl. Sollen die Seiten anhand der in der Seiten-



überschrift enthaltenen Seitenzahlen ausgewählt werden, kann diese Auswahl mit dem Kommando #DVORBEREITE erfolgen.

## Mehrfach-Druck

Wenn von einer Datei (durch eine entsprechende Angabe zur Spezifikation ANZAHL) mehrere Exemplare ausgedruckt werden, so wird bei manchen Betriebssystemen jeweils auch ein Umschlag mit ausgedruckt, der u. U. aus mehreren Seiten besteht. Um vor allem beim mehrfachen Drucken kürzerer Dateien Papier zu sparen, kann zur Spezifikation ANZAHL statt  $n$  auch  $n*m$  angegeben werden. In diesem Fall wird die zur Spezifikation PROTOKOLL angegebene Datei  $m$ -mal in eine interne Hilfsdatei kopiert, die dann  $n$ -mal (d. h. mit nur  $n$  Umschlägen) ausgedruckt wird.

## Heftchen-Druck

Sollen auf Vorder- und Rückseite eines Blattes jeweils zwei Seiten im Querformat in der Reihenfolge gedruckt werden, dass der Papierstapel anschließend nur noch zu einem Heftchen gefaltet zu werden braucht, so kann dies durch Angabe eines entsprechenden Druckertyps zur Spezifikation TYP (z. B. PS-Q2, damit zwei Seiten nebeneinander im Querformat gedruckt werden) und der Angabe HEFT zur Spezifikation SEITEN (damit die Reihenfolge stimmt) erreicht werden. Kann der Drucker beidseitig drucken, muss zusätzlich noch die Option D\_LANG oder D\_KURZ (s. u. »Optionen«) angegeben werden; andernfalls müssen mit der Spezifikation SEITEN=HEFT-VS erst die Vorderseiten bedruckt werden und anschließend mit SEITEN=HEFT-RS die Rückseiten.

## Optionen

Zum Drucken auf HP-kompatiblen Druckern und PostScript-Druckern, die das Papier beidseitig bedrucken können, sind folgende Optionen möglich:

- SMPLX Nur Vorderseite bedrucken
- D\_LANG Vorder- und Rückseite so bedrucken, dass über die lange Papierkante geblättert werden kann (meist für Hochformat)
- D\_KURZ Vorder- und Rückseite so bedrucken, dass über die kurze Papierkante geblättert werden kann (meist für Querformat)

Zum Drucken auf HP-kompatiblen Druckern, die über zwei Papierschächte verfügen, sind folgende Optionen möglich:

- OBEN Papier aus dem oberen Schacht
- UNTEN Papier aus dem unteren Schacht

Zum Drucken auf EPSON-kompatiblen Druckern sind folgende Optionen möglich:

- BD bidirektionaler Druck
- UD unidirektionaler Druck (Voreinstellung)
- H11 Druck auf 11 Zoll hohes Papier
- H12 Druck auf 12 Zoll hohes Papier
- LQ Schönschreibmodus (Voreinstellung)
- DRAFT Schnellschreibmodus

## Beispiele

Ausdrucken der Datei `arbeit` (ohne Interpretation der eventuell in den Daten enthaltenen Formatieranweisungen und Steuerzeichen) auf einem HP-LaserJet, der den vom Betriebssystem vorgegebenen Druckernamen `pr1` hat:

```
#DV, arbeit, HPLJ, +  
#DR, , HPLJ, pr1
```

Ausdrucken der Datei `arbeit` (mit Interpretation der in den Daten enthaltenen Formatieranweisungen und Steuerzeichen) auf einem HP-LaserJet, der den vom Betriebssystem vorgegebenen Druckernamen `pr2` hat:

```
#FO, arbeit, , HPLJ, +  
#DR, , HPLJ, pr2
```

Unter Windows kann vor dem Ausdrucken auf dem Drucker das Ergebnis zur Kontrolle auf dem Bildschirm angezeigt werden:

```
#FO, arbeit, , HPLJ, +  
#DR, , WIN-10, +
```

Zum anschließenden Ausdrucken auf dem Drucker genügt das Kommando `#DRUCKE`; das Kommando `#FORMATIERE` braucht (falls die Daten in der Datei `arbeit` nicht geändert wurden) nicht wiederholt zu werden:

```
#DR, , HPLJ, pr2
```

Ausdrucken der Datei `arbeit` (mit Interpretation der in den Daten enthaltenen Formatieranweisungen und Steuerzeichen) auf einem PostScript-Drucker, der das Papier beidseitig bedrucken kann und den vom Betriebssystem vorgegebenen Druckernamen `ps3` hat. Der Ausdruck der einzelnen Seiten soll in der Reihenfolge erfolgen, dass diese nach Falten des Papierstapels zu einem Heftchen in der richtigen Reihenfolge stehen.

```
#FO, arbeit, , PS-Q2, +  
#DR, , PS-Q2, ps3, SEITEN=HEFT, OPTIONEN=D_KURZ
```

# Analysieren von Dateien

#DUMPE

## Kommando

#DUMPE

DATEI	=	datei	Name der zu analysierenden Datei.
PROTOKOLL	=	+	* Ergebnis der ausgeführten Anweisungen ins Ablaufprotokoll ausgeben.
	=	datei	Name der Datei, in der das Ergebnis der ausgeführten Anweisungen protokolliert wird.

## Leistung

Mit diesem Kommando kann der Inhalt von Dateien analysiert (z. B. die Struktur einer Fremd-Datei herausgefunden) und verändert werden. Letzteres sollte jedoch nur bei genauer Kenntnis der jeweiligen Dateistruktur geschehen. Nach dem Aufruf werden Anweisungen erwartet. Eine Liste der möglichen Anweisungen wird nach der Eingabe eines Fragezeichens ausgegeben. Beendet wird das Programm durch die Eingabe von »\*EOF« an Stelle einer Anweisung.

# Daten zum Drucken vorbereiten

#DVORBEREITE

## Kommando

#DVORBEREITE

Für dieses Kommando gibt es eine eigene Beschreibung.

## Leistung

Mit diesem Programm können Daten zum Drucken vorbereitet werden. In den Daten enthaltene Steuerzeichen werden dabei nicht interpretiert, sondern wie normale Zeichen behandelt. Zur Gestaltung des Ausdrucks sind u. a. folgende Angaben möglich: Anzahl der Spalten, Seiten- und Spaltenüberschrift, Art der Nummerierung, Zeilenabstand.

Außerdem können aus einer zur Spezifikation `QUELLE` angegebenen Protokoll-Datei bestimmte Seiten ausgewählt und in die zur Spezifikation `PROTOKOLL` angegebene Datei kopiert werden.

## Hinweise

Die Daten in der `QUELL`-Datei bleiben unverändert. Das Ergebnis des Programms (die zum Drucken vorbereiteten Daten) wird in die `PROTOKOLL`-Datei ausgegeben. Diese kann anschließend mit dem Kommando `#DRUCKE` ausgedruckt (auf einem Drucker ausgegeben) werden.

Sollen die in den Daten enthaltenen Steuerzeichen interpretiert werden, so müssen die Daten (statt mit dem Kommando `#DVORBEREITE`) mit dem Kommando `#FORMATIERE` bzw. `#SATZ` zum Drucken aufbereitet werden.

# Edieren von Daten/Dateien

#EDIERE

## Kommando

#EDIERE

DATEI	= datei	Name der Datei mit den zu edierenden Daten.
	= -STD-	Daten in der Standard-Editor-Datei edieren.
	= -	Keine Daten edieren.
MODUS	= T	Nummerierung der Sätze im Textmodus.
	= P	Nummerierung der Sätze im Programmmodus.
	= -STD-	* Modus automatisch einstellen.
	= ...	(Weitere Modi sind unten beschrieben)
	= !	Wie -STD-, jedoch nach dem Start des Editors die zuletzt gültigen Definitionen in die zur Spezifikation DEFINITIONEN angegebene Datei schreiben; alle in dieser Datei bereits vorhandenen Daten zuvor löschen.
DEFINITIONEN	= -	* Keine zusätzlichen Definitionen; die zuletzt gültigen beibehalten.
	= datei	Name der Datei mit den zusätzlich zu übernehmenden Definitionen bzw. Name der Datei für die zuletzt gültigen Definitionen, falls MODUS=! angegeben ist.
	= *	Die zusätzlich zu übernehmenden Definitionen folgen auf das #EDIERE-Kommando und sind mit *EOF abgeschlossen.
	= -STD-	Alle Definitionen löschen und die voreingestellten Definitionen wieder übernehmen.
	= +	Alle Definitionen löschen und die voreingestellten sowie die im Segment EDIT der INI-Datei (siehe Seite 89) stehenden Definitionen wieder übernehmen.
	= !	Editor neu initialisieren.
WARTEN	= +	Beim Starten des Editors vor dem Anzeigen des Editorfensters auf eine Bestätigung (mit der Return-Taste) warten.
	= -	* Beim Starten des Editors vor dem Anzeigen des Editorfensters nicht auf eine Bestätigung warten.

MAKRO	= name	Name des Editormakros, das nach dem Start des Editors automatisch aufgerufen und ausgeführt werden soll.
	= -	* Nach dem Start des Editors kein Makro automatisch aufrufen.
ZEILEN	= n	Höhe des Editor-Fensters auf n (n = 10 bis 120) Zeilen einstellen.
	= ++	Windows und macOS: Höhe des Editor-Fensters auf die Anzahl Zeilen einstellen, die maximal in einem Editor-Fenster auf dem Bildschirm dargestellt werden können.
	= n;no;nu	Höhe des Editor-Fensters auf n (n = 10 bis 120 oder ++ oder +) Zeilen einstellen. Falls das Textfeld horizontal unterteilt wird, oberes Textfeld auf no (no = 3 bis 110) und unteres auf nu (nu = 3 bis 110) Zeilen einstellen. Oberes und unteres Textfeld dürfen zusammen maximal 114 Zeilen hoch sein.
	= +	* Eingestellte Zeilenzahl beibehalten.
SPALTEN	= n	Breite des Editor-Fensters auf n (n = 80 bis 240) Zeichen einstellen.
	= ++	Windows und macOS: Breite des Editor-Fensters auf die Anzahl Spalten einstellen, die maximal in einem Editor-Fenster auf dem Bildschirm dargestellt werden können.
	= n;n1;nr	Breite des Editor-Fensters auf n (n = 80 bis 240 oder ++ oder +) Spalten einstellen. Falls das Textfeld vertikal unterteilt wird, linkes Textfeld auf n1 (n1 = 40 bis 198) und rechtes auf nr (nr = 40 bis 198) Spalten einstellen. Linkes und rechtes Textfeld dürfen zusammen maximal 238 Spalten breit sein.
	= +	* Eingestellte Spaltenzahl beibehalten.

## Leistung

Mit diesem Kommando kann der Editor gestartet werden. Mit ihm können Texte und Programme am Bildschirm eingegeben und geändert werden.

Wird der Editor ohne Spezifikationsangaben aufgerufen, so wird die zuletzt mit dem Editor edierte Datei im dabei verwendeten Modus ediert; wird der Editor beim ersten Mal nach dem Initialisieren der TUSTEP-Sitzung ohne Spezifikationsangaben aufgerufen, so wird die Standard-Editor-Datei im Modus P ediert.

Falls die zu edierende Datei nicht angemeldet ist, aber (auf dem Träger TUSTEP\_DSK) existiert, wird nachgefragt, ob Sie angemeldet werden soll. Für diese Anfrage gibt es drei zulässige Antworten:

- 1) L Die Datei zum Lesen anmelden.
- 2) S Die Datei zum Schreiben anmelden.
- 3) N Die Datei nicht anmelden.

Falls die zu edierende Datei nicht angemeldet ist und (auf dem Träger TUSTEP\_DSK) auch nicht existiert, wird nachgefragt, ob Sie eingerichtet werden soll. Für diese Anfrage gibt es drei zulässige Antworten:

- 1) P Die Datei als permanente Datei einrichten.
- 2) T Die Datei als temporäre Datei einrichten.
- 3) N Die Datei nicht einrichten

## Hinweise

Der Editor kann auch mit dem Kommando #\*E (siehe Seite 257) aufgerufen werden. Außerhalb von TUSTEP kann der Editor über den Dateimanager gestartet werden. Dies ist für Windows auf Seite 103, für Linux auf Seite 104 und für macOS auf Seite 105 beschrieben.

## Modi

Neben den oben angegebenen Modi gibt es noch weitere, deren vollständige Bezeichnung aus drei Teilwerten besteht, die ohne Trennzeichen hintereinander angegeben werden (z. B. +P-):

- Der erste Teilwert kann »+« oder »-« sein und gibt an, ob die Sätze (z. B. nach einer Zeige-Anweisung) mit (+) oder ohne (-) Satznummern im Editorfenster angezeigt werden sollen. Eine Ausnahme bilden die »Such-Anweisungen für strukturierte Daten«; hier werden die Daten unabhängig von dieser Einstellung immer ohne Satznummer ins Editorfenster angezeigt. Zu beachten ist, dass Sätze, die ohne Satznummern angezeigt werden, nicht korrigiert werden können.
- Der zweite Teilwert kann »T« oder »P« sein und gibt an, ob die Sätze im Textmodus (mit Seitennummer) oder im Programmmodus (ohne Seitennummer) nummeriert bzw. angezeigt werden sollen.
- Der dritte Teilwert kann »+« oder »-« sein und gibt an, ob Akzentbuchstaben und mit dem Steuerzeichen # codierte Zeichen im Editorfenster als solche dargestellt (+) oder in der Eingabe-Codierung (-) dargestellt werden sollen, ob also z. B. ein e mit Gravis als è oder als %\e dargestellt werden soll. Diese Einstellung ist jedoch nur von Bedeutung, wenn mit dem Kommando #DEFINIERE (siehe Seite 132) auch ein Code eingestellt wurde, in dem solche Zeichen unterstützt werden.

Wird zur Spezifikation MODUS der Wert -STD- (Voreinstellung) angegeben, so wird der Modus automatisch nach folgenden Kriterien eingestellt:

- Wenn die Datei schon mit dem Editor bearbeitet wurde, der Editor dabei Schreib-erlaubnis hatte und die Datei seither nicht mehr verändert wurde, wird der Modus eingestellt, der beim letzten Mal verwendet wurde.

- Wenn die Sätze der Datei im Textmodus nummeriert sind (dies wird angenommen, wenn die Seitennummer des letzten Satzes nicht Null ist), wird Modus T eingestellt.
- In allen anderen Fällen wird Modus P eingestellt.

## Einschränkungen

In der zu edierenden Datei müssen die Satznummern alle aufsteigend und voneinander verschieden sein, da die einzelnen Sätze im Editor über die Satznummern angesprochen werden.

Soll ein Satz geändert werden, so darf dieser nicht länger als 8000 Zeichen sein bzw. auf Grund der Änderung länger als 8000 Zeichen werden.

Soll ein Satz angezeigt werden, der wegen seiner Länge nicht ins Editorfenster passt, wird nur eine Meldung mit der Satznummer und der Satzlänge angezeigt.

## Hinweise

Nach dem Aufruf des Editors werden mit der Eingabeaufforderung

```
Gib Anweisung >
```

Editoranweisungen angefordert, durch die dem Editor mitgeteilt wird, was er tun soll. Nachdem eine Anweisung geschrieben ist, muss sie mit der Return-Taste abgeschickt werden. Erst dann wird sie interpretiert und ausgeführt. Die Anweisungen für den Editor sind im Kapitel »Editor« ab Seite 253 beschrieben.

## Definitionen

In den zu der Spezifikation DEFINITIONEN angegebenen Daten können Tabulatoren, Zeichen- und Stringgruppen, Funktionen, Colorierung der Daten, Parameter, Editormakros, Optionen sowie die Datei für Menüs und Hilfetexte definiert (voreingestellt) werden.

Ist die zur Spezifikation DEFINITIONEN angegebene Datei eine Segment-Datei, so werden nur die Daten des Segments mit dem Namen »DEFINITIONEN« ausgewertet (gilt nicht bei MODUS=!).

Die Definitionen sind identisch mit den entsprechenden Editoranweisungen. Es gibt jedoch einige Definitionen, die als Editoranweisung nicht möglich sind. Sie sind nachfolgend beschrieben:

Tabulatoren:

siehe Editoranweisungen Seite 286

Zeichengruppen:

siehe Editoranweisungen Seite 296

z! Alle Zeichengruppen löschen



**Stringgruppen:**

siehe Editoranweisungen Seite 296

S! Alle Stringgruppen löschen

**Funktionstasten:**

siehe Editoranweisungen Seite 287

F! Alle Funktionstasten löschen

**Colorierung der Daten:**

siehe Editoranweisungen Seite 298

C! Alle Colorierungen löschen

In TUSTEP können für den Hintergrund und für die Schrift jeweils 16 verschiedene Farben verwendet werden. Die Farben werden dazu hexadezimal von 0 bis F nummeriert. Betriebssystembedingt sind diese Farben unter Windows und Linux bzw. macOS nicht in der gleichen Reihenfolge angeordnet. Die Farben 1 und 4, 3 und 6, 9 und C sowie B und E sind jeweils vertauscht; z. B. ist unter Windows die Farbe 9 blau und die Farbe C rot, unter Linux und macOS ist die Farbe 9 rot und die Farbe C blau. Werden Farbangaben in Editor-Definitionen verwendet, so muss mit der folgenden Anweisung angegeben werden, unter welchem Betriebssystem die Definitionen (mit den auf diesem System geltenden Farbnummern) erstellt wurden, damit unter Windows und Linux bzw. macOS die Farben gleich dargestellt werden.

```
C=system
```

Hierbei ist `system` durch `WINDOWS`, `LINUX` oder `MAC` zu ersetzen. Diese Einstellung gilt für alle nachfolgenden Definitionen bis zur nächsten `COLOR`-Anweisung bzw. bis zum Ende der Definitionen.

**Tag-Attribut-Prüfung:**

siehe Editoranweisungen Seite 300

T! Alle Tag-Attribut-Prüfungen löschen

**Definieren zusätzlicher Tag- und Attribut-Namen:**

siehe Editoranweisung Seite 323

**Definieren der leeren Tags:**

siehe Editoranweisung Seite 323

**Parameter:**

siehe Editoranweisungen Seite 312

P! Alle Parameter löschen

**Makros:**

siehe Editoranweisungen Seite 288

Y! Alle Makros und Makroleisten löschen

Optionen:

siehe Automatisches Einstellen der Optionen Seite 334

Datei mit Menüs, Hilfetexten und zusätzlichen Definitionen:

- d=name Name der Datei, die Menüs (vgl. Steuerbefehl `SELECT` Seite 384), Hilfetexte (vgl. Steuerbefehl `DISPLAY` Seite 384) und zusätzlichen Definitionen (vgl. Steuerbefehl `DEFINE` Seite 386) enthält.
- d=+ Die Menüs, Hilfetexte und zusätzlichen Definitionen stehen in der Segment-Datei, in der auch diese Definition steht.
- d=- Keine Datei mit Menüs, Hilfetexte und zusätzlichen Definitionen vorhanden.

Eine Definition kann auf mehrere, unmittelbar aufeinander folgende Zeilen aufgeteilt werden. Fortsetzungszeilen müssen mit einem Leerzeichen beginnen, das bei der Auswertung übergangen wird.

Zeilen, die mit »=« beginnen, werden als Kommentar übergangen.

Es ist sinnvoll, Einstellungen für den Editor schon in der INI-Datei (siehe Seite 89) vorzunehmen, damit bei jeder neuen TUSTEP-Sitzung die Funktionstasten, Editormakros usw. den eigenen Bedürfnissen entsprechend automatisch definiert werden. Dies kann dadurch geschehen, dass in der INI-Datei ein Segment mit dem Namen `EDIT` eingetragen wird, das die entsprechenden Editor-Definitionen enthält. Da diese Einstellungen zu sehr von persönlichen Gewohnheiten abhängig sind, wird nachfolgend nur ein fragmentarisches Beispiel angegeben, um zu zeigen, wie die entsprechenden Definitionen aussehen können.

```

= --- Tabulator ---
TAB,,3 6 9 12 15 18 21 24 27 30
= --- Zeichengruppen ---
z:vk=aeiou
...
= --- Stringgruppen ---
s:uo=/und/oder/
...
= --- Funktionstasten ---
F21=x #*VEMO,<EDITOR>
...
= --- Colorierung ---
C=WINDOWS
C1,l=1A:'#{!}+'#{!}-'
...
= --- Parameter ---
P1,AA=@/
...
= --- Tag-Attribut-Prüfung ---
T1,document=-;</>/
...
= --- Makros und Makroleisten ---
Y,SA_D="Tübingen, den ", DATE_3
...
= --- Datei für Hilfetexte und Menüs ---
D=+
= --- Optionen ---
O=000000-000000 373B17-7484F4A437-7484F4A4B4-171B37-...

```

# Einfügen von Textteilen

#EINFUEGE

## Kommando

#EINFUEGE

Für dieses Kommando gibt es eine eigene Beschreibung.

## Leistung

Mit diesem Programm können Textteile, die in einer Datei stehen und über Kürzel (z. B. eine laufende Nummer) identifiziert werden, in die Daten einer anderen Datei eingefügt (eingemischt) werden. Sie werden jeweils an der Stelle in die Daten eingefügt, an der das gleiche Kürzel steht.

Anwendungsbeispiele:

- Fußnoten in den Text einmischen
- Kurzformen durch Volltext ersetzen
- Bausteinbriefe zusammenstellen
- Serienbriefe erstellen

# Formulare aufbereiten zum Drucken

**#FAUFBEREITE**

## Kommando

#FAUFBEREITE

Für dieses Kommando gibt es eine eigene Beschreibung.

## Leistung

Mit diesem Programm können Daten zum Drucken in einem vorgegebenen Format (z. B. Adressaufkleber, Bibliothekskärtchen, vordruckte Formulare) aufbereitet werden. Dabei kann u. a. angegeben werden:

- die Formulargröße,
- konstante Texte für jedes Formular (auch abhängig vom Vorkommen bestimmter Textteile im jeweiligen Formular),
- Zeilen- und Zeichenpositionen für die einzelnen Textteile,
- welche Textteile in Fortsetzungsformularen wiederholt werden sollen, falls ein Formular nicht ausreicht.

# Fehlerhalt ein/ausschalten

#FEHLERHALT

## Kommando

#FEHLERHALT

MODUS	= EIN	Fehlerhalt einschalten; bei Auftreten eines Fehlers keinen Signalton ausgeben.
	= AUS	Fehlerhalt ausschalten; bei Auftreten eines Fehlers keinen Signalton ausgeben.
	= SIGNAL	Fehlerhalt einschalten; bei Auftreten eines Fehlers einen Signalton ausgeben.
	= LOESCHEN	Fehlerhalt einschalten; bei Auftreten eines Fehlers noch anstehende Kommandos löschen.
TON	= -STD-	Bei Auftreten eines Fehlers den Standard-Ton ausgeben.
	= tonfolge	Bei Auftreten eines Fehlers die angegebene Tonfolge ausgeben.  Welche Angaben für <code>tonfolge</code> erwartet werden, ist beim Kommando #SIGNAL (siehe Seite 222) beschrieben. Die Angabe darf hier jedoch nicht mehr als 40 Zeichen umfassen.

## Leistung

Mit diesem Kommando kann der Fehlerhalt ein- und ausgeschaltet werden sowie die Tonfolge bestimmt werden, die bei Auftreten eines Fehlers ausgegeben werden soll.

Wird das Kommando ohne Spezifikationsangaben aufgerufen, so werden nur die aktuell eingestellten Werte angezeigt.

Wird ein TUSTEP-Programm wegen Fehler (z. B. weil die angegebene Datei nicht vorhanden ist oder weil Fehler in den Parameterangaben sind) abgebrochen, so ist es meist nicht sinnvoll, die nachfolgenden Kommandos trotzdem auszuführen.

Falls der Fehlerhalt ausgeschaltet ist, wird nach dem Auftreten eines Fehlers die Ausführung der Kommandos nicht unterbrochen.

Falls der Fehlerhalt eingeschaltet ist, werden im Batch die restlichen noch anstehenden Kommandos nicht mehr ausgeführt; das gleiche gilt im Dialog, wenn der Fehlerhalt mit dem Modus LOESCHEN eingeschaltet wurde. Andernfalls wird im Dialog nachgefragt, was mit den noch anstehenden Kommandos geschehen soll. Für die Anfrage gibt es drei zulässige Antworten:

- 1) H (Halt) Vor Ausführung der noch anstehenden Kommandos Vorrangkommandos, die anschließend einzeln (!) eingegeben werden, ausführen. Wird dann statt eines weiteren Vorrangkommandos eine leere Eingabezeile abgeschickt, wird die unterbrochene Kommandofolge fortgesetzt.
- 2) L (Löschen) Die noch anstehenden Kommandos löschen, d. h. nicht mehr ausführen.
- 3) W (Weitermachen) Mit der Ausführung der noch anstehenden Kommandos fortfahren.

Beim Initialisieren einer TUSTEP-Sitzung wird der Fehlerhalt auf Modus `SIGNAL` gesetzt.

# Formatieren (autom. Umbruch) von Texten

#FORMATIERE

## Kommando

#FORMATIERE

Für dieses Kommando gibt es eine eigene Beschreibung.

## Leistung

Mit diesem Programm können Texte zum Drucken aufbereitet werden. Die Aufteilung des Textes auf Zeilen und Seiten (einschließlich Silbentrennung, Randausgleich und Fußnoten) wird automatisch vorgenommen und kann über Parameter und über Anweisungen, die im Text enthalten sind, gesteuert werden.

## Hinweise

Die Daten in der QUELL-Datei bleiben unverändert. Das Ergebnis des Programms (die zum Drucken aufbereiteten Daten) wird in die PROTOKOLL-Datei ausgegeben. Diese kann anschließend mit dem Kommando #DRUCKE ausgedruckt (auf einem Drucker ausgegeben) werden.

In die ZIEL-Datei werden die Daten so ausgegeben, wie sie in der QUELL-Datei stehen (mit Ausnahme der Zeichenfolgen, die durch entsprechende Parameterangaben ausgetauscht werden). Dabei wird jedoch die Zeileneinteilung und die Seiten-Zeilenummer dem formatierten Ergebnis angeglichen.

Die ZIEL-Datei kann z. B. als neue Korrekturgrundlage dienen und hat dabei gegenüber der QUELL-Datei den Vorteil, dass im Editor die im ausgedruckten Ergebnis stehenden Seitenzahlen zum Auffinden der zu korrigierenden Textstellen verwendet werden können.

Soll vom formatierten Text ein Register erstellt werden, so muss als Grundlage des Registers die ZIEL-Datei verwendet werden, damit sichergestellt ist, dass die als Referenz verwendeten Seitenzahlen dem aktuellen Stand entsprechen.

Sollen die in den Daten enthaltenen Steuerzeichen nicht interpretiert, sondern mit ausgedruckt werden, so müssen die Daten (statt mit dem Kommando #FORMATIERE) mit dem Kommando #DVORBEREITE zum Drucken aufbereitet werden.



# Halt vor Ausführung noch anstehender Kommandos

#HALT

## Kommando

#HALT

(keine Spezifikationen)

## Leistung

Mit diesem Kommando kann das Abarbeiten einer Kommandofolge (an der Stelle, an der es in dieser Kommandofolge steht) angehalten werden. Dies kann z. B. sinnvoll sein, wenn vor der Ausführung des nächsten Kommandos einer Kommandofolge der USB-Speicher-Stick gewechselt werden soll. Das Kommando fragt, was mit den noch anstehenden Kommandos geschehen soll. Für die Anfrage gibt es drei zulässige Antworten:

- 1) H (Halt) Vor Ausführung der noch anstehenden Kommandos Vorrangkommandos, die anschließend einzeln (!) eingegeben werden, ausführen. Wird dann statt eines weiteren Vorrangkommandos eine leere Eingabezeile abgeschickt, wird die unterbrochene Kommandofolge fortgesetzt.
- 2) L (Löschen) Die noch anstehenden Kommandos löschen, d. h. nicht mehr ausführen.
- 3) W (Weitermachen) Mit der Ausführung der noch anstehenden Kommandos fortfahren.

# Online-Hilfe

#HILFE

## Kommando

#HILFE

(keine Spezifikationen)

## Leistung

Mit diesem Kommando können im Dialog die Texte der TUSTEP-Beschreibung im TUSTEP-Fenster angezeigt werden. Die Auswahl der Texte erfolgt über ein hierarchisch gegliedertes Inhaltsverzeichnis, in dem die einzelnen Themen ausgewählt werden können. Nach dem Aufruf der Online-Hilfe wird ein Auswahlménü mit den Hauptthemen angezeigt. Wird eines der angezeigten Themen ausgewählt, so wird jeweils in einem neuen Fenster das dazugehörige Auswahlménü mit den untergeordneten Themen bzw. der dazugehörige Text angezeigt.

## Hinweise

Die Online-Hilfe kann auch innerhalb des Editors mit der Anweisung `hilfe` oder dem Steuerbefehl `HELP` (Tastenkombination `Ctrl+O` bzw. `Strg+O`) aufgerufen werden.

Zur Anzeige der TUSTEP-Beschreibung kann auch das Kommando `#SUCHE` (ohne Spezifikationsangaben) verwendet werden. Der Zugriff auf bestimmte Stellen kann dabei über die einzelnen Inhaltsverzeichnisse im Volltext oder über einen Index erfolgen. Der Index enthält sowohl die in der Beschreibung vorkommenden Wörter als auch die Steueranweisungen und -befehle und die Zeichen- und Steuercodes, die im Text beschrieben sind.

Die TUSTEP-Beschreibung setzt sich aus mehreren Teilen zusammen. Diese können mit dem Makro `*DRUBE` ausgedruckt werden. Weitere Informationen über das Makro werden mit dem Kommando `#INFORMIERE, *DRUBE` ausgegeben.

## Navigation

Navigation mit der Maus:

In Auswahlménüs wird das zur angeklickten Zeile gehörende Unterménü bzw. der dazugehörige Text angezeigt. Auswahlménüs und Texte, die nicht ins TUSTEP-Fenster passen, können mit dem Mousrad verschoben werden.

Navigation mit der Tastatur:

Welche Tasten mit der in der linken Spalte angegebenen Tastenbezeichnungen gemeint sind, ist anschließend unter »Tastenbezeichnungen« beschrieben.

PfU	Cursor geht eine Zeile nach unten. Befindet er sich in der letzten Zeile des aktiven Fensters und sind noch weitere dazugehörige Daten nach dieser Zeile vorhanden, so wird der Inhalt des Fensters um eine Zeile nach oben verschoben und unten die nachfolgende Zeile angezeigt.
PfO	Cursor geht eine Zeile nach oben. Befindet er sich in der ersten Zeile des aktiven Fensters und sind noch weitere dazugehörige Daten vor dieser Zeile vorhanden, so wird der Inhalt des Fensters um eine Zeile nach unten verschoben und oben die vorangehende Zeile angezeigt.
PfR	In Auswahlmenüs wird das zu der Zeile, in der der Cursor steht, gehörende Untermenü bzw. der dazugehörige Text angezeigt.
PfL	Vom Hauptmenü aus wird das Online-Hilfe-Programm beendet; von Untermenüs und von Texten aus wird in das übergeordnete Menü zurückgekehrt.
Return	Wirkt gleich wie PfR
Delete	Wirkt gleich wie PfL
PgDn	Blättert innerhalb des aktuellen Auswahlmenüs bzw. Textes vorwärts weiter.
PgUp	Blättert innerhalb des aktuellen Auswahlmenüs bzw. Textes rückwärts weiter.
Pos1	Zeigt den Anfang des aktuellen Auswahlmenüs bzw. Textes an.
End	Zeigt das Ende des aktuellen Auswahlmenüs bzw. Textes an.
Escape	Beendet das Online-Hilfe-Programm.

### Tastenbezeichnungen

Return	»Eingabetaste« rechts im Buchstabenblock über der Shift-Taste.
Escape	»Abbruchtaste« ganz oben links, sie ist mit »Esc« beschriftet.

Die folgenden Tasten befinden sich im Mittelblock zwischen Buchstabenblock und Ziffernblock. Gleichbeschriftete Tasten im Ziffernblock können eventuell alternativ verwendet werden. Bei Tastaturen ohne Ziffernblock sind die folgenden Tasten in der rechten oberen und rechten unteren Ecke der Tastatur und erfordern eventuell ein zusätzliches Drücken der Fn-Taste.

Delete	»Löschtaste«, sie ist mit »Entf« oder »Del« beschriftet. Alternativ kann in TUSTEP auch Shift+Backspace verwendet werden.
--------	---

PfO	»Pfeil-nach-oben-Taste«
PfU	»Pfeil-nach-unten-Taste«
PfL	»Pfeil-nach-links-Taste«
PfR	»Pfeil-nach-rechts-Taste«
Pos1	»Anfangtaste«, sie ist mit »Pos1«, »Home« oder einem Pfeil nach links oben beschriftet. Unter macOS kann alternativ $f_n+P_{fL}$ verwendet werden.
End	»Endetaste«, sie ist mit »Ende«, »End« oder einem Pfeil nach rechts unten beschriftet. Unter macOS kann alternativ $f_n+P_{fR}$ verwendet werden.
PgUp	»Bild-rauf-Taste«, sie ist mit »Page Up«, »PgUp« oder mit »Bild« und einem Pfeil nach oben beschriftet. Unter macOS kann alternativ $f_n+P_{fO}$ verwendet werden.
PgDn	»Bild-runter-Taste«, sie ist mit »Page Down«, »PgDn« oder mit »Bild« und einem Pfeil nach unten beschriftet. Unter macOS kann alternativ $f_n+P_{fU}$ verwendet werden.

# Holen von Daten

#HOLE

## Kommando

#HOLE

QUELLE	= datei	Name der Datei, aus der die Daten geholt (kopiert) werden sollen.
	= -STD-	Die zu holenden (zu kopierenden) Daten stehen in der Standard-Text-Datei.
ZIEL	= datei	Name der Datei für die geholten (zu kopierenden) Daten.
	= -STD-	Die geholten (zu kopierenden) Daten in die Standard-Text-Datei ausgegeben.
MODUS	= -STD-	* Daten blockweise kopieren. Falls QUELL-Datei eine Band-Datei ist: Daten der Datei aus der Band-Datei in die ZIEL-Datei kopieren, deren Name zur Spezifikation ZIEL angegeben ist und deren Daten als letzte geändert wurden.
	= -	Daten satzweise kopieren, Satznummern beibehalten.
	= +	Daten satzweise kopieren, Sätze im Programmmodus neu nummerieren.
	= name	Falls QUELL-Datei eine Segment-Datei ist: Segment mit diesem Namen kopieren; falls QUELL-Datei eine Band-Datei ist: Daten der Datei mit diesem Namen aus der Band-Datei kopieren, deren Daten als letzte geändert wurden.
	= ?	Falls QUELL-Datei eine Segment-Datei oder eine Band-Datei ist: Inhaltsverzeichnis kopieren.
	= n	Falls QUELL-Datei eine Band-Datei ist: Daten der ersten Datei mit dem zur Spezifikation ZIEL angegebenen Namen aus der Band-Datei in die ZIEL-Datei kopieren, wobei die Suche nach dieser Datei mit der n-ten Datei in der Band-Datei beginnt.
	= n : name	Falls QUELL-Datei eine Band-Datei ist: Daten der ersten Datei mit diesem Namen aus der Band-Datei in die ZIEL-Datei kopieren, wobei die Suche nach die-

ser Datei mit der n-ten Datei in der Band-Datei beginnt.

LOESCHEN = - \* Daten in der ZIEL-Datei nicht löschen.  
 = + Daten in der ZIEL-Datei zuerst löschen.

## Leistung

Mit diesem Kommando können

- Daten aus einer Datei in eine andere Datei blockweise (besonders schnell) kopiert werden.
- Daten aus einer Datei in eine andere Datei satzweise kopiert werden. Auf Wunsch werden die Sätze dabei im Programmmodus neu nummeriert. Die ggf. beim Edieren entstandenen Lücken werden beseitigt und die Sätze in ihrer logischen Reihenfolge angeordnet (vgl. »Reorganisieren von Dateien« Seite 40).
- Daten aus einem Segment einer Segment-Datei in eine andere (normale) Datei kopiert werden.
- Daten aus einer Datei einer Band-Datei in eine andere (normale) Datei kopiert werden.

## Beispiele

Unter Windows die Datei mit dem Namen `arbeit` von einem USB-Speicher-Stick mit dem Laufwerksbuchstaben `E` blockweise in die Datei mit dem Namen `text` auf die Festplatte kopieren; falls die Datei `text` schon Daten enthält, diese zuvor löschen:

```
#AN, -*arbeit, TR=E
#HO, -*arbeit, text, , +
#AB, -*arbeit
```

Die Daten des Segments `test` in der Datei mit dem Namen `sammel` in die Datei mit dem Namen `hilf` kopieren; falls die Datei schon Daten enthält, diese zuvor löschen:

```
#HO, sammel, hilf, test, +
```

Die Daten in der Datei `hilf` im Programmmodus nummerieren. Dazu werden die Daten satzweise in die Standard-Text-Datei kopiert, dabei neu nummeriert und dann blockweise zurückkopiert:

```
#HO, hilf, -STD-, +, +
#HO, -STD-, hilf, , +
```

# Informieren über Makro-Datei, Variablen u. a.

#INFORMIERE

## Kommando

#INFORMIERE

MAKRO	= +	Namen der aktuellen Makro-Dateien ins Ablaufprotokoll ausgeben.
	= \$	Liste der Makros aus der ersten Makro-Datei ins Ablaufprotokoll ausgeben.
	= \$\$	Liste der Makros aus der zweiten Makro-Datei ins Ablaufprotokoll ausgeben.
	= \$\$\$	Liste der Makros aus der dritten Makro-Datei ins Ablaufprotokoll ausgeben.
	= -STD-	Liste der Makros, die in den TUSTEP-internen Makro-Dateien enthalten sind, ins Ablaufprotokoll ausgeben.
	= \$name	Beschreibung des Makros mit dem angegebenen Namen aus einer der aktuellen Makro-Dateien ins Ablaufprotokoll ausgeben; die Suche nach dem Makro beginnt in der ersten Makro-Datei.
	= \$\$name	Wie \$name, jedoch beginnt die Suche nach dem Makro in der zweiten Makro-Datei.
	= \$\$\$name	Wie \$name, jedoch beginnt die Suche nach dem Makro in der dritten Makro-Datei.
	= *name	Beschreibung des Makros mit dem angegebenen Namen aus einer TUSTEP-internen Makro-Datei ins Ablaufprotokoll ausgeben.
	= name	Wie \$name; wird jedoch kein Makro mit dem angegebenen Namen in einer der aktuellen Makro-Dateien gefunden, werden die TUSTEP-internen Makro-Dateien nach einem Makro mit dem angegebenen Namen durchsucht.
	= -	* Keine Information über aktuelle Makro-Dateien oder über Makros ausgeben.
VARIABLEN	= name	Definition der TUSTEP-Variablen mit dem angegebenen Namen ins Ablaufprotokoll ausgeben.

		Es können auch mehrere Variablenamen angegeben werden; sie müssen durch Apostroph getrennt sein.
	= +	Definition aller definierten TUSTEP-Variablen ins Ablaufprotokoll ausgeben.
	= -	* Keine Information über Variablen ausgeben.
PROJEKT	= +	Aktuellen Projektnamen ins Ablaufprotokoll ausgeben.
	= -	* Aktuellen Projektnamen nicht ausgeben.
NAME	= +	Aktuellen Namen ins Ablaufprotokoll ausgeben.
	= -	* Aktuellen Namen nicht ausgeben.
CODE	= +	Aktuelle Code-Einstellung für die Eingabe vom TUSTEP-Fenster und die Ausgabe in das TUSTEP-Fenster ins Ablaufprotokoll ausgeben.
	= -	* Aktuelle Code-Einstellung nicht ausgeben.
FUNKTIONEN	= +	Definition aller für die Kommandoebene definierten Funktionstasten ins Ablaufprotokoll ausgeben.
	= -	* Keine Information über Funktionstasten ausgeben.
ZEILEN	= +	Aktuelle Zeilenzahl ins Ablaufprotokoll ausgeben.
	= -	* Aktuelle Zeilenzahl nicht ausgeben.
SPALTEN	= +	Aktuelle Spaltenzahl ins Ablaufprotokoll ausgeben.
	= -	* Aktuelle Spaltenzahl nicht ausgeben.
SIGNAL	= +	Tonhöhe und Tonlänge des aktuellen Fehlersignals ins Ablaufprotokoll ausgeben.
	= -	* Tonhöhe und Tonlänge des aktuellen Fehlersignals nicht ausgeben.
FARBEN	= +	Codes der aktuellen Farben ins Ablaufprotokoll ausgeben.
	= -	* Codes der aktuellen Farben nicht ausgeben.
DATEINAMEN	= name	Definition des angegebenen »definierten Dateinamens« ins Ablaufprotokoll ausgeben.
		Es können auch mehrere Dateinamen angegeben werden; sie müssen durch Apostroph getrennt sein.
	= +	Definition aller »definierten Dateinamen« ins Ablaufprotokoll ausgeben.
	= -	* Keine Information über »definierte Dateinamen« ausgeben.



MAUS	= +	Aktuelle Einstellung der Mausrad-Verzögerung ins Ablaufprotokoll ausgeben.
	= -	* Aktuelle Einstellung der Mausrad-Verzögerung nicht ausgeben.

## Leistung

Mit diesem Kommando können folgende Informationen ausgegeben werden:

- Namen der aktuellen Makro-Dateien
- Namen der Makros in den aktuellen Makro-Dateien
- Namen der Makros in den TUSTEP-internen Makro-Dateien.
- Beschreibungen von Makros aus den aktuellen Makro-Dateien
- Beschreibungen von Makros aus den TUSTEP-internen Makro-Dateien
- Inhalt von TUSTEP-Variablen
- Namen aller definierten TUSTEP-Variablen und ihr Inhalt
- Aktueller Projektname, der zur Ergänzung der Dateinamen verwendet wird, falls dieser nicht explizit mit dem Dateinamen angegeben wird
- Aktueller Name, der bei Druckausgaben auf dem Deckblatt verwendet werden soll
- Aktuelle Code-Einstellung für die Eingabe vom TUSTEP-Fenster und die Ausgabe in das TUSTEP-Fenster
- Belegung der Funktionstasten auf Kommandoebene
- Anzahl der Zeilen, die im TUSTEP-Fenster angezeigt werden
- Anzahl der Spalten, die im TUSTEP-Fenster angezeigt werden
- Tonhöhe und Tonlänge des Fehlersignals, das unter Windows bei unerwarteter Tastatureingabe verwendet wird
- Codes der Farben, die für das Ablaufprotokoll im TUSTEP-Fenster verwendet werden
- Pfad mit Dateinamen von »definierten Dateinamen«
- Namen aller »definierten Dateinamen« mit ihren Pfaden und Dateinamen
- Aktuelle Einstellung der Mausrad-Verzögerung.

Wird das Kommando ohne Spezifikationsangaben aufgerufen, so werden folgende Information angezeigt:

- Name und Erstellungsdatum der verwendeten TUSTEP-Version
- Nummer und Name der aktuellen Sitzung sowie Datum und Uhrzeit des Sitzungsbeginns.

**Hinweis**

Alle Einstellungen, die sich mit dem Kommando #INFORMIERE anzeigen lassen, können mit dem Kommando #DEFINIERE (siehe Seite 132) geändert werden.

TUSTEP-Variablen können außerdem in Makros mit der Makroanweisung DEFINE (siehe Seite 422) definiert werden; auf ihren Inhalt kann in Kommandos und in Makros zugegriffen werden. Makrovariablen, die in Makros definiert werden, können mit diesem Kommando nicht abgefragt werden.

Als Beschreibung eines Makros gilt der Kommentar (das sind die Zeilen mit »\$\$-« in Spalte 1 bis 3), der unmittelbar am Anfang eines Makros in der Makro-Datei steht.

# Korrekturanweisungen ausführen

#KAUSFUEHRE

## Kommando

#KAUSFUEHRE

Für dieses Kommando gibt es eine eigene Beschreibung.

## Leistung

Mit diesem Programm können Texte (ohne Verwendung eines Editors) korrigiert werden. Mit Korrekturanweisungen, die in einer Datei stehen, können

- Zeilen (z. B. Zeile 2 auf Seite 3)
- Wörter (z. B. 2. Wort in Zeile 3 auf Seite 4)
- Zeichen (z. B. 2. Zeichen in Zeile 3 auf Seite 4  
oder 2. Zeichen im 3. Wort in Zeile 4 auf Seite 5)

ersetzt, gelöscht oder eingefügt werden. Auf Wunsch wird ein Protokoll der ausgeführten Korrekturen erstellt.

## Hinweis

Die Korrekturanweisungen für dieses Programm werden meist mit dem Kommando #VERGLEICHE erstellt und dann überarbeitet.

# Kopieren, Auswählen, Modifizieren von Texten

#KOPIERE

## Kommando

#KOPIERE

Für dieses Kommando gibt es eine eigene Beschreibung.

## Leistung

Mit diesem Programm können Dateien nicht nur kopiert, sondern gleichzeitig auch in vielfältiger Weise bearbeitet werden.

Über Parameter kann u. a. gesteuert werden

- die Überprüfung der Daten
- die Auswahl der Daten
- die Umstellung und Ergänzung von einzelnen Textteilen
- die Ersetzung von Zeichenfolgen
- die Bearbeitung von Kalenderdaten
- das Rechnen mit im Text enthaltenen Zahlen
- die Form der Ausgabe

## Listen von Dateinamen u. a.

#LISTE

### Kommando

#LISTE

MODUS	= -STD-	* Wenn zur Spezifikation DATEI keine Datei angegeben ist: Kurz-Informationen über die angemeldeten Dateien (einschließlich der Scratch-Dateien) auflisten.  Wenn zur Spezifikation DATEI eine TUSTEP-Datei oder eine Fremd-Datei angegeben ist: Den Anfang und das Ende dieser Datei auflisten.  Wenn zur Spezifikation DATEI eine Band-Datei angegeben ist: Die Namen der am Anfang und Ende in dieser Band-Datei stehenden Dateien auflisten.
	= SCRATCH	Kurz-Informationen der Scratch-Dateien auflisten.
	= n	Die ersten n Sätze der zur Spezifikation DATEI angegebenen Datei auflisten. Falls dies eine Band-Datei ist, werden die Namen der ersten n darin enthaltenen Dateien aufgelistet.
	= n ; m	Die ersten n und die letzten m Sätze der zur Spezifikation DATEI angegebenen Datei auflisten. Falls dies eine Band-Datei ist, werden die Namen der ersten n und der letzten m darin enthaltenen Dateien aufgelistet.
	= DATEINAMEN	Namen der existierenden (angemeldeten und nicht angemeldeten) Dateien eines Projekts auflisten.
	= TITEL	Wie DATEINAMEN; zusätzlich zu jedem Dateinamen auch den Dateititel (soweit vorhanden) auflisten.
	= SEGMENTE	Namen der angemeldeten Dateien (einschließlich der Scratch-Dateien) mit Segmenten, die mit der Hole-Anweisung des Editors oder mit dem Kommando #HOLE aus einer Segment-Datei kopiert wurden, jeweils mit Name des Segments und Name der Datei, aus der das Segment stammt; Segmente, die geändert und danach noch nicht wieder zurückkopiert wurden, werden markiert.

	=	PROJEKTNAMEN	Namen der existierenden Projekte eines Datenträgers auflisten.
	=	VARIABLEN	Namen und Inhalte der System-Variablen auflisten, die in TUSTEP benutzt werden können.
	=	TRAEGER	Namen und Inhalte der System-Variablen auflisten, die in TUSTEP-Kommandos als Angabe zur Spezifikation TRAEGER verwendet werden können und deren Inhalte jeweils den vollständigen Pfad eines Verzeichnisses angeben, auf das zugegriffen werden kann.  Unter Windows werden zusätzlich noch die Laufwerksbuchstaben der betriebsbereiten Laufwerke aufgelistet.
	=	DRUCKERTYPEN	Druckertypen auflisten, die in der verwendeten TUSTEP-Version unterstützt werden.
DATEI	=	-	* Normalfall.
	=	name	Bei den Modi <code>-STD-</code> , <code>»n«</code> und <code>»n;m«</code> : Name der Datei, deren Anfang und Ende aufgelistet werden soll; bei allen anderen Modi (mit Ausnahme von <code>SCRATCH</code> ) Name der Datei, in die die entsprechende Liste ausgegeben werden soll.
LOESCHEN	=	-	* Daten in der zur Spezifikation DATEI angegebenen Datei nicht löschen.
	=	+	Daten in der zur Spezifikation DATEI angegebenen Datei löschen, falls in diese auf Grund der Angabe zur Spezifikation MODUS eine Liste ausgegeben wird.
PROJEKT	=	name	Bei <code>MODUS=DATEINAMEN</code> : Name des Projekts, von dem die Namen aller Dateien aufgelistet werden sollen.
	=	+	* Bei <code>MODUS=DATEINAMEN</code> : Namen der Dateien des aktuellen Projekts auflisten.
	=	<code>-STD-</code>	Bei <code>MODUS=DATEINAMEN</code> : Namen der Dateien des Projekts, das beim Initialisieren der TUSTEP-Sitzung eingestellt wurde, auflisten; dieses Projekt ist durch die System-Variable <code>TUSTEP_PRJ</code> vorgegeben.
	=	-	Bei <code>MODUS=DATEINAMEN</code> : Namen der Dateien auflisten, die keinem Projekt zugeordnet sind.
TRAEGER	=	name	Bei den Modi <code>DATEINAMEN</code> und <code>PROJEKTNAMEN</code> : Name der System-Variablen, die den Pfad für die Dateien bzw. Projekte enthält, deren Namen aufgelistet werden sollen.  Bei den Modi <code>-STD-</code> , <code>»n«</code> und <code>»n;m«</code> : Name der Sys-

		tem-Variablen, die den Pfad für die zur Spezifikation DATEI angegebene Datei enthält.
		Unter Windows kann auch direkt der Laufwerksbuchstabe angegeben werden, falls der Pfad keinen Verzeichnisnamen enthält.
	= -STD-	* Bei den Modi DATEINAMEN und PROJEKTNAMEN: Namen der Dateien bzw. der Projekte auflisten, die sich auf dem Standard-Träger für permanente Dateien befinden; dieser ist durch die System-Variable TUSTEP_DSK vorgegeben. Bei den Modi -STD-, »n« und »n;m«: Die zur Spezifikation DATEI angegebene Datei befindet sich auf dem Standard-Träger für temporäre bzw. permanente Dateien; diese sind durch die System-Variable TUSTEP_SCR bzw. TUSTEP_DSK vorgegeben.
POSITIV	= -	* Normalfall.
	= ...	Falls eine Liste mit Dateinamen ausgegeben wird, nur solche Namen in die Liste aufnehmen, die mindestens einem der angegebenen Muster entsprechen.  Die Besonderheiten bei der Angabe von Mustern sind unter »Muster zur Auswahl der Namen/Dateien« auf Seite 178 beschrieben.
NEGATIV	= -	* Normalfall.
	= ...	Falls eine Liste mit Dateinamen ausgegeben wird, nur solche Namen in die Liste aufnehmen, die keinem der angegebenen Muster entsprechen.  Die Besonderheiten bei der Angabe von Mustern sind unter »Muster zur Auswahl der Namen/Dateien« auf Seite 178 beschrieben.
TYP	= -	* Keine Auswahl auf Grund des Dateityps treffen.
	= FDF	Beim Auflisten von Dateinamen nur Dateien vom Typ FDF berücksichtigen.
	= SEQ	Beim Auflisten von Dateinamen nur Dateien vom Typ SEQ berücksichtigen.
	= RAN	Beim Auflisten von Dateinamen nur Dateien vom Typ RAN berücksichtigen.
	= TUSTEP	Beim Auflisten von Dateinamen nur Dateien vom Typ SEQ und Typ RAN berücksichtigen.
	= TAPE	Beim Auflisten von Dateinamen nur Dateien vom Typ TAPE berücksichtigen.
ERSETZEN	= -	* Normalfall.

---

	= ...	Bei den Modi DATEINAMEN und PROJEKTNAMEN: In der Liste mit den Namen die angegebenen Ersetzungen vornehmen, bevor sie in die zur Spezifikation DATEI angegebene Datei ausgegeben werden.  Die Besonderheiten bei der Angabe von Ersetzungen sind unter »Zeichenfolgen zum Ersetzen in den Dateinamen« auf Seite 179 beschrieben.
CODE	= -	* Normalfall.
	= +	Aktuelle Umcodiertabelle für das TUSTEP-Fenster auflisten.
	= ASCII	Daten dem internationalen ASCII-Code (siehe Tabelle Seite 815) entsprechend umcodieren.
	= EBCDIC	Daten dem internationalen EBCDIC-Code (vgl. Code-Tabellen Seite 838) entsprechend umcodieren.
	= IBMPC	Daten dem IBM-PC-Zeichensatz (siehe Tabelle Seite 817) entsprechend umcodieren.
	= ANSI	Wie CP1252.
	= CPnnn	Daten der Code-Page nnn (nnn = 437, 850, 852, 1250, 1252, 10000, 10029) entsprechend umcodieren (siehe Tabellen ab Seite 818).
	= ISO8859	Daten dem Zeichensatz 1 der ISO-Norm 8859 (siehe Tabelle Seite 826) entsprechend umcodieren.
	= LATINn	Daten dem Code Latin n (n = 1 bis 10) entsprechend umcodieren (siehe Tabellen ab Seite 828).
	= LINUX	Daten dem Linux-Zeichensatz (siehe Tabelle Seite 827) entsprechend umcodieren.
	= DECMCS	Daten dem »DEC Multinational Character Set« (siehe Tabelle Seite 825) entsprechend umcodieren.
	= UNICODE	Wie UTF16.
	= UTF16	Daten den UTF-16-Konventionen entsprechend umcodieren.
	= UTF8	Daten den UTF-8-Konventionen entsprechend umcodieren.
MUSTER	= -	* Normalfall.
	= ...	Falls Dateinamen aufgelistet werden, nur die Namen der TUSTEP-Dateien (Fremd-Dateien und Band-Dateien bleiben unberücksichtigt) in die Liste aufnehmen, die im Dateititel oder in mindestens einer Zeile ein angegebenes Muster enthalten.



Falls Anfang und/oder Ende einer Datei aufgelistet werden, nur die Zeilen in die Liste aufnehmen, die ein angegebenes Muster enthalten.

Falls Namen der in einer Band-Datei stehenden Dateien aufgelistet werden, nur die Dateinamen in die Liste aufnehmen, die ein angegebenes Muster enthalten.

Die Besonderheiten bei der Angabe von Mustern sind unter »Muster zur Auswahl der Namen/Dateien« auf Seite 178 beschrieben.

Aus syntaktischen Gründen dürfen Nummernzeichen, Kommata, Gleichheitszeichen, Apostrophe, Anführungszeichen und Leerzeichen nicht in den Mustern vorkommen; diese Einschränkung gilt nicht, wenn als Spezifikationswert \* angegeben wird.

= \* Die Muster folgen auf das #LISTE-Kommando und sind mit \*EOF abgeschlossen.

Alle Muster müssen durch ein frei wählbares Begrenzungszeichen begrenzt sein. Der besseren Lesbarkeit wegen empfiehlt es sich, die Zeileneinteilung so zu wählen, dass jede Zeile mit einem Begrenzungszeichen anfängt und mit einem Begrenzungszeichen endet.

Zur Spezifikation OPTIONEN muss mindestens eine der Optionen AND, OR oder USER angegeben werden.

OPTIONEN = - \* Normalfall (keine Optionen).

= . . . Optionen für die zur Spezifikation MUSTER angegebene Recherchier-Tabelle. Diese Optionen sind im Kapitel »Makros« ab Seite 441 beschrieben.

Darüber hinaus können noch die Optionen TITEL und DATEN angegeben werden. Wird TITEL, aber nicht DATEN angegeben, so wird das Muster nur im Dateititel gesucht; wird DATEN, aber nicht TITEL angegeben, so wird das Muster nur in den Daten gesucht. In allen anderen Fällen wird das Muster im Dateititel und in den Daten gesucht.

Es können mehrere Optionen angegeben werden; sie müssen durch Apostroph getrennt sein.

## Leistung

Mit diesem Kommando können die nachfolgend genannten Informationen ausgegeben werden. Wenn nichts anderes angegeben ist, werden sie ins Ablaufprotokoll ausgegeben.

- Namen der angemeldeten Dateien oder der Scratch-Dateien

Zu den angemeldeten Dateien zählen auch Dateien, die eingerichtet und noch nicht abgemeldet oder noch nicht gelöscht wurden. Außer dem Dateinamen werden bei TUSTEP-Dateien noch folgende Informationen aufgelistet:

- Dateityp
  - ob zum Lesen (L) oder Schreiben (S) angemeldet; eine Scratch-Datei ist daran zu erkennen, dass diese Angabe fehlt
  - Datum der letzten Änderung
  - Anzahl der Sätze in der Datei
  - Größe der Datei in KB
  - Belegung der Datei mit Daten in Prozent
  - Name des Datenträgers (der System-Variablen)
- Anfang und Ende einer Datei

Falls die Datei, deren Anfang/Ende aufgelistet werden soll, nicht angemeldet ist, wird sie auf dem angegebenen Träger automatisch an- und wieder abgemeldet.

Um den Inhalt von Fremd-Dateien anzeigen zu können, müssen die Daten dem jeweiligen Code entsprechend umcodiert werden. Da der Code häufig nicht automatisch erkannt werden kann, sollte er mit der Spezifikation `CODE` angegeben werden.

- Namen, Änderungsdatum und Größe der in einer Band-Datei enthaltenen Dateien.
- Namen der existierenden Dateien und Projekte
- Namen und Inhalt von System-Variablen
- Druckertypen, die unterstützt werden

Der Druckertyp muss beim Aufbereiten von Daten zum Drucken mit dem Parameter `DRT` und beim Drucken mit dem Kommando `#DRUCKE` zur Spezifikation `TYP` angegeben werden.

## Hinweise

Soll z. B. eine Kommandofolge für alle Dateien, deren Dateinamen eine bestimmte Zeichenfolge enthalten bzw. nicht enthalten, ausgeführt werden, so können diese Dateinamen mit dem Kommando `#LISTE` ausgewählt und in eine Datei geschrieben werden; diese Datei kann dann, nachdem ggf. die darin stehenden Dateinamen mit dem Editor kontrolliert und korrigiert wurden, beim Kommando `#TUE` (siehe Seite 233) zur Spezifikation `SCHLEIFE` angegeben werden.

Mit dem Kommando `#INFORMIERE` (siehe Seite 167) können weitere Informationen abgefragt werden.

## Muster zur Auswahl der Namen/Dateien

Zu den Spezifikationen `POSITIV` bzw. `NEGATIV` können Muster (Zeichenfolgen) angegeben werden, von denen mindestens eines auf den Namen zutreffen muss bzw.

von denen keines auf den Namen zutreffen darf, damit der Name in die Liste aufgenommen wird.

Zur Spezifikation `MUSTER` können Muster (Zeichenfolgen) angegeben werden, von denen mindestens eines im Dateititel oder in einer Zeile der Datei vorkommen muss, damit der Name der Datei in die Liste aufgenommen wird.

Die Muster dürfen nicht in Begrenzungszeichen eingeschlossen werden. Mehrere Muster können durch Apostroph getrennt angegeben werden. Falls jedoch zur Spezifikation `MUSTER` der Wert `*` angegeben ist und die Muster auf das `#LISTE`-Kommando folgen, müssen diese Muster in Begrenzungszeichen eingeschlossen werden. Das Begrenzungszeichen kann in diesem Fall frei gewählt werden.

Das Fragezeichen und der Stern haben eine spezielle Bedeutung. Ein Fragezeichen steht stellvertretend für ein beliebiges `TUSTEP`-Zeichen und ein Stern für null bis beliebig viele beliebige `TUSTEP`-Zeichen.

An Stelle eines einzelnen Zeichens einer Zeichenfolge kann auch eine in eckige Klammern eingeschlossene Gruppe von Zeichen angegeben werden. Die Zeichen innerhalb der eckigen Klammern werden als eine Zeichengruppe mit den angegebenen Zeichen interpretiert; sie steht stellvertretend für irgend eines der Zeichen.

Die möglichen Angaben zu den Spezifikationen `POSITIV`, `NEGATIV` und `MUSTER` entsprechen denen einer Recherchier-Tabelle (Parameterart XII). Diese ist für die Parameter-Konvention `»{ }«` ab Seite 724 und für die Parameter-Konvention `»<>«` ab Seite 756 beschrieben. Es sind jedoch nicht alle Angaben, die in einer Recherchier-Tabelle möglich sind, auch für die Angaben zu den Spezifikationen `POSITIV` und `NEGATIV` sinnvoll. Nach welcher Konvention die Parameter interpretiert werden, kann mit dem Kommando `#PARAMETER` (siehe Seite 207) eingestellt werden.

## Zeichenfolgen zum Ersetzen in den Dateinamen

Zur Spezifikation `ERSETZEN` können paarweise Zeichenfolgen angegeben werden, deren jeweils erste Zeichenfolge die Suchzeichenfolge ist, die durch die jeweils zweite Zeichenfolge, die Ersatzzeichenfolge, ersetzt werden soll. Außerdem können ggf. Zeichenfolgen angegeben werden, die beim Ersetzen übergangen werden sollen (Ausnahmezeichenfolgen).

Such- und Ersatzzeichenfolgen dürfen verschieden lang sein. Sie müssen durch ein Begrenzungszeichen voneinander getrennt werden. Das Begrenzungszeichen ist das erste Zeichen des Spezifikationswertes und ist frei wählbar. Aus syntaktischen Gründen dürfen jedoch Nummernzeichen, Kommata, Gleichheitszeichen, Apostrophe, Anführungszeichen und Leerzeichen weder als Begrenzungszeichen verwendet werden, noch in den Such-, Ersatz- oder Ausnahmezeichenfolgen vorkommen. Das Umschalten von Zeichenfolgenpaaren auf Ausnahmezeichenfolgen geschieht dadurch, dass an einer Stelle, an der eine Suchzeichenfolge erwartet wird, nochmals ein Begrenzungszeichen steht. Das letzte Zeichen des Spezifikationswertes muss ebenfalls ein Begrenzungszeichen sein.

Die möglichen Angaben zur Spezifikation ERSETZEN entsprechen denen einer Austausch-Tabelle (Parameterart X). Diese ist für die Parameter-Konvention »{ }« ab Seite 721 und für die Parameter-Konvention »<>« ab Seite 752 beschrieben. Es sind jedoch nicht alle Angaben, die in einer Austausch-Tabelle möglich sind, auch für die Angaben zur Spezifikation ERSETZEN sinnvoll. Nach welcher Konvention die Parameter interpretiert werden, kann mit dem Kommando #PARAMETER (siehe Seite 207) eingestellt werden.

## Beispiele

Namen und Kurz-Informationen der angemeldeten Dateien auflisten:

```
#LI
```

Namen der im aktuellen Projekt existierenden Dateien auflisten:

```
#LI, DA
```

Namen der im aktuellen Projekt existierenden Dateien mit der Namensweiterung `tf` oder `pdf` auflisten:

```
#LI, DA, POS=*tf'*.pdf
```

Namen der im aktuellen Projekt existierenden TUSTEP-Dateien auflisten, die das Wort `hilfe` oder `SOS` enthalten:

```
#LI, DA, MUSTER=hilfe'sos, word
```

Namen der im Projekt `doku` existierenden Dateien auflisten:

```
#LI, DA, PR=doku
```

Unter Windows die Namen der Dateien auflisten, die im Hauptverzeichnis des USB-Speicher-Sticks mit dem Laufwerksbuchstaben `E` stehen:

```
#LI, DA, PR=-, TR=E
```

Anfang und Ende der Datei `arbeit` auflisten:

```
#LI, , arbeit
```

Alle Versionen der Datei `text`, die in der Band-Datei `backup` enthalten sind, auflisten:

```
#LI, 9999, backup, POS=text
```

Druckertypen auflisten, bei denen im Namen `PS` vorkommt:

```
#LI, DR, POS=*ps*
```

Reorganisieren aller im Projekt doku stehenden Dateien, deren Namen mit »text« beginnt:

```
#DA, liste
#LI, DA, liste, +, PR=doku, POS=text*
#TU, *, SCHLEIFE=liste
#AN, , ?0
#RE, ?0, -STD-, +, +
#AB, ?0
*EOF
#LO, , liste
```

# Löschen von Daten/Dateien und Projekten

#LOESCHE

## Kommando

#LOESCHE

DATEN	= datei	Name der Datei mit den zu löschenden Daten. Es können auch mehrere Dateinamen angegeben werden; sie müssen durch Apostroph getrennt sein. Zu den Spezifikationen PROJEKT, TRAEGER, POSITIV und NEGATIV darf nur die Voreinstellung der jeweiligen Spezifikation angegeben werden; sie ist in diesem Fall aber bedeutungslos.
	= -	* In keiner Datei Daten löschen.
	= +	In den mit den Spezifikationen PROJEKT/TRAEGER/POSITIV/NEGATIV ausgewählten Dateien die Daten löschen.
DATEI	= datei	Name der zu löschenden Datei. Es können auch mehrere Dateinamen angegeben werden; sie müssen durch Apostroph getrennt sein. Zu den Spezifikationen PROJEKT, TRAEGER, POSITIV und NEGATIV darf nur die Voreinstellung der jeweiligen Spezifikation angegeben werden; sie ist in diesem Fall aber bedeutungslos.
	= -	* Keine Datei löschen.
	= +	Die mit den Spezifikationen PROJEKT/TRAEGER/POSITIV/NEGATIV ausgewählten Dateien löschen.
PROJEKT	= name	Name des Projekts, von dem Dateien gelöscht werden sollen bzw. Name des Projekts, das gelöscht werden soll. Soll ein Projekt gelöscht werden, so darf zu den beiden Spezifikationen DATEN und DATEI nichts (bzw. nur die Voreinstellung »-«) angegeben werden und zur Spezifikation TRAEGER muss der Träger des zu löschenden Projekts angegeben werden.
	= +	Dateien des aktuellen Projekts löschen.
	= -STD-	Dateien des Projekts löschen, das beim Initialisieren

		der TUSTEP-Sitzung eingestellt wurde; dieses ist durch die System-Variable TUSTEP_PRJ vorgegeben.
	= -	Dateien löschen, die keinem Projekt zugeordnet sind.
	= ?	* Dateien unabhängig davon löschen, welchem Projekt sie zugeordnet sind.
TRAEGER	= name	Name der System-Variablen, die den Pfad für die zu löschenden Dateien bzw. für das zu löschende Projekt enthält.  Unter Windows kann auch direkt der Laufwerksbuchstabe angegeben werden, falls der Pfad keinen Verzeichnisnamen enthält.
	= -STD-	Dateien bzw. Projekte löschen, die sich auf dem Standard-Träger für permanente Dateien befinden; dieser ist durch die System-Variable TUSTEP_DSK vorgegeben.
	= -	Dateien mit »definierten Dateinamen« löschen.
	= ?	* Dateien unabhängig davon löschen, auf welchem Datenträger sie sich befinden.
POSITIV	= -	* Keine Positiv-Auswahl von Dateien, deren Name einem vorgegebenen Muster entspricht.
	= ...	Nur solche Dateien löschen, deren Dateinamen mindestens einem der angegebenen Muster entspricht.
NEGATIV	= -	* Keine Negativ-Auswahl von Dateien, deren Name einem vorgegebenen Muster entspricht.
	= ...	Nur solche Dateien löschen, deren Dateinamen keinem der angegebenen Muster entspricht.
WISCHEN	= +	Die zu löschenden Daten bzw. die in den zu löschenden Dateien stehenden Daten überschreiben.
	= -	Die zu löschenden Daten bzw. die in den zu löschenden Dateien stehenden Daten nicht überschreiben.
FRAGEN	= -STD-	* Falls DATEN=+ oder DATEI=+ angegeben ist, so wird jeweils nachgefragt, ob die Daten der Datei gelöscht werden dürfen bzw. ob die Datei gelöscht werden darf.
	= +	Jeweils nachfragen, ob die Daten der Datei gelöscht werden dürfen bzw. ob die Datei gelöscht werden darf.
	= -	Daten bzw. Dateien ohne nachzufragen löschen.

## Leistung

Mit diesem Kommando können sowohl Daten in Dateien als auch Dateien selbst gelöscht werden. Werden nur die Daten gelöscht, so existiert die Datei weiterhin, ist aber leer. Wird die Datei gelöscht, so existiert die Datei anschließend nicht mehr. In jedem Fall müssen dazu die betroffenen Dateien zum Schreiben angemeldet sein.

Außerdem können mit diesem Kommando Projekte (Verzeichnisse) gelöscht werden. In einem zu löschenden Projekt dürfen keine Dateien mehr eingetragen sein. Sie müssen ggf. vorher gelöscht oder so umbenannt werden (mit dem Kommando #AENDERE, siehe Seite 122), dass sie damit in ein anderes Verzeichnis eingetragen werden.

Sollen alle zum Schreiben angemeldeten Dateien eines bestimmten Projekts gelöscht werden, genügt es, ein »+« zur Spezifikation DATEN bzw. DATEI und den Namen dieses Projekt zur Spezifikation PROJEKT anzugeben. Entsprechend können alle zum Schreiben angemeldeten Dateien eines Datenträgers durch eine entsprechende Angabe zur Spezifikation TRAEGER gelöscht werden. Wird zu beiden Spezifikationen etwas angegeben, so werden alle angemeldeten Dateien dieses Projekts auf diesem Datenträger gelöscht.

Sollen die Daten in allen Dateien bzw. alle Dateien gelöscht werden, deren Dateinamen eine bestimmte Zeichenfolge enthalten bzw. nicht enthalten, so können zur Spezifikation POSITIV Muster angegeben werden, von denen mindestens dem Dateinamen entsprechen muss, damit die Daten bzw. Dateien gelöscht werden; zur Spezifikation NEGATIV können Muster angegeben werden, von denen keines dem Dateinamen entsprechen darf, damit die Daten bzw. die Datei gelöscht werden. Die möglichen Angaben sind beim Kommando #LISTE auf Seite 178 beschrieben.

Ob die in den Dateien stehenden Daten überschrieben (Datenschutz!) werden sollen, kann mit der Spezifikation WISCHEN angegeben werden. Wird nichts angegeben, so werden die Daten überschrieben, wenn nicht mit dem Kommando #WISCHEN (siehe Seite 250) ein anderer Modus eingestellt wurde.

Falls auf Grund der Spezifikation FRAGEN nachgefragt wird, ob die Daten einer Datei gelöscht werden sollen bzw. ob eine Datei gelöscht werden soll, sind vier Antworten vorgesehen, die abgekürzt werden können:

- 1) JA Die Daten löschen bzw. die Datei löschen.
- 2) NEIN Die Daten nicht löschen bzw. die Datei nicht löschen.
- 3) ALLE Die Daten löschen bzw. die Datei löschen und alle noch zum Löschen vorgesehenen Daten bzw. Dateien löschen ohne nachzufragen.
- 4) KEINE Die Daten nicht löschen bzw. die Datei nicht löschen und alle noch zum Löschen vorgesehenen Daten bzw. Dateien nicht löschen.

Falls mit dem Kommando #PROTOKOLL (siehe Seite 209) der Protokoll-Modus nicht auf SYSTEM eingestellt wurde, kann statt KEINE einzugeben auch die Escape-Taste gedrückt werden.



## Beispiele

Daten in der Datei mit dem Namen `arbeit` löschen:

```
#LO, arbeit
```

Datei mit dem Namen `arbeit` löschen:

```
#LO, , arbeit
```

Alle angemeldeten Dateien des Projekts `doku` löschen:

```
#LO, , +, PR=doku
```

Alle angemeldeten Dateien mit der Namenserweiterung `xy` löschen:

```
#LO, , +, POS=*.xy
```

Unter Windows alle angemeldeten Dateien, die auf dem USB-Speicher-Stick mit dem Laufwerksbuchstaben `E` stehen, löschen:

```
#LO, , +, TR=E
```

# Zusammenstellen einer Kommandofolge

#MAKRO

## Kommando

#MAKRO

QUELLE	= <code>datei</code>	Name der Datei mit den Makroanweisungen und Daten.
	= *	* Makroanweisungen und Daten folgen auf das #MAKRO-Kommando und sind mit *EOF abgeschlossen.
PROTOKOLL	= -	* Keine zusätzliche Protokollierung der erzeugten Kommandofolge.
	= +	Erzeugte Kommandofolge ins Ablaufprotokoll ausgeben.
	= <code>datei</code>	Name der Datei für das Protokoll der erzeugten Kommandofolge.
AUSFUEHRUNG	= +	* Erzeugte Kommandofolge ausführen.
	= -	Erzeugte Kommandofolge nicht ausführen (sinnvoll, wenn die Kommandofolge zu Testzwecken nur protokolliert werden soll).
	= <code>datei</code>	Erzeugte Kommandofolge in die angegebene Datei ausgeben (nicht ausführen).

## Leistung

Die Leistung dieses Kommandos wird durch die zur Spezifikation `QUELLE` angegebenen Makroanweisungen und Daten vorgegeben und kann auf Grund von Zustandsabfragen und Interaktion mit dem Benutzer variiert werden.

Die Leistung kann allein im Ausführen von Makroanweisungen bestehen oder im Zusammenstellen einer Kommandofolge (ggf. einschließlich Parameter), die anschließend automatisch ausgeführt wird.

## Beispiel

Ein Beispiel für die Anwendung dieses Kommandos befindet sich im Anschluss an die Beschreibung der Makros (Seite 664).

# Ausgabe in eine Band-Datei

#MBAUSGABE

## Kommando

#MBAUSGABE

BAND	= name	Name der Band-Datei.
NUMMER	= -STD-	* Dateien hinter die letzte Datei in der Band-Datei ausgeben.
	= n	Dateien ab der n-ten Dateiposition in die Band-Datei ausgeben.
DATEI	= datei	Name der Datei, deren Daten in die Band-Datei ausgegeben werden.  Es können auch mehrere Dateinamen angegeben werden; sie müssen durch Apostroph getrennt sein.
LABEL	= -STD-	* Die Dateien erhalten in der Band-Datei die zur Spezifikation DATEI angegebenen Namen.
	= name	Die Dateien erhalten in der Band-Datei die hier angegebenen Namen.  Die Anzahl der Namen muss mit der Anzahl der Dateien, die zur Spezifikation DATEI angegeben sind, übereinstimmen.
DENSITY	=	(nicht mehr definiert)
CODE	= ASCII	* Band-Datei im ASCII-Code beschreiben.
	= EBCDIC	Band-Datei im EBCDIC-Code beschreiben.
GERAET	= -	* Band-Datei ist angemeldet.
	= -STD-	Pfad für die Band-Datei ist durch die System-Variable TUSTEP_MT1 vorgegeben.
	= name	Name der System-Variablen, die den Pfad für die Band-Datei enthält.  Unter Windows kann auch direkt der Laufwerksbuchstabe angegeben werden, falls der Pfad für die Band-Datei keinen Verzeichnisnamen enthält.
PROTOKOLL	= +	Liste aller in der Band-Datei stehenden Dateien ins Ablaufprotokoll ausgeben.

---

	= -STD-	Liste aller in der Band-Datei stehenden Dateien in die Standard-Protokoll-Datei ausgeben.
	= <code>datei</code>	Name der Datei für die Liste aller in der Band-Datei stehenden Dateien.
	= -	* Keine Liste der Dateien ausgeben.
LOESCHEN	= -	* Daten in der PROTOKOLL-Datei nicht löschen.
	= +	Daten in der PROTOKOLL-Datei zuerst löschen.

## Leistung

Mit diesem Kommando können Dateien in eine Band-Datei (zur Datensicherung und zum Datenaustausch) kopiert werden.

Zusätzlich kann nach dem Kopieren der Dateien eine Liste mit den Namen der Dateien ausgegeben werden, die dann in der Band-Datei stehen. Diese Liste entspricht derjenigen, die mit dem Kommando #MBINFORMIERE (siehe Seite 191) erstellt werden kann.

## Warnung

Werden die Dateien nicht hinter die letzte Datei in die Band-Datei geschrieben, so gehen sowohl die Datei, die überschrieben wird, als auch alle nachfolgenden Dateien, die schon in der Band-Datei standen, verloren.

## Hinweis

Die Band-Datei muss entweder angemeldet sein oder es muss zur Spezifikation GERAET eine System-Variable angegeben werden, die den Pfad zur Band-Datei enthält; im letzteren Fall ist die Angabe eines Projekts zusammen mit dem Namen der Band-Datei nicht erlaubt und es wird auch kein Projektname ergänzt.

## Beispiel

Unter Windows soll die Datei `erfass` ans Ende der Band-Datei `sammel` geschrieben werden. Die Band-Datei `sammel` befinde sich im Hauptverzeichnis (root directory) des Laufwerks F (dies kann z. B. auch ein USB-Speicher-Stick sein).

```
#MBA, sammel, , erfass, GER=F
```

# Eingabe von einer Band-Datei

#MBEINGABE

## Kommando

#MBEINGABE

BAND	= name	Name der Band-Datei.
NUMMER	= -STD-	* Jeweils die Datei mit dem entsprechenden Namen einlesen, deren Daten als letzte geändert wurden, bevor sie in die Band-Datei geschrieben wurde.
	= n	Jeweils die erste Datei mit dem entsprechenden Namen einlesen, wobei die Suche mit der n-ten Datei in der Band-Datei beginnt.
DATEI	= datei	Name der Datei, in die die von der Band-Datei eingelesenen Daten eingetragen werden sollen.  Es können auch mehrere Dateinamen angegeben werden; sie müssen durch Apostroph getrennt sein.
LABEL	= -STD-	* Die einzulesenden Dateien haben in der Band-Datei die zur Spezifikation DATEI angegebenen Namen.
	= name	Die einzulesenden Dateien haben in der Band-Datei die hier angegebenen Namen.  Die Anzahl der Namen muss mit der Anzahl der Dateien, die zur Spezifikation DATEI angegeben sind, übereinstimmen.
LOESCHEN	= -	* Daten in den zur Spezifikation DATEI angegebenen Dateien nicht löschen.
	= +	Daten in den zur Spezifikation DATEI angegebenen Dateien zuerst löschen.
CODE	= ASCII	* Band-Datei ist im ASCII-Code beschrieben.
	= EBCDIC	Band-Datei ist im EBCDIC-Code beschrieben.
GERAET	= -	* Band-Datei ist angemeldet.
	= -STD-	Pfad für die Band-Datei ist durch die System-Variable TUSTEP_MT1 vorgegeben.
	= name	Name der System-Variablen, die den Pfad für die Band-Datei enthält.  Unter Windows kann auch direkt der Laufwerks-

buchstabe angegeben werden, falls der Pfad für die Band-Datei keinen Verzeichnisnamen enthält.

## Leistung

Mit diesem Kommando können Dateien von einer Band-Datei in einzelne Dateien kopiert werden.

## Hinweis

Die Band-Datei muss entweder angemeldet sein oder es muss zur Spezifikation `GERAET` eine System-Variable angegeben werden, die den Pfad zur Band-Datei enthält; im letzteren Fall ist die Angabe eines Projekts zusammen mit dem Namen der Band-Datei nicht erlaubt und es wird auch kein Projektname ergänzt.

## Beispiel

Unter Windows soll die Datei `erfass` von der Band-Datei `sammel` in die Datei `e` kopiert werden. Falls die Datei `e` schon Daten enthält, sollen sie zuvor gelöscht werden. Wenn in der Band-Datei mehrere Dateien mit dem Namen `erfass` vorhanden sind, soll die Datei mit diesem Namen kopiert werden, deren Daten als letzte geändert wurden. Die Band-Datei `sammel` befinde sich im Hauptverzeichnis (root directory) des Laufwerks `F` (dies kann z. B. auch ein USB-Speicher-Stick sein).

```
#MBE, sammel, , e, erfass, +, GER=F
```

# Informieren über eine Band-Datei

#MBINFORMIERE

## Kommando

#MBINFORMIERE

BAND	= name	Name der Band-Datei.
PROTOKOLL	= +	* Liste der in der Band-Datei stehenden Dateien ins Ablaufprotokoll ausgeben.
	= -STD-	Liste der in der Band-Datei stehenden Dateien in die Standard-Protokoll-Datei ausgeben.
	= datei	Name der Datei für die Liste aller in der Band-Datei stehenden Dateien.
LOESCHEN	= -	* Daten in der PROTOKOLL-Datei nicht löschen.
	= +	Daten in der PROTOKOLL-Datei zuerst löschen.
CODE	= ASCII	* Band-Datei ist im ASCII-Code beschrieben.
	= EBCDIC	Band-Datei ist im EBCDIC-Code beschrieben.
GERAET	= -	* Band-Datei ist angemeldet.
	= -STD-	Pfad für die Band-Datei ist durch die System-Variable TUSTEP_MT1 vorgegeben.
	= name	Name der System-Variablen, die den Pfad für die Band-Datei enthält.

Unter Windows kann auch direkt der Laufwerksbuchstabe angegeben werden, falls der Pfad für die Band-Datei keinen Verzeichnisnamen enthält.

## Leistung

Mit diesem Kommando kann eine Liste mit den Namen der Dateien ausgegeben werden, die in einer Band-Datei stehen. Zusätzlich wird ausgegeben, wann die einzelnen Dateien zuletzt geändert wurden, bevor sie mit #MBAUSGABE oder #RETTE in die Band-Datei geschrieben wurden, und wieviele Sätze sie jeweils enthalten (bei TUSTEP-Dateien) bzw. wieviele Kilobytes sie umfassen (bei Fremd-Dateien und Band-Dateien).

## Hinweise

Die Band-Datei muss entweder angemeldet sein oder es muss zur Spezifikation `GERAET` eine System-Variable angegeben werden, die den Pfad zur Band-Datei enthält, angegeben werden; im letzteren Fall ist die Angabe eines Projekts zusammen mit dem Namen der Band-Datei nicht erlaubt.

Das Format der Dateilisten, die mit den Kommandos `#MBAUSGABE` oder `#MBKOPIERE` erstellt werden, entspricht dem Format der mit dem Kommando `#MBINFORMIERE` erstellten Dateilisten.

## Beispiel

Unter Windows soll eine Liste aller in der Band-Datei `sammel` stehenden Dateien ins Ablaufprotokoll ausgegeben werden. Die Band-Datei `sammel` befindet sich im Hauptverzeichnis (root directory) des Laufwerks `F` (dies kann z. B. auch ein USB-Speicherstick sein).

```
#MBI, sammel, GER=F
```



# Kopieren einer Band-Datei

#MBKOPIERE

## Kommando

#MBKOPIERE

QBAND	= name	Name der QUELL-Band-Datei.
QNUMMER	= -STD-	* Jeweils die Datei mit dem entsprechenden Namen kopieren, deren Daten als letzte geändert wurden, bevor sie in die QUELL-Band-Datei geschrieben wurde.
	= n	Jeweils die erste Datei mit dem entsprechenden Namen kopieren, wobei die Suche mit der n-ten Datei in der QUELL-Band-Datei beginnt.
	= n-m	Jeweils die erste Datei mit dem entsprechenden Namen kopieren, wobei die Suche mit der n-ten Datei in der QUELL-Band-Datei beginnt und mit der m-ten Datei endet.
		Es können auch mehrere Bereiche angegeben werden; sie müssen durch Apostroph getrennt sein. Die Bereiche müssen jedoch alle in aufsteigender Reihenfolge angegeben werden und dürfen sich nicht überschneiden.
QLABEL	= -STD-	* Alle Dateien ab der zur Spezifikation QNUMMER angegebenen Dateiposition kopieren.
		Ist zur Spezifikation QNUMMER der Wert -STD- angegeben (Voreinstellung), so wird von gleichnamigen Dateien jeweils nur die Datei kopiert, deren Daten als letzte geändert wurden, bevor sie in die Band-Datei geschrieben wurde.
	= name	Datei mit dem hier angegebenen Namen kopieren.
		Es können auch mehrere Dateinamen angegeben werden; sie müssen durch Apostroph getrennt sein.
QCODE	= ASCII	* QUELL-Band-Datei ist im ASCII-Code beschrieben.
	= EBCDIC	QUELL-Band-Datei ist im EBCDIC-Code beschrieben.
QGERAET	= -	* QUELL-Band-Datei ist angemeldet.
	= -STD-	Pfad für die QUELL-Band-Datei ist durch die System-Variable TUSTEP_MT1 vorgegeben.

	= name	Name der System-Variablen, die den Pfad für die QUELL-Band-Datei enthält.  Unter Windows kann auch direkt der Laufwerksbuchstabe angegeben werden, falls der Pfad für die Band-Datei keinen Verzeichnisnamen enthält.
ZBAND	= name	Name der ZIEL-Band-Datei.
ZNUMMER	= -STD-	* Dateien hinter die letzte Datei in die ZIEL-Band-Datei ausgeben.
	= n	Dateien ab der n-ten Dateiposition in die ZIEL-Band-Datei ausgeben.
ZLABEL	= -STD-	* Die Dateien mit den Namen in die ZIEL-Band-Datei schreiben, die sie in der QUELL-Band-Datei hatten.
	= name	Die Dateien mit den hier angegebenen Namen in die ZIEL-Band-Datei schreiben.  Die Anzahl der Namen muss mit der Anzahl der Dateien, die zur Spezifikation QLABEL angegeben sind, übereinstimmen. Ist zur Spezifikation QLABEL der Wert -STD- angegeben, muss zu ZLABEL ebenfalls -STD- angegeben werden.
ZCODE	= ASCII	* ZIEL-Band-Datei im ASCII-Code beschreiben.
	= EBCDIC	ZIEL-Band-Datei im EBCDIC-Code beschreiben.
ZGERAET	= -	* ZIEL-Band-Datei ist angemeldet.
	= -STD-	Pfad für die ZIEL-Band-Datei ist durch die System-Variable TUSTEP_MT2 vorgegeben.
	= name	Name der System-Variablen, die den Pfad für die ZIEL-Band-Datei enthält.  Unter Windows kann auch direkt der Laufwerksbuchstabe angegeben werden, falls der Pfad für die Band-Datei keinen Verzeichnisnamen enthält.
DENSITY	=	(nicht mehr definiert)
PROTOKOLL	= +	Liste aller in der ZIEL-Band-Datei stehenden Dateien ins Ablaufprotokoll ausgeben.
	= -STD-	Liste aller in der ZIEL-Band-Datei stehenden Dateien in die Standard-Protokoll-Datei ausgeben.
	= datei	Name der Datei für die Liste aller in der ZIEL-Band-Datei stehenden Dateien.
	= -	* Keine Liste der Dateien ausgeben.
LOESCHEN	= -	* Daten in der PROTOKOLL-Datei nicht löschen.

	= +	Daten in der PROTOKOLL-Datei zuerst löschen.
POSITIV	= -	* Keine Positiv-Auswahl von Dateien, deren Name einem vorgegebenen Muster entspricht.
	= . . .	Nur solche Dateien kopieren, deren Name entweder explizit zur Spezifikation QLABEL angegeben ist, oder deren Name mindestens einem der angegebenen Muster entspricht.
NEGATIV	= -	* Keine Negativ-Auswahl von Dateien, deren Name einem vorgegebenen Muster entspricht.
	= . . .	Nur solche Dateien kopieren, deren Name entweder explizit zur Spezifikation QLABEL angegeben ist, oder deren Name keinem der angegebenen Muster entspricht.

## Leistung

Mit diesem Kommando können Dateien von einer Band-Datei in eine andere Band-Datei kopiert werden. Dabei ist eine Auswahl der zu kopierenden Dateien möglich.

Die Auswahl kann mit der Spezifikation QNUMMER auf Grund der Position einer Datei innerhalb der Band-Datei, mit der Spezifikation QLABEL auf Grund des Namens einer Datei oder mit den Spezifikationen POSITIV und NEGATIV auf Grund eines Musters, das der Name der Datei enthält bzw. nicht enthält, getroffen werden.

Soll die Auswahl über ein Muster erfolgen, so können zur Spezifikation POSITIV Muster angegeben werden, von denen mindestens eines dem Dateinamen entsprechen muss, damit die Datei kopiert wird; zur Spezifikation NEGATIV können Muster angegeben werden, von denen keines dem Dateinamen entsprechen darf, damit die Datei kopiert wird. Die möglichen Angaben sind beim Kommando #LISTE auf Seite 178 beschrieben.

Zusätzlich kann nach dem Kopieren der Dateien eine Liste mit den Namen der Dateien ausgegeben werden, die dann in der ZIEL-Band-Datei stehen. Diese Liste entspricht derjenigen, die mit dem Kommando #MBINFORMIERE (siehe Seite 191) erstellt werden kann.

## Warnung

Werden die Dateien nicht hinter die letzte Datei in die Band-Datei geschrieben, so gehen sowohl die Datei, die überschrieben wird, als auch alle nachfolgenden Dateien, die schon in der Band-Datei standen, verloren.

Mit den Spezifikationsangaben QNUMMER=-STD- und QLABEL=-STD- (jeweils Voreinstellung) werden nicht alle Dateien kopiert, falls gleichnamige Dateien in der QUELL-Band-Datei vorhanden sind. Von den Dateien mit gleichem Namen wird in diesem Fall jeweils nur die kopiert, deren Daten als letzte geändert wurden, bevor sie in die QUELL-Band-Datei geschrieben wurde. Sollen alle Dateien kopiert werden, muss QNUMMER=1 angegeben werden.

## Hinweis

Die Band-Dateien müssen entweder angemeldet sein oder es muss zur Spezifikation QGERAET bzw. ZGERAET eine System-Variable angegeben werden, die den Pfad zur Band-Datei enthält; im letzteren Fall ist die Angabe eines Projekts zusammen mit dem Namen der Band-Datei nicht erlaubt und es wird auch kein Projektname ergänzt.

## Beispiele

Unter Windows sollen alle Dateien von der Band-Datei `sammel` in die Band-Datei `sicherung` kopiert werden; falls jedoch die Band-Datei `sammel` Dateien mit gleichem Namen enthält, soll jeweils nur die Datei kopiert werden, deren Daten als letzte geändert wurde, bevor sie in die Band-Datei geschrieben wurde. Die Band-Datei `sammel` befinde sich im Hauptverzeichnis (root directory) des Laufwerks `D`, die Band-Datei `sicherung` im Hauptverzeichnis des Laufwerks `E`.

```
#MBK, QBAND=sammel, QGER=D, ZBAND=sicherung, ZGER=E
```

Unter Windows soll die Datei `alt` von der Band-Datei `eins` in die Band-Datei `zwei` kopiert werden und dort den Namen `neu` erhalten. Falls die Band-Datei `eins` mehrere Dateien mit dem Namen `alt` enthält, soll die Datei kopiert werden, deren Daten als letzte geändert wurden, bevor sie in die Band-Datei geschrieben wurde. Sowohl die Band-Datei `eins` als auch die Band-Datei `zwei` befinde sich im Hauptverzeichnis (root directory) des Laufwerks `D`.

```
#MBK, QBAND=eins, , alt, QGER=D, ZBAND=zwei, , neu, ZGER=D
```

# Labeln einer Band-Datei

#MBLABEL

## Kommando

#MBLABEL

BAND	= name	Name der Band-Datei.
DENSITY	=	(nicht mehr definiert)
CODE	= ASCII	* Band-Datei im ASCII-Code labeln.
	= EBCDIC	Band-Datei im EBCDIC-Code labeln.
GERAET	= -	* Band-Datei ist angemeldet.
	= -STD-	Pfad für die Band-Datei ist durch die System-Variable TUSTEP_MT1 vorgegeben.
	= name	Name der System-Variablen, die den Pfad für die Band-Datei enthält.  Unter Windows kann auch direkt der Laufwerksbuchstabe angegeben werden, falls der Pfad für die Band-Datei keinen Verzeichnisnamen enthält.
OWNER	= name	Band-Datei einrichten.  Der Name wird als Bestandteil des Labels in die Band-Datei geschrieben.
	= -STD-	* Wie name, wobei als Name der mit dem Kommando #DEFINIERE (siehe Seite 132) eingestellte Name eingesetzt wird.
	= +	Alle Dateien in der Band-Datei löschen.
	= -	Band-Datei löschen.

## Leistung

Mit diesem Kommando können Band-Dateien eingerichtet und gelöscht werden.

## Hinweis

Die Band-Datei muss entweder angemeldet sein oder es muss zur Spezifikation GERAET eine System-Variable angegeben werden, die den Pfad zur Band-Datei enthält; im letzteren Fall ist die Angabe eines Projekts zusammen mit dem Namen der Band-Datei nicht erlaubt und es wird auch kein Projektname ergänzt.

## Beispiele

Unter Windows soll eine Band-Datei mit dem Namen `sammel` im Hauptverzeichnis (root directory) des Laufwerks `F` (dies kann z. B. auch ein USB-Speicher-Stick sein) eingerichtet werden; dabei soll in den Label der Name `peter` eingetragen werden.

```
#MBL, sammel, GER=F, peter
```

Unter Windows soll die Band-Datei mit dem Namen `sammel` im Hauptverzeichnis (root directory) des Laufwerks `F` gelöscht werden.

```
#MBL, sammel, GER=F, -
```

# Testen einer Band-Datei

#MBTEST

## Kommando

#MBTEST

BAND	= name	Name der Band-Datei.
NUMMER	= -STD-	* Jeweils die Datei mit dem entsprechenden Namen testen, deren Daten als letzte geändert wurden, bevor sie in die Band-Datei geschrieben wurde.
	= n	Jeweils die erste Datei mit dem entsprechenden Namen testen, wobei die Suche mit der n-ten Datei in der Band-Datei beginnt.
DATEI	= datei	Name der Datei, deren Daten mit den Daten der in der Band-Datei zu testenden Datei verglichen werden sollen.  Es können auch mehrere Dateinamen angegeben werden; sie müssen durch Apostroph getrennt sein.
LABEL	= -STD-	* Die zu testenden Dateien haben in der Band-Datei die zur Spezifikation DATEI angegebenen Namen.
	= name	Die zu testenden Dateien haben in der Band-Datei die hier angegebenen Namen.  Die Anzahl der Namen muss mit der Anzahl der Dateien, die zur Spezifikation DATEI angegeben sind, übereinstimmen.
PROTOKOLL	= +	* Fehlermeldung ins Ablaufprotokoll ausgeben (und Fehlerflag setzen), falls sich die zu vergleichenden Daten unterscheiden.
	= -	Keine Fehlermeldung ausgeben (nur Fehlerflag setzen), falls sich die zu vergleichenden Daten unterscheiden.
CODE	= ASCII	* Band-Datei ist im ASCII-Code beschrieben.
	= EBCDIC	Band-Datei ist im EBCDIC-Code beschrieben.
GERAET	= -STD-	* Pfad für die Band-Datei ist durch die System-Variable TUSTEP_MT1 vorgegeben.
	= name	Name der System-Variablen, die den Pfad für die Band-Datei enthält.

Unter Windows kann auch direkt der Laufwerksbuchstabe angegeben werden, falls der Pfad für die Band-Datei keinen Verzeichnisnamen enthält.

## Leistung

Mit diesem Kommando kann geprüft werden, ob die Daten von Dateien mit denen in einer Band-Datei übereinstimmen.

## Hinweis

Die Band-Datei muss entweder angemeldet sein oder es muss zur Spezifikation `GERAET` eine System-Variable angegeben werden, die den Pfad zur Band-Datei enthält; im letzteren Fall ist die Angabe eines Projekts zusammen mit dem Namen der Band-Datei nicht erlaubt und es wird auch kein Projektname ergänzt.

## Beispiel

Unter Windows soll geprüft werden, ob die Daten der Datei `e` schon in der aktuellen Fassung in der Datei `erfass` der Band-Datei `sammel` stehen. Wenn in der Band-Datei mehrere Dateien mit dem Namen `erfass` vorhanden sind, sollen die Daten der letzten Datei mit diesem Namen geprüft werden. Die Band-Datei `sammel` befindet sich im Hauptverzeichnis (root directory) des Laufwerks `F` (dies kann z. B. auch ein USB-Speicher-Stick sein).

```
#MBT, sammel, , e, erfass, GER=F
```



# Mischen von Daten/Dateien

#MISCHE

## Kommando

#MISCHE

QUELLE	= datei	Namen der Dateien mit den zu mischenden Daten; die einzelnen Dateinamen müssen durch Apostroph getrennt sein.
ZIEL	= datei	Name der Datei für die gemischten Daten.
	= -STD-	Die gemischten Daten in die Standard-Text-Datei ausgeben.
SORTIERFELD	= sf-S	Eingabedaten sind nach dem Sortierfeld <i>sf</i> aufsteigend (steigend) sortiert.
	= sf-F	Eingabedaten sind nach dem Sortierfeld <i>sf</i> absteigend (fallend) sortiert.
	= sf	Wie <i>sf-S</i> .
		Für <i>sf</i> ist eine der folgenden Angaben einzusetzen:
		0 Sortierung nach Satznummern.
		<i>n-m</i> Sortierfeld beginnt auf Zeichenposition <i>n</i> und endet mit der Zeichenposition <i>m</i> .
		<i>n+m</i> Sortierfeld beginnt auf Zeichenposition <i>n</i> und ist <i>m</i> Zeichen lang.
		Es können auch mehrere Sortierfelder angegeben werden; sie müssen durch Apostroph getrennt sein.
LOESCHEN	= -	* Daten in der ZIEL-Datei nicht löschen.
	= +	Daten in der ZIEL-Datei zuerst löschen.
TILGEN	= -	* Sätze unverändert in die ZIEL-Datei ausgeben.
	= n-m	Vor der Ausgabe in die ZIEL-Datei in jedem Satz die Zeichen von Position <i>n</i> bis Position <i>m</i> tilgen (eliminieren).
	= n+m	Vor der Ausgabe in die ZIEL-Datei in jedem Satz <i>m</i> Zeichen ab der Position <i>n</i> tilgen (eliminieren).
	= +	In die ZIEL-Datei nur die Daten aus der DATEN-Datei in der durch das Mischen erzeugten Reihenfolge ausgeben.

DATEN	= -	* Normalfall.
	= <code>datei</code>	Beim Mischen Daten in der entsprechenden Reihenfolge aus der angegebenen <code>DATEN</code> -Datei einlesen und in die <code>ZIEL</code> -Datei ausgeben.
	= <code>-STD-</code>	Beim Mischen Daten in der entsprechenden Reihenfolge aus der Standard-Daten-Datei einlesen und in die <code>ZIEL</code> -Datei ausgeben.

## Leistung

Mit diesem Kommando können Dateien, die sortierte Daten enthalten, gemischt werden.

Die Angaben zur Spezifikation `SORTIERFELD` müssen gleich sein wie die Angabe zu dieser Spezifikation beim Sortieren der zu mischenden Daten mit dem Kommando `#SORTIERE`.

Das Programm prüft nicht nach, ob die Daten tatsächlich nach dem angegebenen Sortierfeld sortiert sind. Das Sortierfeld wird nur verwendet, um festzustellen, welcher der von den verschiedenen Dateien eingelesenen Sätze als nächster auf die `ZIEL`-Datei ausgegeben werden soll.

Sind bei Sätzen, die aus verschiedenen `QUELL`-Dateien stammen, die Sortierfelder identisch, so ist für die Reihenfolge, in der die Sätze in die `ZIEL`-Datei ausgegeben werden, die Reihenfolge maßgebend, in der die entsprechenden Dateien zur Spezifikation `QUELLE` angegeben sind.

Die Angabe der Sortierfelder wird auf ihre Plausibilität geprüft. In seltenen Fällen kann es vorkommen, dass ungewöhnliche Sortierfeld-Angaben notwendig und richtig sind, aber nicht akzeptiert werden. In einem solchen Fall kann die Angabe zur Spezifikation `SORTIERFELD` mit einem Ausrufezeichen abgeschlossen und so die Überprüfung unterdrückt werden.

Mit der Spezifikation `TILGEN` können Sortierschlüssel eliminiert werden, die nach dem Mischen nicht mehr benötigt werden.

## Beispiele

Sortieren und Mischen von zwei Dateien, in deren Daten der Sortierschlüssel ab Spalte 17 steht und 20 Zeichen lang ist:

```
#SO, datei1, datei1, 17+20, +
#SO, datei2, datei2, 17+20, +
#MI, datei1'datei2, datei3, 17+20, +
```

# Modi merken/zurücksetzen

#MODI

## Kommando

#MODI

MODUS	= RETTEN	Aktuell eingestellte Modi merken.
	= HOLEN	Modi auf die zuletzt gemerkten einstellen.
PROTOKOLL	= -STD-	* Normalfall.
	= -	Keine Meldung ausgeben. Bei MODUS=RETTEN: Zusätzlich das Kommando #PROTOKOLL, - (siehe Seite 209) ausführen.
	= /	Keine Meldung im TUSTEP-Fenster, aber Meldung im Zweitprotokoll, falls es eingeschaltet ist. Bei MODUS=RETTEN: Zusätzlich das Kommando #PROTOKOLL, / (siehe Seite 209) ausführen.
	= +	Meldung ausgeben. Bei MODUS=RETTEN: Zusätzlich das Kommando #PROTOKOLL, + (siehe Seite 209) ausführen.

## Leistung

Mit diesem Kommando können die mit den nachfolgend angegebenen Kommandos eingestellten Modi gemerkt (MODUS=RETTEN) und (nachdem sie ggf. geändert wurden) wieder auf die gemerkten Werte zurückgesetzt werden (MODUS=HOLEN).

```
#FEHLERHALT, EIN/AUS/SIGNAL  
#PARAMETER, EIN/AUS/{ } / <> / NEU / ALT  
#PROTOKOLL, EIN/AUS / - / + / SYSTEM / PORTIONIERT / FREILAUFEND  
#WISCHEN, EIN/AUS
```

Werden die Modi wiederholt gerettet (ohne sie jeweils wieder zu holen), so werden sie in einem Stack gespeichert. Beim Holen werden jeweils die zuletzt geretteten Modi eingestellt und die entsprechenden Werte vom Stack gelöscht.

Dieses Kommando kann nur in Kommandofolgen verwendet werden. Ist eine Kommandofolge abgearbeitet und sind noch Modi im Stack gespeichert, so werden automatisch die zuerst geretteten Modi wieder eingestellt und alle Werte vom Stack gelöscht. Dies gilt insbesondere auch, wenn bei einem Fehlerhalt (siehe Kommando #FEHLERHALT Seite 158) die verbleibenden Kommandos einer Kommandofolge gelöscht und dadurch die Kommandofolge nicht vollständig ausgeführt wird.

**Beispiel**

In einer Kommandofolge sollen die Parameter der einzelnen Kommandos nicht protokolliert werden. Danach sollen die Parameter wieder protokolliert bzw. weiterhin nicht protokolliert werden, je nachdem wie der Modus vor der Ausführung der Kommandofolge eingestellt war.

```
#MODI, RETTEN
#PARAMETER, AUS
#KO, quelle, ziel, , +, *
...
*EOF
#MODI, HOLEN
```

# Normieren/Beenden von TUSTEP

#NORMIERE

## Kommando

#NORMIERE

BEENDEN	= -	* TUSTEP-Sitzung beenden und neu beginnen.
	= +	TUSTEP-Sitzung beenden.
WISCHEN	= +	Daten in den temporären Dateien überschreiben.
	= -	Daten in den temporären Dateien nicht überschreiben.

## Leistung

Mit diesem Kommando kann die TUSTEP-Sitzung beendet und ggf. automatisch wieder neu begonnen werden.

Beim Beenden der TUSTEP-Sitzung werden

- alle Standard-Dateien gelöscht,
- alle temporären Dateien (Scratch-Dateien) gelöscht,
- alle angemeldeten Dateien abgemeldet.

Ob die in temporären Dateien (Scratch-Dateien) stehenden Daten überschrieben (Datenschutz!) werden sollen, kann mit der Spezifikation `WISCHEN` angegeben werden. Wird nichts angegeben, so werden die Daten überschrieben, wenn nicht mit dem Kommando `#WISCHEN` (siehe Seite 250) ein anderer Modus eingestellt wurde.

Falls die TUSTEP-Sitzung wieder neu begonnen werden soll, wird TUSTEP neu initialisiert. Dies ist unter »Beginnen einer TUSTEP-Sitzung« auf Seite 87 beschrieben.

# Neu-Nummerieren von Verweisen

#NUMMERIERE

## Kommando

#NUMMERIERE

Für dieses Kommando gibt es eine eigene Beschreibung.

## Leistung

Mit diesem Programm können (laufende) Nummern und die dazugehörigen Verweise aktualisiert werden. Dazu können in einem Text an entsprechend gekennzeichneten Stellen laufende Nummern eingesetzt werden. Eine dort ggf. bereits vorhandene (alte) Nummer wird dabei ersetzt. Gleichzeitig können entsprechend gekennzeichnete Verweise, die sich auf die alten Nummern beziehen, aktualisiert werden.

Außerdem kann das Programm an entsprechend gekennzeichneten Stellen Verweise, die sich auf eine Seiten-Zeilen-Nummer beziehen, aktualisieren, nachdem sich die Seiten-Zeilen-Einteilung verändert hat.

# Parameter-Modi einstellen

#PARAMETER

## Kommando

#PARAMETER

MODUS	= AUS	Protokollierung der Parameter ausschalten.
	= EIN	Protokollierung der Parameter einschalten.
	= { }	Zur Kennzeichnung von Zeichen- und Stringgruppen, Häufigkeitsangaben, Verweisen usw. werden die geschweiften Klammern verwendet.
	= <>	Zur Kennzeichnung von Zeichen- und Stringgruppen, Häufigkeitsangaben, Verweisen usw. werden die spitzen Klammern verwendet.
	= NEU	»Neue« Konvention (s. u.) für Parameter einstellen (entspricht <>).
	= ALT	»Alte« Konvention (s. u.) für Parameter einstellen.

## Leistung

Mit diesem Kommando kann eingestellt werden,

- ob die Parameter im Ablaufprotokoll aufgelistet werden oder nicht.
- nach welcher Konvention die Parameter interpretieren werden.

Die mit diesem Kommando eingestellte Parameter-Konvention gilt für alle Kommandos, auch für den Editor. Sie gilt jedoch nicht für Makros; in Makros muss mit der `MODE`-Anweisung (siehe Seite 413 ff.) die im jeweiligen Makro gültige Konvention zur Interpretation von Zeichen-, und Stringgruppen sowie Such-, Austausch- und Recherchier-Tabellen eingestellt werden.

Wird zur Spezifikation `MODUS` nichts angegeben, so wird nur ausgegeben, ob die Parameter-Protokollierung ein- oder ausgeschaltet ist und nach welcher Konvention die Parameter interpretiert werden.

Beim Initialisieren einer TUSTEP-Sitzung wird die Parameter-Protokollierung ausgeschaltet und die Konvention »{ }« eingestellt.

## Hinweis

Parameter können nach drei verschiedenen Konventionen geschrieben werden:

Die »alte« Konvention ist die ursprüngliche in TUSTEP. Sie wurde konzipiert, als die

Lochkarte noch das Standardmedium für die Daten- und Programmeingabe war. Dabei musste berücksichtigt werden, dass auf Lochkarten üblicherweise Groß- und Kleinbuchstaben nicht unterschieden wurden und Umlaute nicht vorgesehen waren (MODUS=ALT).

Die »neue« Konvention ist auf neuere Eingabemedien abgestimmt, bei denen diese Einschränkungen nicht mehr bestehen. Zur Kennzeichnung von Zeichen- und Stringgruppen, Häufigkeitsangaben, Verweisen usw. werden dabei die spitzen Klammern verwendet (MODUS=NEU, gleichbedeutend mit MODUS=<>).

Die Mustererkennung (pattern matching) wurden nach und nach erweitert und verbessert. Dabei wurde bei den neu hinzukommenden Codierungen immer darauf geachtet, dass auch zuvor erstellte Parameter noch die gleiche Wirkung wie seither hatten. Das hat dazu geführt, dass die Codierungen unübersichtlich wurden und nicht mehr leicht zu merken und auch nicht mehr leicht zu lesen sind. Darüber hinaus ist die direkte Verwendung von spitzen Klammern nicht möglich, wenn Parameter mit einem XML-Editor geschrieben oder geändert werden sollen. Deshalb wurden die Angaben zur Mustererkennung neu konzipiert und damit eine dritte Konvention für die Parameter geschaffen, bei der die Codierungen leichter zu merken und auch leichter zu lesen sind. Zur Kennzeichnung von Zeichen- und Stringgruppen, Häufigkeitsangaben, Verweisen usw. werden dabei die geschweiften Klammern verwendet (MODUS={ }).

Für neue Projekte empfiehlt es sich, diese dritte Konvention (MODUS={ }) zu verwenden. Für den Editor und die Programme mit Parametern ist diese dritte Konvention voreingestellt. Bei Projekten, die diese Konvention nicht verwenden, muss die verwendete Konvention (i.d.R. MODUS=<>) explizit eingestellt werden.

Einzelheiten zur »dritten« Konvention sind im Kapitel »{}-Parameter« ab Seite 701 beschrieben, die zur »alten« und »neuen« Konvention sind im Kapitel »<>-Parameter« ab Seite 735 beschrieben.



## Zweitprotokoll ein/ausschalten u. a.

#PROTOKOLL

### Kommando

#PROTOKOLL

MODUS	= STARTE	Zweitprotokoll einschalten und löschen.
	= EIN	Zweitprotokoll einschalten.
	= AUS	Zweitprotokoll ausschalten.
	= LOESCHE	Zweitprotokoll löschen.
	= NEUE_SEITE	Im Zweitprotokoll eine neue Seite beginnen, falls es eingeschaltet ist; im TUSTEP-Fenster den gerade angezeigten Teil des Ablaufprotokolls löschen.
	= ZEIGE	Zweitprotokoll ausschalten und im TUSTEP-Fenster als Ablaufprotokoll anzeigen.
	= EDIERE	Zweitprotokoll ausschalten und edieren.
	= KOPIERE	Zweitprotokoll ausschalten und in die zur Spezifikation DATEI angegebene Datei kopieren.
	= DRUCKE	Zweitprotokoll ausschalten und auf dem zur Spezifikation GERAET angegebenen Drucker ausgeben.
	= -	Protokoll-Ausgabe unterdrücken.
	= /	Protokoll-Ausgabe nur im TUSTEP-Fenster unterdrücken, nicht aber im Zweitprotokoll, falls es eingeschaltet ist.
	= +	Protokoll-Ausgabe nicht unterdrücken.
	= FREILAUFEND	Protokoll-Ausgabe ins TUSTEP-Fenster soll fortlaufend, d. h. ohne Unterbrechung erfolgen.
	= PORTIONIERT	Protokoll-Ausgabe ins TUSTEP-Fenster soll in Portionen erfolgen, wobei die nächste Portion jeweils nach einer Bestätigung (mit der Return-Taste) ausgegeben wird.
	= SYSTEM	Protokoll-Ausgabe ins TUSTEP-Fenster soll in der Weise erfolgen, wie es in dem jeweiligen Betriebssystem üblich ist.
TYP	= ...	Bei MODUS=DRUCKE: Typ des Druckers, auf dem ausgedruckt werden soll.

			Mit dem Kommando #LISTE, DRUCKERTYPEN (siehe Seite 173) können die möglichen Druckertypen aufgelistet werden.
GERAET	= -STD-	*	Bei MODUS=DRUCKE: Die System-Variable TUSTEP_PRN enthält den Namen des Druckers, auf dem ausgedruckt werden soll; falls diese System-Variable nicht definiert ist, wird der Druckername vom Betriebssystem erfragt.
	= name		Bei MODUS=DRUCKE: Name einer System-Variablen, die den Namen des Druckers enthält, auf dem ausgedruckt werden soll.
			Der Name des Druckers kann eventuell direkt angegeben werden; siehe dazu unter »Druckernamen«.
	= +		Ausgabe ins Ablaufprotokoll; bei TYP=WIN-xx in ein eigenes Fenster.
ANZAHL	= 1	*	Bei MODUS=DRUCKE: Zweitprotokoll 1-mal ausdrucken.
	= n		Bei MODUS=DRUCKE: Zweitprotokoll n-mal ausdrucken.
VORSPANN	= -	*	Kein Vorspann.
	= *		Vorspann folgt auf das #PROTOKOLL-Kommando und ist mit *EOF abgeschlossen.
	= datei		Name der Datei mit dem Vorspann.
SEITEN	= -	*	Zweitprotokoll vollständig ausdrucken.
	= n / n-m		Nur die Seite n bzw. nur die Seiten n bis m ausdrucken.
			Es können auch mehrere Seiten/Bereiche angegeben werden; sie müssen durch Apostroph getrennt sein.
UMSCHLAG	= -STD-	*	Ob von TUSTEP bei Druckausgaben ein Umschlag (Deck- und Endblatt mit Namen) ergänzt wird, ist vom jeweiligen Rechner und vom Druckertyp abhängig.
	= -		Von TUSTEP keinen Umschlag ergänzen. Diese Angabe wird ignoriert, falls dies vom Systemverwalter für bestimmte Druckertypen nicht zugelassen ist.
	= +		Von TUSTEP einen Umschlag ergänzen.
DATEI	= -	*	Normalfall.
	= datei		Bei MODUS=KOPIERE: Name der TUSTEP-Datei, in die das Zweitprotokoll kopiert werden soll.
			Bei MODUS=DRUCKE: Name der Fremd-Datei, in die die Codes zur Steuerung des Druckers ausgegeben

		werden sollen. Es können bis zu zehn Dateinamen angegeben werden; sie müssen durch Apostroph getrennt sein.
LOESCHEN	= -	* Daten in den zur Spezifikation DATEI angegebenen Dateien nicht löschen.
	= +	Daten in den zur Spezifikation DATEI angegebenen Dateien zuerst löschen.
OPTIONEN	= ...	* Bei MODUS=DRUCKE: Angaben zur Modifikation der erzeugten SteuerCodes für den Drucker.
ZUSATZ	= ...	Bei MODUS=DRUCKE: Name einer System-Variablen, die zusätzliche Angaben für das Betriebssystem enthält (vgl. System-Variable TUSTEP_LPR Seite 76). Welche Angaben sinnvoll oder im Einzelfall notwendig sind, ist beim Systemverwalter zu erfragen.
PORTION	= -STD-	* Bei MODUS=DRUCKE: Ausdruck bei Zeilendruckern in Portionen zu 500 Seiten, bei Matrix- und Laserdruckern in Portionen zu 200 Seiten unterteilen.
	= n	Bei MODUS=DRUCKE: Ausdruck in Portionen zu n Seiten unterteilen.
TITEL	= -	* Normalfall
	= titel	Unter Windows: Bei TYP=WIN-xx und GERAET=+ angegebenen Titel in die Kopfzeile des Fensters einfügen, in dem die Daten angezeigt werden. Falls schon Fenster mit dem gleichen Titel vorhanden sind, diese schließen.

## Leistung

Mit diesem Kommando kann eingestellt werden, dass zusätzlich zum Ablaufprotokoll ein Zweitprotokoll geführt wird, in das die Meldungen und Protokollausgaben von TUSTEP eingetragen werden.

Ist das Zweitprotokoll eingeschaltet, so werden die Protokollausgaben, die durch die Spezifikationsangabe PROTOKOLL=+ der TUSTEP-Kommandos erzeugt werden, sowie die Liste der Parameter nur ins Zweitprotokoll eingetragen; die Start- und Endmeldungen sowie eventuelle Fehlermeldungen werden sowohl ins Ablaufprotokoll als auch ins Zweitprotokoll eingetragen.

Wird das Kommando ohne Spezifikationsangaben aufgerufen, so wird nur angezeigt, ob das Zweitprotokoll ein- oder ausgeschaltet ist und ob die Protokoll-Ausgabe ins TUSTEP-Fenster auf SYSTEM, PORTIONIERT oder FREILAUFEND eingestellt ist.

Nach dem Initialisieren einer TUSTEP-Sitzung ist das Zweitprotokoll ausgeschaltet und das Ablaufprotokoll auf PORTIONIERT eingestellt.

Das Zweitprotokoll kann mit diesem Kommando

- in das TUSTEP-Fenster ausgegeben werden,
- mit dem Editor bearbeitet werden,
- in eine Datei kopiert werden,
- gelöscht werden,
- ausgedruckt werden.

Sollen nur bestimmte »Seiten« ausgedruckt werden, so können diese mit der Spezifikation SEITEN ausgewählt werden. Die Auswahl erfolgt anhand der Satznummern in der Datei mit dem Zweitprotokoll ohne Rücksicht auf tatsächliche Druckseiten. Eine Auswahl ist in der Regel nur sinnvoll, wenn zuvor mit dem Editor das Zweitprotokoll angeschaut wird und so die Seitenzahlen festgestellt werden.

Kann auf den Drucker, auf dem das Zweitprotokoll ausgedruckt werden soll, nicht direkt ausgegeben werden (z. B. weil er an einen anderen PC angeschlossen ist), so kann zur Spezifikation DATEI eine permanente Fremd-Datei (TYP=FDF) angegeben werden, in die die Codes ausgegeben werden, die für die Steuerung eines Druckers vom angegebenen Typ notwendig sind. Diese Fremd-Datei kann dann mit Kommandos, die im jeweiligen Betriebssystem zur Verfügung stehen, zum gewünschten Drucker übertragen werden. Werden zur Spezifikation DATEI mehrere Dateien angegeben, so werden jeweils so viele Seiten in eine Datei geschrieben, wie zur Spezifikation PORTION angegeben wird, und dann in der nächsten Datei weitergeschrieben.

Bei der Ausführung von ausgetesteten Kommandofolgen kann es sinnvoll sein, das Ablaufprotokoll auf ein Minimum zu reduzieren. Nach dem Einstellen des Modus auf »-« werden nur noch die mit »#+« beginnenden Kommentare, Fehlermeldungen und die zu Eingabeaufforderungen gehörenden Meldungen ausgegeben. Dieser Modus kann mit der Einstellung auf »+« wieder aufgehoben werden. Falls im Dialog am Ende einer Kommandofolge dieser Modus noch nicht aufgehoben ist, wird er automatisch aufgehoben, damit die neu eingegebenen Kommandos wieder normal protokolliert werden.

Im Dialog kann mit diesem Kommando auch die Art der Ausgabe ins TUSTEP-Fenster sowie der Eingabe von der Tastatur eingestellt werden:

MODUS = SYSTEM

Die Ausgabe und die Eingabe erfolgen auf die im jeweiligen Betriebssystem übliche Art. Zum Eingeben, Korrigieren und Wiederholen der Kommandozeile stehen die Funktionstasten und Steuerbefehle nicht oder nur teilweise zur Verfügung.

MODUS = FREILAUFEND

Die Ausgabe erfolgt fortlaufend ohne Unterbrechung bis zur nächsten Eingabeaufforderung. Zum Eingeben, Korrigieren und Wiederholen der Kommandozeile können die Funktionstasten und Steuerbefehle verwendet werden (vgl. Seite 113).

MODUS = PORTIONIERT

Wie MODUS=FREILAUFEND, jedoch wird die Ausgabe angehalten, nachdem ein Fenster voll ausgegeben wurde, und auf eine Bestätigung zur Ausgabe der nächsten Portion gewartet. Dabei kann auch die Größe der nächsten Portion durch Eingabe der entsprechenden Zeilenzahl bestimmt werden. Wird die Escape-Taste oder die Tasten-

kombination `Ctrl+D` bzw. `Strg+D` gedrückt, so erfolgt die Ausgabe fortlaufend ohne Unterbrechung bis zur nächsten Eingabeaufforderung.

Nach dem Initialisieren einer TUSTEP-Sitzung erfolgt die Ausgabe des Ablaufprotokolls portionsweise.

## Druckernamen

Wenn der Name des Druckers, auf dem ausgedruckt werden soll, aus 1 bis 12 Zeichen (Buchstaben, Ziffern und »\_«) besteht, mit einem Buchstaben beginnt, nicht mit »\_« endet und nicht mit dem Namen einer System-Variablen übereinstimmt, kann er direkt zur Spezifikation `GERAET` angegeben werden. Erfüllt der Name diese Bedingungen nicht, muss eine System-Variable definiert werden, die den Druckernamen enthält, und diese zur Spezifikation `GERAET` angegeben werden. Die Namen der möglichen Drucker sind beim Systemverwalter zu erfragen.

## Optionen

Zum Drucken auf HP-kompatiblen Druckern und PostScript-Druckern, die das Papier beidseitig bedrucken können, sind folgende Optionen möglich:

- `SMPLX` Nur Vorderseite bedrucken
- `D_LANG` Vorder- und Rückseite so bedrucken, dass über die lange Papierkante geblättert werden kann (meist für Hochformat)
- `D_KURZ` Vorder- und Rückseite so bedrucken, dass über die kurze Papierkante geblättert werden kann (meist für Querformat)

Zum Drucken auf HP-kompatiblen Druckern, die über zwei Papierschächte verfügen, sind folgende Optionen möglich:

- `OBEN` Papier aus dem oberen Schacht
- `UNTEN` Papier aus dem unteren Schacht

Zum Drucken auf EPSON-kompatiblen Druckern sind folgende Optionen möglich:

- `BD` bidirektionaler Druck
- `UD` unidirektionaler Druck (Voreinstellung)
- `H11` Druck auf 11 Zoll hohes Papier
- `H12` Druck auf 12 Zoll hohes Papier
- `LQ` Schönschreibmodus (Voreinstellung)
- `DRAFT` Schnellschreibmodus

## Anmerkung

Wird das Zweitprotokoll mit `MODUS=DRUCKE` ausgegeben, so wird damit das bis dahin erstellte Zweitprotokoll nicht gleichzeitig gelöscht. Es muss ggf. explizit mit `MODUS=LOESCHE` gelöscht werden.

## Beispiel

Ausführen der Kommandofolge, die in der Datei `work.cmd` steht, und das dabei erzeugte Ablaufprotokoll anschließend auf einem PostScript-Drucker zweiseitig ausdrucken und zusätzlich in der Datei `work.log` speichern:

```
#PR, START
#TUE, work.cmd
#PR, AUS
#PR, DRUCKE, PS-Q2, pd
#PR, KOPIERE, DATEI=work.log
```

Um sicherzustellen, dass kein eventuell schon vorher erstelltes Protokoll mitausgedruckt wird, wird das Zweitprotokoll zuerst gelöscht. Das Ausschalten des Zweitprotokolls könnte im Beispiel entfallen, da das Zweitprotokoll beim anschließenden Drucken automatisch ausgeschaltet würde. Im Beispiel wird angenommen, dass in der System-Variablen `pd` der vom Betriebssystem verwendete Name des PostScript-Druckers steht, auf dem ausgedruckt werden soll.

# Register nach dem Sortieren aufbereiten

**#RAUFBEREITE**

## Kommando

#RAUFBEREITE

Für dieses Kommando gibt es eine eigene Beschreibung.

## Leistung

Mit diesem Programm können Registereinträge bzw. Texteinheiten zu einem Register (z. B. Wortformenregister oder KWIC-Index) oder einem Verzeichnis (z. B. Bibliographie) zusammengefasst und aufbereitet werden.

Die Registereinträge bzw. Texteinheiten können zuvor mit dem Kommando #RVORBEREITE oder #SVORBEREITE erzeugt und zum Sortieren vorbereitet und dann mit dem Kommando #SORTIERE sortiert werden. Es können auch andere Daten verarbeitet werden, falls ein Eingabesatz jeweils eine Texteinheit enthält.

Das Programm bietet u. a. folgende Möglichkeiten: Ergänzung von Steuerzeichen und Kennzeichen, beliebiges Format der Druckausgabe, lebende Kolumnentitel, Zwischenüberschriften, Auswahl von bestimmten Registereinträgen bzw. Texteinheiten, hierarchische Gliederung in Haupt- und Untereinträge, Errechnen von absoluten und relativen Häufigkeiten.

# Retten von Daten

#RETTE

## Kommando

#RETTE

QUELLE	= datei	Name der Datei, aus der die Daten gerettet (kopiert) werden sollen.
	= -STD-	Die zu rettenden (zu kopierenden) Daten stehen in der Standard-Text-Datei.
ZIEL	= datei	Name der Datei für die geretteten (zu kopierenden) Daten.
	= -STD-	Die geretteten (zu kopierenden) Daten in die Standard-Text-Datei ausgegeben.
MODUS	= -STD-	* Falls ZIEL-Datei eine TUSTEP-Datei ist: Alle Sätze von der QUELL-Datei in die ZIEL-Datei kopieren. Dann nachfragen, ob die Daten von der ZIEL-Datei wieder in die QUELL-Datei zurückkopiert werden sollen.  Falls ZIEL-Datei eine Band-Datei ist: Daten der QUELL-Datei als Datei mit dem zur Spezifikation QUELLE angegebenen Namen ans Ende der Band-Datei kopieren.
	= -	Wie -STD-, jedoch nicht nachfragen und die Daten nicht zurückkopieren.
	= +	Wie -STD-, jedoch die Daten ohne nachzufragen wieder zurückkopieren.
	= name	Falls ZIEL-Datei eine Segment-Datei ist: Daten der QUELL-Datei als Segment mit dem angegebenen Namen in die ZIEL-Datei kopieren.  Falls ZIEL-Datei eine Band-Datei ist: Daten der QUELL-Datei als Datei mit dem angegebenen Namen ans Ende der Band-Datei kopieren.
	= n	Falls ZIEL-Datei eine TUSTEP-Datei ist: Nur die Sätze der n-ten aufsteigenden Satznummernfolge von der QUELL-Datei in die ZIEL-Datei kopieren.  Falls ZIEL-Datei eine Band-Datei ist: Daten der QUELL-Datei als Datei mit dem zur Spezifikation



		QUELLE angegebenen Namen als n-te Datei in die Band-Datei kopieren.
	= n:name	Falls ZIEL-Datei eine Band-Datei ist: Daten der QUELL-Datei als Datei mit dem angegebenen Namen als n-te Datei in die Band-Datei kopieren.
	= -;-	Aus den Daten der QUELL-Datei eine (normale) Segment-Datei erstellen; sonst wie MODUS=-.
	= +;-	Aus den Daten der QUELL-Datei eine Mega-Segment-Datei erstellen; sonst wie MODUS=-.
	= -;+	Aus den Daten der QUELL-Datei eine (normale) Segment-Datei erstellen; sonst wie MODUS=+.
	= +;+	Aus den Daten der QUELL-Datei eine Mega-Segment-Datei erstellen; sonst wie MODUS=+.
LOESCHEN	= -	* Daten in der ZIEL-Datei nicht löschen.
	= +	Bei MODUS=name: Ein evtl. mit diesem Namen vorhandenes Segment überschreiben. Bei anderen Modi: Daten in der ZIEL-Datei zuerst löschen.
WISCHEN	= +	Nur wenn zurückkopiert wird: Daten in der QUELL-Datei überschreiben bevor der von der Datei seither belegte Plattenplatz freigegeben wird.
	= -	Nur wenn zurückkopiert wird: Daten in der QUELL-Datei nicht überschreiben bevor der von der Datei seither belegte Plattenplatz freigegeben wird.

## Leistung

Mit diesem Kommando können

- Daten (auch) aus einer unvollständigen, nicht abgeschlossenen Datei in eine andere Datei kopiert werden (vgl. »Retten von Dateien« Seite 40).
- TUSTEP-Dateien, die auf einem anderen Rechnertyp erstellt und von diesem binär übertragen wurden, auf den neuen Rechnertyp umgestellt werden.
- die Sätze einer aufsteigenden Satznummernfolge in eine andere Datei kopiert werden. Eine aufsteigende Satznummernfolge endet jeweils vor bzw. beginnt jeweils mit einer Satznummer, die gleich oder kleiner als die unmittelbar vorangehende ist.
- Daten aus einer Datei in eine andere kopiert werden, wobei ggf. die beim Edieren entstandenen Lücken beseitigt und die Sätze in ihrer logischen Reihenfolge angeordnet werden (vgl. »Reorganisieren von Dateien« Seite 40).
- Daten aus einer Datei in eine Segment-Datei (oder eine leere Datei) kopiert werden, wobei die Daten als Segment abgespeichert werden.

- Daten aus einer Datei (kann auch eine Fremd-Datei sein) in eine Band-Datei kopiert werden, wobei die Daten als Datei mit dem Namen der QUELL-Datei bzw. dem zur Spezifikation MODUS angegebenen Namen in der Band-Datei abgespeichert werden.
- aus (normalen) Segment-Dateien Mega-Segment-Dateien und umgekehrt erstellt werden; dabei werden alle Sätze neu durchnummeriert.

Falls die Daten von der ZIEL-Datei wieder in die QUELL-Datei zurückkopiert werden sollen, wird zuvor der von der QUELL-Datei seither belegte Plattenplatz freigegeben, damit die Datei anschließend nur noch so viel Platz belegt, wie für die Daten erforderlich ist.

Falls auf Grund der Spezifikation MODUS nachgefragt wird, ob die Daten wieder zurückkopiert werden sollen, sind zwei Antworten vorgesehen, die auch abgekürzt werden können:

- 1) JA Die Daten zurückkopieren.
- 2) NEIN Die Daten nicht zurückkopieren.

Falls mit dem Kommando #PROTOKOLL (siehe Seite 209) der Protokoll-Modus nicht auf SYSTEM eingestellt wurde, kann statt NEIN einzugeben auch die Escape-Taste gedrückt werden.

## Warnung

Ist die ZIEL-Datei eine Band-Datei und wird die Datei nicht hinter die letzte Datei in die Band-Datei geschrieben, so gehen sowohl die Datei, die überschrieben wird, als auch alle nachfolgenden Dateien, die schon in der Band-Datei standen, verloren.

## Beispiele

Unter Windows eine Datei mit dem Namen `text` von der Festplatte in die Datei mit dem Namen `arbeit` auf den USB-Speicher-Stick mit dem Laufwerksbuchstaben `E` kopieren:

```
#DA, -*arbeit, SEQ-P, TR=E
#RE, text, -*arbeit
#AB, -*arbeit
```

Nach umfangreichen Änderungen mit dem Editor in der Datei mit dem Namen `erfass`, diese Datei reorganisieren :

```
#RE, erfass, -STD-, +, +
```

Es wurden Daten in eine Datei mit dem Namen `hilf` kopiert. Diese Datei enthielt versehentlich schon Daten, die dabei nicht gelöscht wurden. Die Satznummern der alten und neuen Daten sind alle aufsteigend und die Nummer des letzten Satzes der alten Daten ist größer als die Nummer des ersten Satzes der neuen Daten. Die Datei kann deshalb nicht mit Editor bearbeitet werden. Nun die alten und neuen Daten getrennt in die Dateien mit den Namen `alt` und `neu` kopieren:

```
#RE, hilf, alt, 1
#RE, hilf, neu, 2
```

Die Daten der Datei mit dem Namen `hilf` in das Segment `test` der Datei mit dem Namen `sammel` kopieren; falls diese Datei schon ein Segment mit dem Namen `test` enthält, dieses überschreiben:

```
#RE, hilf, sammel, test, +
```

Die Daten der Segment-Datei `macro` neu nummerieren:

```
#RE, macro, -STD-, -;+, +
```

Die Daten der Segment-Dateien `macro1` und `macro2` in einer Segment-Datei `macro` zusammenfassen:

```
#RE, macro1, macro, -, +
#RE, macro2, macro, -, -
#RE, macro, -STD-, -;+, +
```

# Register zum Sortieren vorbereiten

#RVORBEREITE

## Kommando

#RVORBEREITE

Für dieses Kommando gibt es eine eigene Beschreibung.

## Leistung

Mit diesem Programm können

- Texte in sortierfähige und für Register geeignete Einträge zerlegt werden (z. B. für ein Wortformenregister oder einen KWIC-Index),
- aus Texten entsprechend gekennzeichnete Textteile als Registerinträge übernommen werden (z. B. für ein Autorenregister, Personenregister, Sachregister oder ein Ortsregister).

# Satzherstellung

**#SATZ**

## Kommando

#SATZ

Für dieses Kommando gibt es eine eigene Beschreibung.

## Leistung

Mit diesem Programm können Texte typographisch aufbereitet werden. Die Aufteilung des Textes auf Zeilen und Seiten (einschl. Silbentrennung, Randausgleich, Marginalien, Fußnoten, lebenden Kolumnentiteln usw.) erfolgt automatisch; sie kann über Anweisungen, die im Text enthalten sind, gesteuert werden.

Zur Kontrolle und Weiterverarbeitung der Ergebnisse dieses Programms gibt es TUS-TEP-interne Makros. Mit ihnen kann die Satzausgabe auf PostScript-Druckern oder (für professionellen Satz) auf PostScript-Belichtern erfolgen. Bezüglich der Verfügbarkeit von Fonts bzw. entsprechender Lizenzen für diese Geräte ist ggf. eine Abstimmung mit deren Betreibern erforderlich.

# Ausgeben von Signaltönen

#SIGNAL

## Kommando

#SIGNAL

TON = -STD- \* Den mit dem Kommando #DEFINIERE eingestellten Signalton ausgeben.  
= tonfolge Angegebene Tonfolge ausgeben.  
= \* Die auf das #SIGNAL-Kommando folgende (und mit \*EOF abgeschlossenen) Tonfolge ausgeben.

## Leistung

Mit diesem Kommando kann unter Windows eine Tonfolge ausgegeben werden. Dies kann z. B. verwendet werden, um bei längeren Kommandofolgen das Erreichen von Stellen, an denen wieder eine Tastatureingabe erforderlich ist, zu signalisieren.

Für jeden Ton sind folgende Angaben möglich:

oktav vorzeichen tonhöhe tonlänge

Von diesen Angaben sind nur `tonhöhe` und `tonlänge` obligat.

Mögliche Angaben für `oktav`:

keine Angabe: Ton in der Oktav mit dem Kammerton a (440 Hz)  
- Ton in der Oktav unterhalb der Oktav mit dem Kammerton a (440 Hz)  
+ Ton in der Oktav oberhalb der Oktav mit dem Kammerton a (440 Hz)

Mögliche Angaben für `vorzeichen`:

keine Angabe: Ton nicht erhöhen/erniedrigen  
# Ton um einen Halbton erhöhen  
% Ton um einen Halbton erniedrigen

Wird die Tonfolge im Kommando (also nicht nach dem Kommando) angegeben, so muss `^#` statt `#` angegeben werden, damit das `#` nicht als Beginn des nächsten Kommandos interpretiert wird.

Mögliche Angaben für `tonhöhe`:

einer der Buchstaben c, d, e, f, g, a, h, b  
oder der Buchstabe p für eine Pause

Mögliche Angaben für `tonlänge`:

eine der Ziffern 1, 2, 4, 8 oder 6 für ganze, halbe, viertel, achter und sechzehntel Tonlänge.

Eine ganze Tonlänge ist ungefähr 1,6 Sekunden lang. Jede Tonlänge kann noch um die Hälfte verlängert werden, indem hinter der Ziffer noch ein Punkt angegeben wird.

## Beispiel

```
#SIGNAL, d8'^#f8'a8'+c4
```

ist gleichbedeutend mit

```
#SIGNAL, *  
d8 #f8 a8 +c4  
*EOF
```

# Sortieren von Daten/Dateien

#SORTIERE

## Kommando

#SORTIERE

QUELLE	= datei	Name der Datei mit den zu sortierenden Daten.
	= -STD-	Die zu sortierenden Daten stehen in der Standard-Text-Datei.
ZIEL	= datei	Name der Datei für die sortierten Daten.
	= -STD-	Die sortierten Daten in die Standard-Text-Datei ausgeben.
SORTIERFELD	= sf-S	Nach dem Sortierfeld <i>sf</i> aufsteigend (steigend) sortieren.
	= sf-F	Nach dem Sortierfeld <i>sf</i> absteigend (fallend) sortieren.
	= sf	Wie <i>sf-S</i> .
		Für <i>sf</i> ist eine der folgenden Angaben einzusetzen:
		0 Sortierung nach Satznummern.
		<i>n-m</i> Sortierfeld beginnt auf Zeichenposition <i>n</i> und endet mit der Zeichenposition <i>m</i> .
		<i>n+m</i> Sortierfeld beginnt auf Zeichenposition <i>n</i> und ist <i>m</i> Zeichen lang.
		Es können auch mehrere Sortierfelder angegeben werden; sie müssen durch Apostroph getrennt sein.
LOESCHEN	= -	* Daten in der ZIEL-Datei nicht löschen.
	= +	Daten in der ZIEL-Datei zuerst löschen.
TILGEN	= -	* Sätze nach dem Sortieren unverändert in die ZIEL-Datei ausgeben.
	= <i>n-m</i>	Vor der Ausgabe in die ZIEL-Datei in jedem Satz die Zeichen von Position <i>n</i> bis Position <i>m</i> tilgen (eliminieren).
	= <i>n+m</i>	Vor der Ausgabe in die ZIEL-Datei in jedem Satz <i>m</i> Zeichen ab der Position <i>n</i> tilgen (eliminieren).
	= +	In die ZIEL-Datei nur die Daten aus der DATEN-Datei in der sortierten Reihenfolge ausgeben.



DATEN	= -	* Normalfall.
	= <code>datei</code>	Nach dem Sortieren Daten in der entsprechenden Reihenfolge aus der angegebenen <code>DATEN</code> -Datei einlesen und in die <code>ZIEL</code> -Datei ausgeben.
	= <code>-STD-</code>	Nach dem Sortieren Daten in der entsprechenden Reihenfolge aus der Standard-Daten-Datei einlesen und in die <code>ZIEL</code> -Datei ausgeben.
WISCHEN	= +	Daten in den zum Sortieren benötigten Hilfsdateien vor dem Löschen dieser Dateien überschreiben.
	= -	Daten in den zum Sortieren benötigten Hilfsdateien vor dem Löschen dieser Dateien nicht überschreiben.
TRAEGER	= <code>name</code>	Name der System-Variablen, die den Pfad für die zum Sortieren intern notwendigen Hilfsdateien enthält.  Unter Windows kann auch direkt der Laufwerksbuchstabe angegeben werden, falls der Pfad keinen Verzeichnisnamen enthält.
	= <code>-STD-</code>	* Zum Sortieren intern notwendige Hilfsdateien auf dem Standard-Träger für temporäre Dateien einrichten; dieser ist durch die System-Variable <code>TUSTEP_SCR</code> vorgegeben.

## Leistung

Mit diesem Kommando können Dateien sortiert werden. Die Reihenfolge der Sätze wird dabei durch den Inhalt der angegebenen Sortierfelder bestimmt. Ist bei zwei oder mehreren Sätzen der Inhalt eines Sortierfeldes gleich, so bestimmt das jeweils nächste Sortierfeld die Reihenfolge; ist kein weiteres Sortierfeld angegeben, so bleibt die Reihenfolge dieser Sätze unverändert.

Die zu sortierenden Daten müssen vorher mit dem Kommando `#SVORBEREITE` oder `#RVORBEREITE` zum Sortieren vorbereitet werden; nur in Ausnahmefällen, z. B. wenn nur nach der Satznummer sortiert werden soll, ist dies nicht notwendig. Der Aufbau der Sätze, die sich beim Vorbereiten ergeben, ist bei diesen Programmen beschrieben. Daraus und aus der Reihenfolge, in der die einzelnen Felder der Sätze beim Sortieren berücksichtigt werden sollen, ergibt sich die Angabe zur Spezifikation `SORTIERFELD`.

In vielen Fällen entspricht das Sortierfeld den mit den Kommandos `#SVORBEREITE` bzw. `#RVORBEREITE` erzeugten Sortierschlüsseln. Die Länge dieses Sortierfeldes ergibt sich dann aus der Gesamtlänge der einzelnen Sortierschlüssel, die mit dem Parameter `SSL` festgelegt wurde; es beginnt auf Position 1, falls die Sätze keine Referenz am Satzanfang enthalten, andernfalls auf der ersten Position nach der Referenz; dies ist die Position 17, wenn über den Parameter `IRL` nichts anderes bestimmt wurde.

Die Angabe der Sortierfelder wird auf ihre Plausibilität geprüft. In seltenen Fällen kann es vorkommen, dass ungewöhnliche Sortierfeld-Angaben notwendig und rich-

tig sind, aber nicht akzeptiert werden. In einem solchen Fall kann die Angabe zur Spezifikation `SORTIERFELD` mit einem Ausrufezeichen abgeschlossen und so die Überprüfung unterdrückt werden.

Sollen die Daten nach dem Sortieren auf derselben Datei stehen wie vorher, so kann bei diesem Kommando (im Gegensatz zu allen anderen Kommandos) zur Spezifikation `QUELLE` und `ZIEL` jeweils dieselbe Datei angegeben werden. Zur Spezifikation `LOESCHEN` muss in diesem Fall »+« angegeben werden, damit die unsortierten Daten vor der Ausgabe der sortierten Daten gelöscht werden.

Mit der Spezifikation `TILGEN` können Sortierschlüssel eliminiert werden, die nach dem Sortieren nicht mehr benötigt werden.

Zum Sortieren werden Hilfsdateien benötigt, die von `TUSTEP` automatisch eingerichtet und wieder gelöscht werden. Ob vor dem Löschen dieser Dateien die darin enthaltenen Daten überschrieben (Datenschutz!) werden sollen, kann mit der Spezifikation `WISCHEN` angegeben werden. Wird nichts angegeben, so werden die Daten überschrieben, wenn nicht mit dem Kommando `#WISCHEN` (siehe Seite 250) ein anderer Modus eingestellt wurde.

## Beispiel

Sortieren einer Datei `liste`, wobei beim Vorbereiten der Daten zwei Sortierschlüssel zu je 20 Zeichen erzeugt werden. Diese beiden Sortierschlüssel bilden zusammen ein Sortierfeld, das 40 Zeichen lang ist. Nach dem Sortieren werden die Sortierschlüssel nicht mehr benötigt und deshalb eliminiert. Danach werden die Daten neu durchnummeriert und in die Datei `ergebnis` geschrieben.

```
#SV, liste, -STD-, -, +, *
...
SSL          20 20
...
*EOF

#SO, -STD-, -STD-, 1+40, +, 1+40

#KO, -STD-, ergebnis, +, +
```

# Sortierschlüssel prüfen

#SPRUEFE

## Kommando

#SPRUEFE

Für dieses Kommando gibt es eine eigene Beschreibung.

## Leistung

Mit diesem Programm kann nach dem Sortieren geprüft werden, ob die einzelnen Sortierschlüssel lang genug sind, um die gewünschte Reihenfolge der einzelnen Sätze zu erreichen.

# Durchsuchen von Texten

#SUCHE

## Kommando

#SUCHE

NAME	= -STD-	* TUSTEP-Beschreibung durchsuchen.
	= name	Name des zu durchsuchenden Textes.
TRAEGER	= name	Name der System-Variablen, die den Pfad für die Text- und Index-Dateien enthält.
		Unter Windows kann auch direkt der Laufwerksbuchstabe angegeben werden, falls der Pfad keinen Verzeichnisnamen enthält.
	= -STD-	* Text- und Index-Dateien befinden sich auf dem Standard-Träger für permanente Dateien; dieser ist durch die System-Variable TUSTEP_DSK vorgegeben.

## Leistung

Mit dem Kommando #SUCHE können alle Vorkommen von bestimmten Wörtern oder Wortkombinationen in einem Text aufgesucht und angezeigt werden.

Um mit dem Kommando #SUCHE bestimmte Textstellen aufsuchen zu können, muss der Text zuvor mit dem Makro \*SUCHE aufbereitet werden. Dabei wird der Name des Textes festgelegt und eine Kopie des Textes zusammen mit Index-Dateien unter diesem Namen auf einem Datenträger gespeichert.

Wird das Kommando ohne Spezifikationsangaben aufgerufen, so kann in den Texten der TUSTEP-Beschreibung recherchiert werden.

## Hinweis

Das Kapitel »SUCHE« (siehe ab Seite 667) enthält eine ausführliche Beschreibung dieses Kommandos und ein Beispiel zur Aufbereitung der Daten.

## Daten zum Sortieren vorbereiten

**#SVORBEREITE**

### Kommando

#SVORBEREITE

Für dieses Kommando gibt es eine eigene Beschreibung.

### Leistung

Mit diesem Programm können Texteinheiten (bestehend aus einem oder mehreren Eingabesätzen, auch Korrekturanweisungen) zum Sortieren vorbereitet werden. Dabei kann angegeben werden, nach welchen Kriterien sortiert werden soll.

# Testen/Vergleichen einer Datei

#TESTE

## Kommando

#TESTE

QUELLE	=	datei	Name der Datei mit den zu vergleichenden Daten.
TEST	=	datei	Name der Datei mit den zu vergleichenden Daten.
MODUS	=	+	* Daten und Satznummern der QUELL-Datei mit denen in der TEST-Datei vergleichen.
	=	-	Daten der QUELL-Datei mit denen in der TEST-Datei vergleichen.
	=	name	Name des Segments in der QUELL-Datei, dessen Daten mit denen in der TEST-Datei verglichen werden sollen.
PROTOKOLL	=	+	* Fehlermeldung ins Ablaufprotokoll ausgeben (und Fehlerflag setzen), falls sich die zu vergleichenden Daten unterscheiden.
	=	-	Keine Fehlermeldung ausgeben (nur Fehlerflag setzen), falls sich die zu vergleichenden Daten unterscheiden.

## Leistung

Mit diesem Kommando kann geprüft werden, ob die Daten einer Datei mit denen einer anderen Datei übereinstimmen.

## Beispiele

Prüfen, ob die Daten der Datei mit dem Namen `erfass` geändert wurden, seitdem sie in die Datei mit dem Namen `erfass.sik` kopiert wurden:

```
#TEST, erfass.sik, erfass
```

Prüfen, ob die Daten der Datei `e` und die Daten des Segments `xy` in der Datei `makro` identisch sind:

```
#TEST, makro, e, xy
```

# Titel einer Datei definieren/übertragen

#TITEL

## Kommando

#TITEL

QUELLE	= *	Titel folgt auf das #TITEL-Kommando und ist mit *EOF abgeschlossen.
	= datei	Name der Datei, deren Titel in die ZIEL-Datei übertragen wird.
	= -STD-	* Gemarkten Titel in die ZIEL-Datei übertragen.
ZIEL	= datei	Name der Datei, deren Titel definiert wird.
	= -STD-	* Titel der QUELL-Datei (intern) merken.

## Leistung

Mit diesem Kommando kann

- ein Dateititel neu definiert werden,
- ein Dateititel in eine andere Datei übertragen werden,
- ein Dateititel gemerkt werden,
- der gemerkte Dateititel in eine Datei übertragen werden

## Beispiele

Eine Datei soll einen neuen Dateititel erhalten:

```
#TI, *, datei
Dies ist der Text, der als Titel
ohne Zeilenwechsel eingetragen wird.
*eof
```

In einer Kommandofolge werden Daten einer Datei sortiert und in eine andere Datei geschrieben. Dabei soll auch der Titel der Datei übernommen werden:

```
#SV, qdat, -STD-, -, +, *  
...  
*eof  
...  
...  
#KO, -STD-, zdat, , +, *  
...  
*eof  
#TI, qdat, zdat
```

In einer Kommandofolge wird eine Datei mit neuen Daten beschrieben. Dabei soll der Titel dieser Datei erhalten bleiben:

```
...  
...  
#TI, daten, -STD-  
#KO, hilfsdatei, daten, , +, *  
...  
*eof  
#TI, -STD-, daten
```



# Ausführen einer Kommandofolge

#TUE

## Kommando

#TUE

KOMMANDO	= datei	Name der Datei (Programm- oder Segment-Datei) mit den auszuführenden Kommandos.
	= -STD-	Die auszuführenden Kommandos stehen in der Standard-Editor-Datei.
	= *	Die auszuführenden Kommandos folgen auf das #TUE-Kommando und sind mit *EOF abgeschlossen. In diesem Fall ist zur Spezifikation BEREICH nur die Angabe »-« zugelassen.
BEREICH	= -	* Gesamte Kommandofolge ausführen, die zur Spezifikation KOMMANDO angegeben ist.
	= pos	Nur die Kommandofolge ausführen, die in der zur Spezifikation KOMMANDO angegebenen Programm-Datei ab der Satzposition pos steht.
	= pos1-pos2	Nur die Kommandofolge ausführen, die in der zur Spezifikation KOMMANDO angegebenen Programm-Datei von der Satzposition pos1 bis zur Satzposition pos2 (je einschließlich) steht.
	= name	Nur die Kommandofolge ausführen, die in der zur Spezifikation KOMMANDO angegebenen Segment-Datei im Segment mit dem hier angegebenen Namen steht.
	= name-pos	Nur die Kommandofolge ausführen, die im Segment mit dem hier angegebenen Namen ab der Satzposition pos steht.
	= name-pos1-pos2	Nur die Kommandofolge ausführen, die im Segment mit dem hier angegebenen Namen von der Satzposition pos1 bis zur Satzposition pos2 (je einschließlich) steht.
PARAMETER	= -	* Keine Parameter.
	= param	Parameter, die an entsprechend gekennzeichneten Stellen in die Kommandofolge eingesetzt werden sollen.

SCHLEIFE	= n	Kommandofolge n-mal ausführen; der Schleifenzähler, der an entsprechend gekennzeichneten Stellen in die Kommandofolge eingesetzt wird, soll die Werte 1, 2, 3, ... bis n annehmen.
	= n-m	Kommandofolge m-n+1-mal ausführen; der Schleifenzähler soll dabei die Werte n bis m annehmen. Falls n Null ist, beim ersten Durchlauf, bei dem der Schleifenzähler also den Wert 0 hat, an den entsprechend gekennzeichneten Stellen statt der »0« eine leere Zeichenfolge einsetzen.
	= 0-0	* Kommandofolge 1-mal ausführen; dabei an den entsprechend gekennzeichneten Stellen für den Schleifenzähler eine leere Zeichenfolge einsetzen.
	= datei	Kommandofolge für jeden Satz in der angegebenen Datei ausführen; dabei wird an den entsprechend gekennzeichneten Stellen der Satzinhalt statt eines Schleifenzählers eingesetzt.
KENNZEICHEN	= -STD-	* Entspricht der Angabe »-«, falls PARAMETER=- und SCHLEIFE=0-0 (d. i. jeweils die Voreinstellung), und der Angabe »?« in allen anderen Fällen.
	= -	In der Kommandofolge sind keine Stellen gekennzeichnet, an denen Parameter oder Schleifenzähler eingesetzt werden sollen.
	= x	Als Kennzeichen für Stellen, an denen Parameter oder der Schleifenzähler in die Kommandofolge eingesetzt werden sollen, soll das hier angegebene Zeichen gelten.
PROTOKOLL	= -	* Keine zusätzliche Protokollierung der Kommandofolge.
	= +	Kommandofolge ins Ablaufprotokoll ausgeben.
	= datei	Name der Datei für das Protokoll der Kommandofolge.
AUSFUEHRUNG	= +	* Kommandofolge ausführen.
	= -	Kommandofolge nicht ausführen (sinnvoll, wenn die Kommandofolge zu Testzwecken nur protokolliert werden soll).

## Leistung

Mit diesem Kommando kann eine (in der Regel in einer Datei stehende) TUSTEP-Kommandofolge ein- oder mehrmals ausgeführt werden. Dabei können die Kommandos mit Parametern und/oder einer laufenden Nummer (= Schleifenzähler) modifiziert werden.

Soll nur ein bestimmter Teil einer Kommandofolge ausgeführt werden, so kann dieser durch eine Angabe zur Spezifikation `BEREICH` ausgewählt werden. Die verschiedenen Auswahlmöglichkeiten sind oben beschrieben. »pos« steht dabei für eine Satznummer, deren Schreibweise dem Programmmodus (vgl. Seite 33) entspricht. »name« steht für den Namen eines Segments in einer Segment-Datei.

Zur Spezifikation `PARAMETER` können bis zu neun Parameter jeweils durch Apostroph getrennt angegeben werden. Der 1. Parameter wird an allen Stellen in die Kommandofolge eingesetzt, die mit »?1« (bei »?« als Kennzeichen) gekennzeichnet sind; der 2., 3. usw. wird an den mit »?2«, »?3« usw. gekennzeichneten Stellen eingesetzt. Werden zur Spezifikation `PARAMETER` weniger Parameter angegeben, als in der Kommandofolge vorgesehen sind, so wird für die fehlenden Parameter eine leere Zeichenfolge eingesetzt.

Soll die Kommandofolge mit anderen Parametern wiederholt werden, so können mehrere Parameterfolgen (mit bis zu neun Parametern) durch Strichpunkt getrennt hintereinander angegeben werden. Werden bei einer Wiederholung weniger Parameter angegeben, als in der Kommandofolge vorgesehen sind, so wird für die fehlenden Parameter die gleiche Zeichenfolge eingesetzt wie beim vorangehenden Mal. Das gleiche gilt, wenn für einen Parameter eine leere Zeichenfolge (d. h. nur der abschließende Apostroph) angegeben wird. Die angegebenen Parameter gelten also jeweils zugleich als Voreinstellung für die nachfolgenden Wiederholungen.

Die Kommandos können auch durch eine laufende Nummer modifiziert werden. Sie entspricht dem Wert des Schleifenzählers und wird an den Stellen in die Kommandofolge eingesetzt, die mit »?0« (bei »?« als Kennzeichen) gekennzeichnet sind. Die Kommandofolge wird für jeden Wert des Schleifenzählers ausgeführt. Welche Werte er durchlaufen soll, kann zur Spezifikation `SCHLEIFE` angegeben werden. Wird die Kommandofolge auch auf Grund der Angaben zur Spezifikation `PARAMETER` wiederholt, so wird jede dieser Wiederholungen mit der zur Spezifikation `SCHLEIFE` angegebenen Häufigkeit entsprechend oft ausgeführt.

An Stelle eines Schleifenzählers kann an den mit »?0« (bei »?« als Kennzeichen) gekennzeichneten Stellen auch eine Zeichenfolge eingesetzt werden. Die Zeichenfolgen müssen in der zur Spezifikation `SCHLEIFE` angegebenen Datei stehen, wobei jeder Satz eine Zeichenfolge enthält. Führende und abschließende Leerzeichen werden ignoriert. Die Kommandofolge wird für jede Zeichenfolge ausgeführt. Wird die Kommandofolge auch auf Grund der Angaben zur Spezifikation `PARAMETER` wiederholt, so wird jede dieser Wiederholungen mit jedem Satz der zur Spezifikation `SCHLEIFE` angegebenen Datei ausgeführt.

Zur Spezifikation `KENNZEICHEN` kann als Kennzeichen für die Stellen, an denen Parameter bzw. Schleifenzähler eingesetzt werden sollen, jedes Zeichen außer Minuszeichen, Nummernzeichen, Komma, Gleichheitszeichen, Apostroph und Anführungszeichen angegeben werden. Das angegebene Zeichen kennzeichnet jedoch nur dann (und immer dann) eine solche Stelle in der Kommandofolge, wenn unmittelbar dahinter eine Ziffer folgt. Kennzeichen und Ziffer werden in jedem Fall (ggf. durch eine leere Zeichenfolge) ersetzt, auch wenn zur Spezifikation `PARAMETER` und/oder `SCHLEIFE` nichts angegeben wird. Folgt auf ein Kennzeichen in der Kommandofolge keine Ziffer, so bleibt dieses Zeichen unverändert.

## Hinweise

Soll eine Kommandofolge z. B. für alle Dateien ausgeführt werden, deren Dateinamen eine bestimmte Zeichenfolge enthalten bzw. nicht enthalten, so können diese Dateinamen mit dem Kommando #LISTE (siehe Seite 173) in eine Datei geschrieben werden; diese Datei kann dann, nachdem ggf. die darin stehenden Dateinamen mit dem Editor kontrolliert und korrigiert wurden, zur Spezifikation SCHLEIFE angegeben werden.

Falls eine Kommandofolge nicht jeweils unverändert ausgeführt werden soll, ist es in der Regel sinnvoll, nicht mehr das Kommando #TUE, sondern Makros (siehe Kapitel »Makros« ab Seite 403) zu verwenden. Mit ihnen kann eine Kommandofolge in vielfältiger Weise modifiziert werden, nicht nur auf Grund der Angaben beim Aufruf des Makros, sondern z. B. auch in Abhängigkeit von den eingegebenen Antworten auf die im Makro enthaltenen Fragen.

# Umwandeln/Verschlüsseln von Daten/Dateien

#UMWANDLE

## Kommando

#UMWANDLE

QUELLE	= datei	Name der Datei mit den umzuwandelnden Daten.
	= -STD-	Die umzuwandelnden Daten stehen in der Standard-Text-Datei.
	= *	Die umzuwandelnden Daten folgen auf das #UMWANDLE-Kommando und sind mit *EOF abgeschlossen.
ZIEL	= datei	Name der Datei für die umgewandelten Daten.
	= -STD-	Die umgewandelten Daten in die Standard-Text-Datei ausgeben.
MODUS	= -STD-	* Entspricht der Angabe
		+1 falls die QUELL-Datei eine Fremd-Datei und die ZIEL-Datei eine TUSTEP-Datei ist und zur Spezifikation CODE nicht BINAER angegeben ist.
		-1 falls die QUELL-Datei eine TUSTEP-Datei und die ZIEL-Datei eine Fremd-Datei ist und zur Spezifikation CODE nicht BINAER angegeben ist.
		0 in allen anderen Fällen.
	= ...	Daten im angegebenen Modus umwandeln.
		Die möglichen Modi und ihre Wirkung sind unten im Abschnitt »Modi« beschrieben.
LOESCHEN	= -	* Daten in der ZIEL- und in der PROTOKOLL-Datei nicht löschen.
	= +	Daten in der ZIEL- und in der PROTOKOLL-Datei zuerst löschen.
SCHLUESSEL	= -	* Daten nicht ver- oder entschlüsseln.
	= datei	Name der Datei mit dem Schlüssel zum Ver- bzw. Entschlüsseln.
	= *	Der Schlüssel zum Ver- bzw. Entschlüsseln folgt auf das #UMWANDLE-Kommando (ggf. vor den Daten für die Spezifikation QUELLE) und ist mit *EOF abgeschlossen.

CODE	= -	* Keine zusätzliche Umcodierung. Bei den Modi TX, XT, TK, KT, BX und XB darf zu dieser Spezifikation nur »-« angegeben werden. Bei allen anderen Modi gilt folgendes: Ist die QUELL-Datei eine Fremd-Datei und die ZIEL-Datei eine TUSTEP-Datei oder umgekehrt, so muss der Code für die Daten der Fremd-Datei angegeben werden.
	= ASCII	Daten dem internationalen ASCII-Code (siehe Tabelle Seite 815) entsprechend umcodieren.
	= ASCII-EBCDIC	Daten von ASCII nach EBCDIC (vgl. Code-Tabellen Seite 840) umcodieren.
	= EBCDIC	Daten dem internationalen EBCDIC-Code (vgl. Code-Tabellen Seite 838) entsprechend umcodieren.
	= EBCDIC-ASCII	Daten von EBCDIC nach ASCII (vgl. Code-Tabellen Seite 840) umcodieren.
	= IBMPC	Daten dem IBM-PC-Zeichensatz (siehe Tabelle Seite 817) entsprechend umcodieren.
	= ANSI	Wie CP1252.
	= CPnnn	Daten der Code-Page nnn (nnn = 437, 850, 852, 1250, 1252, 10000, 10029) entsprechend umcodieren (siehe Tabellen ab Seite 818).
	= ISO8859	Daten dem Zeichensatz 1 der ISO-Norm 8859 (siehe Tabelle Seite 826) entsprechend umcodieren.
	= LATINn	Daten dem Code Latin n (n = 1 bis 10) entsprechend umcodieren (siehe Tabellen ab Seite 828).
	= LINUX	Daten dem Linux-Zeichensatz (siehe Tabelle Seite 827) entsprechend umcodieren.
	= DECMCS	Daten dem »DEC Multinational Character Set« (siehe Tabelle Seite 825) entsprechend umcodieren.
	= BINAER	Keine zusätzliche Umcodierung.
	= UNICODE	Wie UTF16.
	= UTF16	Daten den UTF-16-Konventionen entsprechend umcodieren.
	= UTF8	Daten den UTF-8-Konventionen entsprechend umcodieren.
	= xx:yy	Zeichen mit dem hexadezimalen Code xx in Zeichen mit dem hexadezimalen Code yy umcodieren. Mehrere Code-Paare können durch Apostroph ge-

		trennt angegeben werden. Außerdem kann jeder der zuvor angegebenen Spezifikationswerte durch solche Code-Paare ergänzt werden, jedoch nicht bei UTF16 und UTF8.
NL	= WIN	In Fremd-Dateien CR LF (= 0D0A) als Trennzeichen zwischen Datensätzen verwenden.
	= DOS	Wie WIN.
	= UNIX	In Fremd-Dateien LF (= 0A) als Trennzeichen zwischen Datensätzen verwenden.
	= -STD-	* Wie WIN, wenn der Anfang der QUELL-Datei Codes enthält, die der Angabe WIN entsprechen; wie UNIX, wenn der Anfang der QUELL-Datei Codes enthält, die der Angabe UNIX entsprechen; sonst: unter Windows wie WIN, unter Linux wie UNIX.
	= MIX	Beim Umwandeln einer Fremd- in eine TUSTEP-Datei LF (= 0A) und CR LF (= 0D0A) als Trennzeichen zwischen Datensätzen interpretieren.
	= ALL	Beim Umwandeln einer Fremd- in eine TUSTEP-Datei CR (= 0D), LF (= 0A) und CR LF (= 0D0A) als Trennzeichen zwischen Datensätzen interpretieren.
	= xx	In Fremd-Dateien das Zeichen mit dem hexadezimalen Code xx als Trennzeichen zwischen Datensätzen verwenden.
	= LS	Bei CODE=UTFn: In Fremd-Dateien die Codierung für »Line Separator« (= 2028) als Trennzeichen zwischen Datensätzen verwenden.
	= PS	Bei CODE=UTFn: In Fremd-Dateien die Codierung für »Paragraph Separator« (= 2029) als Trennzeichen zwischen Datensätzen verwenden.
	= nnn	Falls nnn eine mindestens 3-stellige Dezimalzahl (ggf. mit führenden Nullen) ist:  Beim Umwandeln einer Fremd- in eine TUSTEP-Datei nach jeweils nnn Zeichen einen neuen Datensatz beginnen.  Beim Umwandeln einer TUSTEP- in eine Fremd-Datei alle Datensätze mit Leerzeichen auf nnn Zeichen auffüllen.
SATZLAENGE	= -STD-	* Normalfall.
	= n	Beim Umwandeln in eine TUSTEP-Datei Datensätze mit mehr als n Zeichen bei Leerzeichen in mehrere

Datensätze mit maximal  $n$  Zeichen aufteilen; jedoch möglichst nur bei Leerzeichen außerhalb von Tags (d. h. außerhalb von »<<« und »>>«) aufteilen.

Falls zur Spezifikation KENNUNG eine Kennung angegeben ist, dürfen die Datensätze auch unmittelbar vor einem Anfangs-Tag und unmittelbar nach einem Ende-Tag aufgeteilt werden.

=  $n-m$

Wie  $n$ ; falls jedoch keine entsprechende Stelle zum Aufteilen des Satzes gefunden wird, dürfen die Datensätze bis zu  $m$  Zeichen lang werden.

KENNUNG = -STD-

\* Normalfall.

= . . .

Wenn beim Umwandeln einer Fremd- in eine TUSTEP-Datei Datensätze aufgeteilt werden, die angegebene Kennung vor den Stellen in die Datensätze einfügen, an denen sie aufgeteilt werden.

Wenn beim Umwandeln einer TUSTEP- in eine Fremd-Datei ein Datensatz mit der angegebenen Kennung endet, diese Kennung entfernen und den nachfolgenden Datensatz anhängen.

Als Kennung darf eine beliebige Zeichenfolge angegeben werden; sie darf jedoch keine Leerzeichen, Minuszeichen, Nummernzeichen, Kommata, Gleichheitszeichen, Apostrophe oder Anführungszeichen enthalten.

NUMMERIERUNG = -STD-

\* Falls als QUELL-Datei eine TUSTEP-Datei angegeben ist, Nummerierung der Sätze beibehalten; andernfalls Sätze im Textmodus nummerieren.

=  $n$

Sätze neu nummerieren.

Der Zahlenwert der letzten drei Ziffern von  $n$  wird jeweils auf die Unterscheidungsnummer aufaddiert. Falls  $n$  mehr als dreistellig ist, wird der Zahlenwert der übrigen Ziffern auf die Zeilennummer aufaddiert.

SEITENNUMMER = -

\* Normalfall.

= . . .

Wenn beim Umwandeln einer Fremd- in eine TUSTEP-Datei ein Datensatz nur die angegebene Kennung und eine Seitennummer enthält, diese Seitennummer für die Satznummerierung übernehmen und den Datensatz nicht in die ZIEL-Datei ausgeben.

Wenn beim Umwandeln einer TUSTEP- in eine Fremd-Datei sich die Seitennummer der Satznummer ändert, einen zusätzlichen Datensatz ausgeben, der die angegebene Kennung und die neue Seitennummer enthält.



Als Kennung darf eine beliebige Zeichenfolge angegeben werden; sie darf jedoch keine Leerzeichen, Minuszeichen, Nummernzeichen, Kommata, Gleichheitszeichen, Apostrophe oder Anführungszeichen enthalten.

Wird statt einer Kennung ein in spitze Klammern eingeschlossener Name einer TUSTEP-Variablen angegeben, so wird als Kennung der Inhalt dieser Variablen verwendet. Falls diese Kennung ein Nummernzeichen enthält, wird die Seitennummer nicht am Ende der Sätze erwartet bzw. eingefügt, sondern an Stelle des Nummernzeichens.

OPTIONEN	= -	* Normalfall.
	= ZEILEN	Beim Umwandeln einer TUSTEP-Datei in eine TUSTEP- oder Fremd-Datei aufeinander folgende Datensätze, die die gleiche Seiten- und Zeilennummer (ohne Rücksicht auf die Unterscheidungsnummer) haben, zu einem Datensatz zusammenfassen.
	= ZIRKUMFLEX	Das Zeichen Zirkumflex als normales Zeichen und nicht als Steuerzeichen interpretieren.
	= BOM	Bei Dateien, die im Unicode codiert sind: Byte-Order-Mark am Dateianfang unterdrücken.
	= RLM	Bei Dateien, die im Unicode codiert sind: Right-Left-Mark und Left-Right-Mark unterdrücken.
	= SYM	Bei Dateien, die im Unicode codiert sind: Codes F0xx wie xx aus dem Symbol-Font interpretieren. Es können auch mehrere Optionen angegeben werden; sie müssen durch Apostroph getrennt sein.
PROTOKOLL	= +	* Liste der hexadezimalen Codes ins Ablaufprotokoll ausgegeben.
	= -	Keine Liste der hexadezimalen Codes ausgeben.
	= -STD-	Liste der hexadezimalen Codes in die Standard-Protokoll-Datei ausgegeben.
	= datei	Name der Datei für die Liste der hexadezimalen Codes.

## Leistung

Mit diesem Kommando werden Daten nach den zur Spezifikation MODUS und CODE angegebenen Regeln umgewandelt bzw. umcodiert und ggf. mit dem zur Spezifikation SCHLUESSEL angegebenen Schlüssel verschlüsselt bzw. entschlüsselt.

Die Dateien, die zur Spezifikation QUELLE und ZIEL angegeben sind, können dabei

vom gleichen Typ (Fremd-Datei oder TUSTEP-Datei) oder von verschiedenem Typ sein. Dadurch ist es möglich, den Inhalt einer Fremd-Datei in eine TUSTEP-Datei zu kopieren und umgekehrt.

Wird von einer Fremd-Datei in eine TUSTEP-Datei umgewandelt, so werden die Daten von dem zur Spezifikation `CODE` angegebenen Code in den TUSTEP-Code umcodiert. Zeichen, die im jeweiligen Code nicht definiert sind, werden in die Form `#[xx]` umgewandelt, wobei `xx` den hexadezimalen Code des jeweiligen Zeichens angibt. Kommen die Codes `00` oder `FF` unmittelbar hintereinander vor, so werden sie in die Form `#[n*00]` bzw. `#[n*FF]` umgewandelt; dabei gibt `n` an, wie oft der jeweilige Code vorkam.

Wird von einer TUSTEP-Datei in eine Fremd-Datei umgewandelt, so werden die Daten vom TUSTEP-Code in den zur Spezifikation `CODE` angegebenen Code umcodiert. Zeichenfolgen der Form `#[xx]`, wobei `xx` ein hexadezimaler Code von `00` bis `FF` ist, werden in das Zeichen umgewandelt, das diesem Code entspricht. Zeichenfolgen der Form `#[n*xx]`, wobei `n` eine Dezimalzahl ist, werden in `n` entsprechende Codes umgewandelt.

Eine verschlüsselte Datei wird dadurch entschlüsselt, dass sie mit dem gleichen Schlüssel, der beim Verschlüsseln verwendet wurde, nochmals umgewandelt wird. Der Schlüssel ist eine mindestens 40 und maximal 400 Zeichen lange Zeichenfolge, die frei gewählt werden kann.

Die Reihenfolge der Bearbeitung der Daten auf Grund der Spezifikationen `MODUS`, `SCHLUESSEL` und `CODE` ist von der Angabe zur Spezifikation `MODUS` abhängig. Sie ist `CODE-MODUS-SCHLUESSEL` bei Modus `+0`, `+1`, `+2`, `+4`, `XT`, `XB`, `TK` und `SCHLUESSEL-MODUS-CODE` bei Modus `-0`, `-1`, `-2`, `-3`, `-4`, `TX`, `BX`, `KT`.

Bei `MODUS=0` darf nur zu einer der beiden Spezifikationen `SCHLUESSEL` und `CODE` etwas anderes als »-« (jeweils Voreinstellung) angegeben werden.

## Achtung

Daten, die mit diesem Kommando verschlüsselt wurden, dürfen nicht in Fremd-Dateien gespeichert werden! Sie können sonst möglicherweise nicht mehr korrekt entschlüsselt werden.

## Modi

- 0 Daten nicht zusätzlich umwandeln.
- +0 Wie 0, jedoch prüfen, ob die einzelnen Zeichen (nicht Zeichenkombinationen) in TUSTEP zulässig sind.
- 0 Wie 0, jedoch prüfen, ob die einzelnen Zeichen (nicht Zeichenkombinationen) in TUSTEP zulässig sind.
- +1 Die aus dem Zeichen »^« und einem Zeichen aus dem ASCII-Zeichensatz bestehende Eingabe-Codierungen (vgl. Tabellen Seite 763 und Seite 815) werden in die entsprechenden TUSTEP-Zeichen umgewandelt.

Beispiel:

Aus »Er löste die TUSTEP-Übungsaufgabe großartig«  
 wird »Er löste die TUSTEP-Übungsaufgabe großartig«.

- 1 Umkehrung von  $n=+1$ . Die Zeichen aus dem TUSTEP-Zeichensatz, die nicht im ASCII-Zeichensatz enthalten sind, werden durch die aus dem Zeichen »^« und einem Zeichen aus dem ASCII-Zeichensatz bestehende Eingabe-Codierung (vgl. Tabellen auf Seite 763 und Seite 815) ersetzt.

Falls jedoch auf Grund der Spezifikation CODE vom TUSTEP-Code in einen anderen Code umcodiert wird, werden diejenigen Zeichen aus dem TUSTEP-Zeichensatz, die im anderen Code vorgesehen sind, diesem Code entsprechend umcodiert und nicht durch »^x« ersetzt.

- +2 Wie  $n=+1$ ; zusätzlich werden, wenn nichts anderes durch das Bereichsumschaltzeichen »<« gefordert wird, alle Buchstaben in Kleinbuchstaben umcodiert. Enthalten die Daten das Zeichen »<«, so werden ab diesem Zeichen die Buchstaben in Großbuchstaben umcodiert. Mit dem Zeichen »>« wird die Wirkung von »<« wieder aufgehoben, d. h. die Buchstaben werden wieder in Kleinbuchstaben umcodiert. Die Zeichen »<« und »>« werden nicht übertragen. Sollen die Zeichen »<« bzw. »>« selbst dargestellt werden, so müssen sie mit »<<« bzw. mit »>>« codiert sein.

Beispiel:

Aus »<E>R L^OSTE DIE <TUSTEP-^U>BUNGSAUFGABE GRO^SARTIG«  
 wird »Er löste die TUSTEP-Übungsaufgabe großartig«.

- 2 Umkehrung von  $n=+2$ . Vor bzw. nach einem oder mehreren aufeinander folgenden Großbuchstaben wird das Bereichsumschaltzeichen »<« bzw. »>« eingesetzt. Kommen die Zeichen »<« und »>« vor, so werden sie verdoppelt. Außerdem werden die Zeichen aus dem TUSTEP-Zeichensatz, die nicht im ASCII-Code enthalten sind, durch die aus dem Zeichen »^« und einem Zeichen aus dem ASCII-Zeichensatz bestehende Eingabe-Codierung (vgl. Tabellen auf Seite 763 und 815) ersetzt.
- 3 Vor jedem Großbuchstaben wird ein »<« eingesetzt. Kommen die Zeichen »<« und »>« vor, so werden sie verdoppelt. Außerdem werden die Zeichen aus dem TUSTEP-Zeichensatz, die nicht im ASCII-Code enthalten sind, durch die aus dem Zeichen »^« und einem Zeichen aus dem ASCII-Zeichensatz bestehende Eingabe-Codierung (vgl. Tabellen auf Seite 763 und 815) ersetzt.
- 4 Die Daten in die hexadezimale Form (für jedes Zeichen eine hexadezimale Code-Angabe von 00 bis FF) umwandeln.
- +4 Umkehrung von -4. Jeweils zwei Zeichen werden als hexadezimale Code-Angabe interpretiert und in das Zeichen mit dem entsprechenden Code umgewandelt. Die Daten dürfen nur aus Paaren von hexadezimalen Ziffern bestehen.
- TX Die Textdaten aus der QUELL-Datei im Datenaustauschformat in die ZIEL-Datei schreiben.
- XT Die Daten, die im Datenaustauschformat in der QUELL-Datei stehen, als Textdaten in die ZIEL-Datei schreiben.
- BX Die Binärdaten aus der QUELL-Datei im Datenaustauschformat in die ZIEL-Datei schreiben.

- XB** Die Daten, die im Datenaustauschformat in der QUELL-Datei stehen, als Binärdaten in die ZIEL-Datei schreiben.
- TK** Die Textdaten aus der QUELL-Datei in komprimierter Form in die ZIEL-Datei schreiben.
- KT** Die Daten, die in komprimierter Form in der QUELL-Datei stehen, als Textdaten in die ZIEL-Datei schreiben.
- TQ** Die Textdaten aus der QUELL-Datei in Quoted-Printable-Codierung in die ZIEL-Datei schreiben.
- QT** Die Daten, die in Quoted-Printable-Codierung in der QUELL-Datei stehen, als Textdaten in die ZIEL-Datei schreiben.
- <>** Die Zeichen < und > werden als Umschaltzeichen auf lateinische Schrift und auf die zuvor eingestellte Schrift interpretiert. Außerdem werden Character-Entities (z. B. `&auml;`, `&^#123;` und `&^#x01EF;`) auch nach Umschalten auf eine nicht-lateinische Schrift in lateinischer Schrift beibehalten. Dieser Modus ist nur erlaubt, wenn zur Spezifikation CODE entweder UNICODE, UTF16 oder UTF8 angegeben ist. Näheres ist im folgenden Kapitel beschrieben.
- &** Beim Umwandeln einer Fremd- in eine TUSTEP-Datei werden Character-Entities in die entsprechenden TUSTEP-Codierungen umgewandelt (z. B. `&auml;` in `ä`); außerdem werden #, \$, %, &, @, \, \_ zu ^#, ^\$, ^%, ^&, ^@, ^\, ^\_. Dieser Modus ist nur erlaubt, wenn zur Spezifikation CODE entweder ANSI oder ISO8859 angegeben ist.
- +&** Beim Umwandeln einer Fremd- in eine TUSTEP-Datei werden Character-Entities in die entsprechenden TUSTEP-Codierungen umgewandelt (z. B. `&auml;` in `ä` und `&#x03B2;` in `#g+b#g-`); das Zeichen ^ hat die gleiche Bedeutung wie bei Modus +1; außerdem gelten die gleichen Besonderheiten wie beim Umwandeln von UTF-16 oder UTF-8 nach TUSTEP, die nachfolgend beschrieben sind. Beim Modus +& muss zur Spezifikation CODE ISO8859 angegeben werden.
- &** Beim Umwandeln einer TUSTEP- in eine Fremd-Datei werden Zeichen, die im Code ISO-8859-1 nicht darstellbar sind, in Character-Entities der Form `&#xxxx;` umgewandelt (z. B. `#g+b#g-` in `&#x03B2;`); die anderen Zeichen werden wie bei Modus -1 umgewandelt; außerdem gelten die gleichen Besonderheiten wie beim Umwandeln von TUSTEP nach UTF-16 oder UTF-8, die nachfolgend beschrieben sind. Beim Modus -& muss zur Spezifikation CODE ISO8859 angegeben werden.

### Besonderheiten beim Umwandeln von TUSTEP nach UTF-16 oder UTF-8

^#, ^\$, ^%, ^&, ^@, ^\, ^\_ werden zu #, \$, %, &, @, \, \_

#, \$, %, &, @, ^ werden zu ^#, ^\$, ^%, ^&, ^@, ^^

\ wird zu SHY (Soft Hyphen, 00AD)

\_ wird zu NBSP (No-Break Space, 00A0)

< und > bleiben < und > (Steuerzeichen, 003C und 003E)

^< und ^> werden zu < und > (Druckzeichen, 2329 und 232A)

Schriftumschaltungen (z. B. #G+ und #G-) werden ausgewertet und die entsprechenden Unicode-Zeichen generiert. Die Schriftumschaltungen selbst werden übernommen (z. B. als ^#G+ und ^#G-), falls der Kennbuchstabe für die Schrift (z. B. G) ein Großbuchstabe ist, andernfalls fällt die Schriftumschaltung weg.

Andere Auszeichnungen (z. B. #1+ und #1- oder #G: und #g:) werden in jedem Fall (z. B. als ^#1+ und ^#1- oder ^#G: und ^#g:) übernommen.

Wird MODUS=<> angegeben (Voreinstellung ist MODUS=-1), so schaltet das Steuerzeichen < (z. B. zwischen #g+ und #g-) automatisch auf lateinische Schrift um und das nachfolgende Steuerzeichen > wieder automatisch auf die zuvor eingestellte Schrift (z. B. griechische Schrift) zurück. Die Steuerzeichen < und > werden in jedem Fall übernommen. Außerdem werden die Namen von Character-Entities (z. B. &auml;) auch nach Umschalten auf eine nicht-lateinische Schrift in lateinischer Schrift beibehalten.

Zeichenfolgen der Form # [xxxx], wobei xxxx ein hexadezimaler Code von 0000 bis FFFF ist, werden in das Unicode-Zeichen umgewandelt, das diesem Code entspricht. Zeichenfolgen der Form # [n\*xxxx], wobei n eine Dezimalzahl ist, werden in n entsprechende Unicode-Zeichen umgewandelt.

Soll das Zeichen ^ unverändert übernommen werden und nicht wie oben angegeben zu ^^ werden, so muss es zuvor in # [005E] ausgetauscht werden.

## Besonderheiten beim Umwandeln von UTF-16 oder UTF-8 nach TUSTEP

#, \$, %, &, @, \, \_ werden zu ^#, ^\$, ^%, ^&, ^@, ^\, ^\_

^#, ^\$, ^%, ^&, ^@, ^\, ^\_, ^^ werden zu #, \$, %, &, @, \, \_, ^

SHY (Soft Hyphen, 00AD) wird zu \

NBSP (No-Break Space, 00A0) wird zu \_

< und > (Steuerzeichen, 003C und 003E) bleiben < und >

< und > (Druckzeichen, 2329 und 232A) werden zu ^< und ^>

Soweit erforderlich werden Schriftumschaltungen (z. B. #g+ und #g- für griechische Schrift) eingefügt.

Wird MODUS=0 angegeben (Voreinstellung ist MODUS=-1), so gilt das Zeichen ^ nicht als Steuerzeichen und wird unverändert übernommen. Dies bedeutet, dass die oben in der zweiten Zeile angegebenen Umcodierungen (z. B. ^# in #) entfallen und die dort angegebenen Zeichen entsprechend den in der ersten Zeile angegebenen Umcodierungen umgewandelt werden.

Unicode-Zeichen, für die in TUSTEP keine Codierung vorgesehen ist, werden in die Form # [xxxx] umgewandelt, wobei xxxx den hexadezimalen Code des jeweiligen Zeichens angibt. Kommen die Codes 0000 oder FFFF unmittelbar hintereinander vor, so werden sie in die Form # [n\*0000] bzw. # [n\*FFFF] umgewandelt; dabei gibt n an, wie oft der jeweilige Code vorkam.

**Einschränkung**

Wird zur Spezifikation ZIEL eine Fremd-Datei angegeben, so muss diese leer sein, oder es muss die Spezifikation LOESCHEN=+ angegeben werden, damit die Daten in dieser Datei gelöscht werden. Anhängen an bereits vorhandene Daten ist nur bei TUSTEP-Dateien möglich.

# Vergleichsergebnisse aufbereiten zum Drucken

#VAUFBEREITE

## Kommando

#VAUFBEREITE

Für dieses Kommando gibt es eine eigene Beschreibung.

## Leistung

Mit diesem Programm können Abweichungen einer oder mehrerer Textversionen von einem gemeinsamen Grundtext zusammen mit dem Grundtext zeilensynoptisch zum Drucken aufbereitet werden. Die Abweichungen müssen in Form von Korrekturanweisungen vorliegen, die mit einem Korrekturschlüssel versehen sind, wie er von den Kommandos #VERGLEICHE oder #SVORBEREITE erzeugt wird. Dabei wird zeilenweise unter den Wörtern des Grundtextes, für die eine Abweichung festgestellt wurde, der abweichende Wortlaut (Variante) ausgegeben. Übereinstimmungen zwischen dem Grundtext und den übrigen Textversionen, sowie Übereinstimmungen zwischen den Varianten selbst, werden markiert.

# Vergleichen fortlaufender Texte

#VERGLEICHE

## Kommando

#VERGLEICHE

Für dieses Kommando gibt es eine eigene Beschreibung.

## Leistung

Mit diesem Programm können zwei Textversionen (A und B) miteinander verglichen werden. Die festgestellten Unterschiede werden in der zu PROTOKOLL angegebenen Datei protokolliert. Außerdem können die Unterschiede in Form von Korrekturanweisungen in die zu KORREKTUR angegebene Datei ausgegeben werden. Sie entsprechen den Konventionen für die Korrekturanweisungen des Kommandos #KAUSFUEHRE; würde mit den Korrekturanweisungen die Version A korrigiert, entstünde (u. U. bis auf die Zeileneinteilung und die Satznummern) die Version B.

Die Zeileneinteilung der beiden Versionen kann völlig verschieden sein. Auslassungen bzw. Einfügungen werden (zeitaufwändig) bis etwa zur Länge einer DIN-A4-Seite erkannt. Auslassungen bzw. Einfügungen beliebiger Länge können durch entsprechende Angaben berücksichtigt werden. Es ist auch möglich, nur bestimmte Dateibereiche zu vergleichen.



# Wahlschalter setzen/löschen

#WAHLSCHALTER

## Kommando

#WAHLSCHALTER

SETZE	= WS <sub>n</sub>	Wahlschalter <i>n</i> setzen. Statt der Angabe WS <sub>n</sub> kann, mit gleicher Bedeutung, auch nur die Zahl <i>n</i> angegeben werden.  Es können auch mehrere Wahlschalter angegeben werden; sie müssen durch Apostroph getrennt sein.
	= -	* Keinen Wahlschalter setzen.
LOESCHE	= WS <sub>n</sub>	Wahlschalter <i>n</i> löschen. Statt der Angabe WS <sub>n</sub> kann, mit gleicher Bedeutung, auch nur die Zahl <i>n</i> angegeben werden.  Es können auch mehrere Wahlschalter angegeben werden; sie müssen durch Apostroph getrennt sein.
	= -	* Keinen Wahlschalter löschen.

## Leistung

Es gibt in TUSTEP sieben Wahlschalter, die von 1 bis 7 durchnummeriert sind. Sie können mit diesem Kommando gesetzt bzw. gelöscht werden.

Wird das Kommando ohne Spezifikationsangaben aufgerufen, so wird nur angezeigt, welche Wahlschalter gesetzt bzw. gelöscht sind.

Beim Initialisieren einer TUSTEP-Sitzung werden alle Wahlschalter gelöscht.

## Hinweis

Die Wahlschalter können zur Steuerung der TUSTEP-Programme verwendet werden:

- Wird auf Parametern in Spalte 4 und 5  $+n$  bzw.  $-n$  angegeben, so wird der betreffende Parameter nur ausgewertet, falls der Wahlschalter *n* gesetzt ( $+n$ ) bzw. gelöscht ( $-n$ ) ist.
- In Makros können diese Wahlschalter abgefragt werden, und in Abhängigkeit davon kann das Abarbeiten des Makros fortgesetzt werden.
- Beim Programm KOPIERE können (durch entsprechende Parameterangaben) diese Wahlschalter zu Anfang in die programminternen Wahlschalter übernommen werden, und am Ende können sie entsprechend den programminternen Wahlschaltern gesetzt bzw. gelöscht werden.

# Wischen (Überschreiben) von Daten ein/ausschalten

#WISCHEN

## Kommando

#WISCHEN

MODUS = AUS Wischen (Überschreiben) der Daten ausschalten.  
= EIN Wischen (Überschreiben) der Daten einschalten.

## Leistung

Mit diesem Kommando kann eingestellt werden, ob beim Löschen von Dateien zuerst die darin enthaltenen Daten überschrieben werden sollen (MODUS=EIN) oder nicht (MODUS=AUS). Wird zur Spezifikation MODUS nichts angegeben, so wird nur angezeigt, ob das Überschreiben ein- oder ausgeschaltet ist.

Beim Initialisieren einer TUSTEP-Sitzung wird das Überschreiben ausgeschaltet.

Das Überschreiben der Daten betrifft nicht nur das explizite Löschen von Dateien mit dem Kommando #LOESCHE, sondern auch das automatische Löschen bei den Kommandos #ABMELDE, #SORTIERE, #BEENDE und #NORMIERE. Bei all diesen Kommandos kann jedoch unabhängig vom Modus, der mit dem Kommando #WISCHEN eingestellt wurde, angegeben werden, ob die Daten im Einzelfall überschrieben werden sollen oder nicht; in diesem Fall hat die Angabe bei dem jeweiligen Kommando Vorrang.

## Anmerkung

Bei einigen Betriebssystemen besteht die Möglichkeit, dass auf Daten von gelöschten Dateien wieder zugegriffen werden kann. Dies kann auf dem eigenen Rechnern von Vorteil sein, ist aber auf einem allgemein zugänglichen Rechner aus Datenschutzgründen nicht wünschenswert, da ja auch fremde Benutzer auf solche Daten zugreifen können. Deshalb können in TUSTEP die Daten in Dateien, die gelöscht werden sollen, zuvor mit Nullen überschrieben werden.

## Zeitabfrage

#ZEIT

### Kommando

#ZEIT

(keine Spezifikationen)

### Leistung

Dieses Kommando bewirkt die Ausgabe des Datums, der Uhrzeit sowie (mit Ausnahme der Windows-Version) der bisher in der Sitzung verbrauchten Rechenzeit.



**Editor**

Editor

---

Aufruf des Editors . . . . .	257
Zwischenablage – Editor-Zwischenspeicher . . . . .	258
Zeichenvorrat . . . . .	259
Tastaturbelegung . . . . .	259
Tastenbezeichnungen . . . . .	259
Tastenkombinationen (Auswahl) . . . . .	261
Mausaktionen (Auswahl) . . . . .	268
Eingabehilfe . . . . .	269
Editorfenster . . . . .	270
Darstellung der Sätze im Textfeld . . . . .	270
Nummerierung der Sätze . . . . .	271
Änderungen im Textfeld . . . . .	271
Aufruf der Online-Hilfe . . . . .	271
Reorganisieren der Dateien . . . . .	272
Einfache Anweisungen . . . . .	273
b Beenden des Editors . . . . .	273
e Eintragen, Einfügen neuer Daten . . . . .	273
z Zeigen der Daten im Textfeld . . . . .	275
Korrigieren der Daten im Textfeld . . . . .	275
l Löschen eines Satzes oder Bereichs . . . . .	277
k Kopieren eines Satzes oder Bereichs . . . . .	278
u Umnummerieren, Umstellen eines Satzes oder Bereichs . . . . .	279
Organisatorische Anweisungen . . . . .	280
n Namen für Datei und Segment abfragen/einstellen . . . . .	280
h Holen (Kopieren) aus einer Datei . . . . .	281
r Retten (Abspeichern) in eine Datei . . . . .	282
d Editor-Datei abfragen/wechseln . . . . .	284
m Modus abfragen/einstellen . . . . .	285
t Dateititel abfragen/einstellen/löschen . . . . .	286
t Tabulator abfragen/einstellen/löschen . . . . .	286
f Funktionen aufrufen/definieren/löschen/abfragen . . . . .	287
y Makros aufrufen/definieren/löschen/abfragen . . . . .	288
i Zeichen- und Stringgruppen definieren/abfragen . . . . .	296
c Colorierung definieren/wechseln/löschen/abfragen . . . . .	298
t Tag-Prüfung definieren/wechseln/löschen/abfragen . . . . .	300
x Kommandos ausführen . . . . .	303
g Ausgeben und Wiederholen von Anweisungen . . . . .	304
Erweiterte Anweisungen . . . . .	307
z Zeigen unter Bedingungen . . . . .	307

e Einfügen unter Bedingungen . . . . .	308
l Löschen unter Bedingungen . . . . .	309
k Kopieren unter Bedingungen . . . . .	309
u Umstellen unter Bedingungen . . . . .	310
a Austauschen von Zeichenfolgen . . . . .	311
Such-Anweisungen für strukturierte Daten . . . . .	312
p Parameter definieren/löschen . . . . .	312
s Suchen von Texteinheiten . . . . .	315
p Parameter abfragen/löschen/kopieren . . . . .	316
Suchen und Prüfen von Tags . . . . .	317
Prüfen von Klammern . . . . .	324
Angaben von Satzpositionen <code>pos</code> und Dateibereichen <code>ber</code> . . . . .	328
Begrenzung auf bestimmte Spalten <code>begr</code> . . . . .	329
Begrenzung auf Satznummer . . . . .	329
Nur ganze Wörter suchen/austauschen . . . . .	330
Ignorieren von Akzent-Codierungen . . . . .	330
Ignorieren von Auszeichnungen und Schriftumschaltungen . . . . .	330
Vergleichs-Tabelle <code>vtab</code> . . . . .	331
Such-Tabelle <code>stab</code> . . . . .	331
Austausch-Tabelle <code>atab</code> . . . . .	332
Recherchier-Tabelle <code>rtab</code> . . . . .	332
Anpassen des Editors (Optionen) . . . . .	333
Tastenkombinationen für Funktionsaufrufe . . . . .	341
Tastenkombinationen für Makroaufrufe . . . . .	342
Mausaktionen für Makroaufrufe . . . . .	344
Tastenkombinationen für Steuerbefehle . . . . .	346
Steuerbefehle im Editor . . . . .	350
Cursor . . . . .	351
Blättern, Scrollen . . . . .	357
Einfügen, Überschreiben . . . . .	359
Löschen und Einfügen . . . . .	361
Markieren, Kopieren, Löschen, Einfügen, Suchen . . . . .	365
Wiederherstellen gelöschter/geänderter Sätze . . . . .	372
Lesezeichen . . . . .	373
Verzweigen in andere Makros . . . . .	375
Anzeigen von Daten . . . . .	382
Anzeigen von Meldungen . . . . .	382
Anzeigen von Makroleisten . . . . .	383

---

Anzeigen von Menüs . . . . .	384
Anzeigen von Hilfetexten . . . . .	384
Anzeigen vorgesehener Tags . . . . .	385
Anzeigen des Inhalts einer Segment-Datei . . . . .	386
Übernehmen von zusätzlichen Definitionen . . . . .	386
Tags . . . . .	386
Laufende Nummer . . . . .	389
Satznummer . . . . .	389
Seitennummer . . . . .	389
Dateiname und Segmentname . . . . .	390
Aktuelles Datum . . . . .	390
Aktuelle Uhrzeit . . . . .	391
Aktuelle Satzposition . . . . .	391
Aktuelle Wortposition . . . . .	392
Zwischenablage . . . . .	392
Anzeigen von Dateien, Starten von Anwendungen . . . . .	394
Carriage Return . . . . .	394
Anzeige-Modus . . . . .	395
Fehlersuche . . . . .	396
Weitere Steuerbefehle . . . . .	397



## Aufruf des Editors

Der Editor kann mit dem Kommando #EDIERE (siehe Seite 149) oder mit dem im Folgenden beschriebenen Makro \*E gestartet werden. Der Unterschied besteht im Wesentlichen darin, dass mit dem Makro die zu edierende Datei zuvor angemeldet wird, und dass auch Textdaten aus Dateien von anderen Programmen (Fremd-Dateien, z. B. ASCII-Dateien) ediert werden können.

Der Editor kann auch außerhalb von TUSTEP über den Dateimanager gestartet werden. Dies ist für Windows auf Seite 103, für Linux auf Seite 104 und für macOS auf Seite 105 beschrieben.

### Aufruf des Makros \*E

#\*E

LESEN	= datei	Name der Datei, die zum Lesen angemeldet und ediert werden soll.
SCHREIBEN	= datei	Name der Datei, die zum Schreiben angemeldet und ediert werden soll.
TRAEGER	= name	Name der System-Variablen, die den Pfad für die anzumeldende und zu edierende Datei enthält.  Unter Windows kann auch direkt der Laufwerksbuchstabe angegeben werden, falls der Pfad keinen Verzeichnisnamen enthält.
	= -STD-	Dateien anmelden, die sich auf dem Standard-Träger für permanente Dateien befinden; dieser ist durch die System-Variable TUSTEP_DSK vorgegeben.
	= -	Zur Spezifikation LESEN bzw. SCHREIBEN ist ein »definierter Dateiname« (siehe Seite 29) angegeben.
MAKRO	= name	Name des Editormakros, das nach dem Start des Editors automatisch aufgerufen und ausgeführt werden soll.
	= -	* Nach dem Start des Editors kein Makro automatisch aufrufen.
CODE	= name	Name des Codes, in dem die Daten codiert sind, falls die Datei eine Fremd-Datei ist.
	= ASCII	Daten dem internationalen ASCII-Code (siehe Tabelle Seite 815) entsprechend umcodieren.
	= ANSI	Wie CP1252.
	= CPnnn	Daten der Code-Page nnn (nnn = 437, 850, 852, 1250, 1252, 10000, 10029) entsprechend umcodieren (siehe Tabellen ab Seite 818).

---

= ISO8859	Daten dem Zeichensatz 1 der ISO-Norm 8859 (siehe Tabelle Seite 826) entsprechend umcodieren.
= LATINn	Daten dem Code Latin n (n = 1 bis 10) entsprechend umcodieren (siehe Tabellen ab Seite 828).
= LINUX	Daten dem Linux-Zeichensatz (siehe Tabelle Seite 827) entsprechend umcodieren.
= DECMCS	Daten dem »DEC Multinational Character Set« (siehe Tabelle Seite 825) entsprechend umcodieren.
= UNICODE	Wie UTF16.
= UTF16	Daten den UTF-16-Konventionen entsprechend umcodieren.
= UTF8	Daten den UTF-8-Konventionen entsprechend umcodieren.

## Leistung

Mit dem Makro \*E kann eine Datei zum Lesen oder Schreiben angemeldet und mit dem Editor bearbeitet werden.

Wenn die betreffende Datei eine Fremd-Datei ist, werden die Daten umgewandelt, in eine interne TUSTEP-Datei geschrieben und der Editor mit dieser internen Datei gestartet. Falls die Daten geändert werden, wird nach dem Beenden des Editors gefragt, ob die Daten in die ursprüngliche Datei zurückgeschrieben werden sollen. Für diese Anfrage gibt es vier zulässige Antworten:

- 1) J Daten zurückschreiben.
- 2) N Daten nicht zurückschreiben.
- 3) V Geänderte Daten mit den Originaldaten vergleichen und die Unterschiede anzeigen. Danach Anfrage wiederholen.
- 4) E Editor nochmals aufrufen.

Werden beim Aufruf des Makros keine Spezifikationswerte angegeben, so wird ein Fenster angezeigt, in dem sowohl der Projekt- und Dateiname als auch der Träger ausgewählt werden kann.

## Zwischenablage – Editor-Zwischenspeicher

Der Editor kennt neben der »Zwischenablage« auch noch einen zweiten Speicherbereich, in dem Daten zwischengespeichert werden können; er wird mit »Editor-Zwischenspeicher« bezeichnet. Auf die Zwischenablage kann auch außerhalb des Editors zugegriffen werden, auf den Editor-Zwischenspeicher kann nur der Editor zugreifen. Der Inhalt der Zwischenablage kann nur ersetzt oder eingefügt werden, der Inhalt des Editor-Zwischenspeichers kann auch modifiziert werden, bevor er wieder eingesetzt wird.

Auf beide Speicherbereiche kann im Editor mit Steuerbefehlen zugegriffen werden. Für die Zwischenablage sind sie im Abschnitt »Zwischenablage« ab Seite 392 beschreiben, für den Editor-Zwischenspeicher im Abschnitt »Markieren, ...« ab Seite 365. Auf die Zwischenablage kann außerdem mit der Anweisung EZ (siehe Abschnitt »Eintragen, Einfügen« ab Seite 273) zugegriffen werden.

Außer mit dem Editor kann auch mit Makroanweisungen (siehe Kapitel »Makros« ab Seite 403) auf die Zwischenablage zugegriffen werden.

Unter Windows kann außerhalb von TUSTEP direkt auf die »Zwischenablage« zugegriffen werden, da für die TUSTEP-Zwischenablage und die Windows-Zwischenablage derselbe Speicherbereich verwendet wird.

Unter Linux und macOS muss ein Makro aufgerufen werden, um die Daten von der TUSTEP-Zwischenablage in das Clipboard bzw. Pasteboard des Betriebssystems und umgekehrt zu transferieren (siehe Seite 55).

Unter macOS kann mit `CMD+V` der Inhalt des Pasteboard von macOS an der Cursor-Position eingefügt werden.

## Zeichenvorrat

Unabhängig davon, welcher Zeichenvorrat auf der Tastatur vorhanden ist, können auf der Tastatur nur solche Zeichen verwendet werden, die im Zeichenvorrat des mit dem Kommando `#DEFINIERE` (siehe Seite 132) eingestellten Code vorhanden sind. Wie Zeichen codiert werden, die nicht im Zeichenvorrat der Tastatur oder nicht dem des eingestellten Codes enthalten sind, ist dem Kapitel »Zeichenvorrat« ab Seite 759 zu entnehmen.

## Tastaturbelegung

Falls Tasten oder Tastenkombinationen nicht die in dieser Beschreibung angegebene Wirkung haben, kann es daran liegen, dass die in der Installationsanleitung angegebenen Einstellungen nicht (alle) vorgenommen wurden oder dass TUSTEP nicht in der Weise aufgerufen wurde, wie dies in der Installationsanleitung und in dieser Beschreibung ab Seite 77 angegeben ist.

## Tastenbezeichnungen

- |       |  |
|-------|--|
| Shift | »Umschalttaste« für Großbuchstaben.  |
| Ctrl  | »Control-Taste« links im Buchstabenblock unter der Shift-Taste; auf deutschen Tastaturen meist mit »Strg« beschriftet. |

Alt	»Alt-Taste« zwischen linker Ctrl-Taste und Leertaste.
Return	»Eingabetaste« rechts im Buchstabenblock über der Shift-Taste.
Enter	»Eingabetaste« rechts unten im Ziffernblock. Alternativ kann in TUSTEP auch <code>Shift+Return</code> oder <code>Ctrl+E</code> verwendet werden.
Backspace	»Rücktaste« rechts im Buchstabenblock über der »Eingabetaste«.
Escape	»Abbruchtaste« ganz oben links, sie ist mit »Esc« beschriftet.
Plus	»Plus-Taste« im Ziffernblock. Alternativ kann in TUSTEP auch <code>Ctrl+B</code> verwendet werden.

Die folgenden Tasten befinden sich im Mittelblock zwischen Buchstabenblock und Ziffernblock. Gleichbeschriftete Tasten im Ziffernblock können eventuell alternativ verwendet werden; in Kombination mit den Tasten `Shift`, `Ctrl` und `Alt` rufen sie jedoch Makros auf (siehe »Tastenkombinationen für Makroaufrufe« Seite 342). Bei Tastaturen ohne Ziffernblock sind die folgenden Tasten in der rechten oberen und rechten unteren Ecke der Tastatur und erfordern eventuell ein zusätzliches Drücken der `Fn`-Taste.

Insert	»Einfügetaste«, sie ist mit »Einfg«, »Insert« oder »Ins« beschriftet. Auf MAC-Tastaturen fehlt diese Taste.
Delete	»Löschtaste«, sie ist mit »Entf« oder »Del« beschriftet. Alternativ kann in TUSTEP auch <code>Shift+Backspace</code> verwendet werden.
PfO	»Pfeil-nach-oben-Taste«
PfU	»Pfeil-nach-unten-Taste«
PfL	»Pfeil-nach-links-Taste«
PfR	»Pfeil-nach-rechts-Taste«
Pos1	»Anfangtaste«, sie ist mit »Pos1«, »Home« oder einem Pfeil nach links oben beschriftet. Unter macOS kann alternativ <code>fn+PfL</code> verwendet werden.
End	»Endetaste«, sie ist mit »Ende«, »End« oder einem Pfeil nach rechts unten beschriftet. Unter macOS kann alternativ <code>fn+PfR</code> verwendet werden.
PgUp	»Bild-rauf-Taste«, sie ist mit »Page Up«, »PgUp« oder mit »Bild« und einem Pfeil nach oben beschriftet. Unter macOS kann alternativ <code>fn+PfO</code> verwendet werden.
PgDn	»Bild-runter-Taste«, sie ist mit »Page Down«, »PgDn« oder mit »Bild« und einem Pfeil nach unten beschriftet. Unter macOS kann alternativ <code>fn+PfU</code> verwendet werden.

## Tastenkombinationen (Auswahl)

Welche Tasten mit der in der linken Spalte angegebenen Tastenbezeichnungen gemeint sind, ist im vorangehenden Kapitel »Tastenbezeichnungen« auf Seite 259 beschrieben.

Pos1	<p>Cursor springt an den Zeilenanfang; steht der Cursor schon dort, springt er an den Zeilenanfang in der vorangehenden Zeile.</p> <p>Falls die Shift-Taste niedergehalten und die genannte Taste gedrückt wird und außerdem das Markieren noch nicht (mit dem Steuerbefehl MRK_INI) initialisiert ist, wird zuerst das Markieren initialisiert und dann erst der Cursor an die neue Position bewegt.</p>
End	<p>Cursor springt ans Zeilenende; steht der Cursor schon dort, springt er ans Zeilenende in der nächsten Zeile.</p> <p>Falls die Shift-Taste niedergehalten und die genannte Taste gedrückt wird und außerdem das Markieren noch nicht initialisiert ist, wird zuerst das Markieren initialisiert und dann erst der Cursor an die neue Position bewegt.</p>
Plus+Pos1	Löscht alle Zeichen links von der aktuellen Cursor-Position bis zum Zeilenanfang. Alle in der Zeile folgenden Zeichen werden um die entsprechende Anzahl von Zeichen nach links verschoben. Der Cursor steht danach am Anfang der Zeile.
Plus+End	Löscht alle Zeichen von der aktuellen Cursor-Position bis zum Zeilenende.
Enter	Bestätigen der Änderungen bzw. der Eingabe. Daten werden in die Datei übertragen; Anweisungen werden ausgeführt. Der Cursor springt an den Anfang des Eingabebereichs in der Anweisungszeile.
Shift+Return	Alternative zur Enter-Taste.
Ctrl+Enter	Fügt den im Editor-Zwischenspeicher gemerkten Text vor der aktuellen Cursor-Position ein. Alle in der Zeile folgenden Zeichen werden nach rechts verschoben. Der Cursor wird mitverschoben und steht danach hinter den eingefügten Zeichen.
Delete	<p>Löscht das Zeichen auf der aktuellen Cursor-Position und verschiebt alle in der Zeile folgenden Zeichen um eine Position nach links.</p> <p>Ist im Editorfenster Text markiert, so wird der gesamte markierte Text gelöscht, ohne ihn im Editor-Zwischenspeicher zu merken.</p>
PgDn	Blättert vorwärts, beginnend mit dem Satz, in dem der Cursor steht; steht der Cursor in der Anweisungszeile: beginnend mit dem Satz, der dem letzten im Textfeld stehenden folgt; bei leerem Textfeld: beginnend mit dem Satz, der auf die aktuelle Satzposition folgt.

---

PgUp	Blättert rückwärts, beginnend mit dem Satz, in dem der Cursor steht; steht der Cursor in der Anweisungszeile: beginnend mit dem Satz, der dem ersten im Textfeld stehenden vorangeht; bei leerem Textfeld: beginnend mit dem Satz, der der aktuellen Satzposition vorangeht.
Shift+Backspace	Alternative zur Delete-Taste.
Ctrl+Backspace	Löscht das Textfeld und die Anweisungszeile. Dies ist sinnvoll, wenn Änderungen im Editorfenster unberücksichtigt bleiben sollen oder wenn eine Anweisung gegeben werden soll, die nicht in die Anweisungszeile passt.
Shift+Ctrl+Backspace	Löscht den Satz, in dem der Cursor steht und positioniert den Cursor an den Anfang des nächsten Satzes. Bei n-maliger Ausführung werden n aufeinander folgende Sätze gelöscht.
Ctrl+A	Cursor springt ans Ende des Textes in der Anweisungszeile.
Ctrl+C	Kopiert den markierten Text in die Zwischenablage.
Ctrl+T	Tauscht das Zeichen auf der aktuellen Cursor-Position mit dem unmittelbar davor stehenden Zeichen aus.
Ctrl+V	Fügt den Inhalt der Zwischenablage vor der aktuellen Cursor-Position ein. Alle in der Zeile folgenden Zeichen werden nach rechts verschoben. Der Cursor wird mitverschoben und steht danach hinter den eingefügten Zeichen.
Ctrl+X	Kopiert den markierten Text in die Zwischenablage und löscht den markierten Text im Editorfenster.
Ctrl+Z	Fügt die zuletzt mit der Delete- und/oder der Backspace-Taste gelöschte Zeichenfolgen vor der aktuellen Cursor-Position ein. Alle in der Zeile folgenden Zeichen werden nach rechts verschoben. Der Cursor wird mitverschoben und steht danach hinter den eingefügten Zeichen.
Ctrl+Pfo	Cursor springt ans Ende der vorhergehenden hervorgehobenen Zeichenfolge. Falls von der aktuellen Cursor-Position bis zum Anfang des Textfeldes keine solche vorhanden ist, springt der Cursor in die Anweisungszeile.
Ctrl+PFU	Cursor springt an den Anfang der nächsten hervorgehobenen Zeichenfolge. Falls von der aktuellen Cursor-Position bis zum Ende des Textfeldes keine solche vorhanden ist, springt der Cursor in die Anweisungszeile.
Ctrl+Pfl	Cursor springt an den Anfang des »Wortes«, in dem er steht. Steht der Cursor am Anfang eines »Wortes«, springt er an den Anfang des vorhergehenden »Wortes«; steht der Cursor am Anfang des ersten »Wortes« einer Zeile oder davor, springt er hinter das letzte »Wort« in der vorhergehenden Zeile auf die Position, an der das nächste »Wort« beginnen kann.

Ctrl+Pfr	Cursor springt an den Anfang des nächsten »Wortes«. Steht der Cursor im letzten »Wort« einer Zeile, so springt er hinter dieses »Wort« auf die Position, an der das nächste »Wort« beginnen kann; steht der Cursor schon hinter dem letzten »Wort«, springt er an den Anfang des ersten »Wortes« in der nächsten Zeile.
Alt+Pfo	Tauscht den Satz, in dem der Cursor steht, mit dem im Textfeld unmittelbar vorangehenden Satz aus; die Satznummern werden dabei nicht mit ausgetauscht. Bei n-maliger Ausführung wird der Satz um n Sätze nach oben verschoben.
ALT+Pfu	Tauscht den Satz, in dem der Cursor steht, mit dem im Textfeld unmittelbar nachfolgenden Satz aus; die Satznummern werden dabei nicht mit ausgetauscht. Bei n-maliger Ausführung wird der Satz um n Sätze nach unten verschoben.
Alt+Pfl	Tauscht das Zeichen auf der aktuellen Cursor-Position mit dem unmittelbar davor stehenden Zeichen aus; der Cursor rückt um eine Position nach links. Bei n-maliger Ausführung wird das Zeichen um n Zeichen nach links verschoben.
Alt+Pfr	Tauscht das Zeichen auf der aktuellen Cursor-Position mit dem unmittelbar danach stehenden Zeichen aus; der Cursor rückt um eine Position nach rechts. Bei n-maliger Ausführung wird das Zeichen um n Zeichen nach rechts verschoben.
Ctrl+Home	Definiert die Anfangsposition der Markierung.
Ctrl+End	Kopiert den markierten Text in den Editor-Zwischenspeicher. Falls der Editor-Zwischenspeicher schon Text enthält, wird dieser vorher gelöscht.
Ctrl>Delete	Kopiert den markierten Text in den Editor-Zwischenspeicher und löscht den markierten Text im Editorfenster. Falls der Editor-Zwischenspeicher schon Text enthält, wird dieser vorher gelöscht.
Ctrl+Insert	Fügt den im Editor-Zwischenspeicher gemerkten Text vor der aktuellen Cursor-Position ein. Alle in der Zeile folgenden Zeichen werden nach rechts verschoben. Der Cursor wird mitverschoben und steht danach hinter den eingefügten Zeichen.
Plus+Plus+Backspace	Löscht das Textfeld und die Anweisungszeile. Dies ist sinnvoll, wenn Änderungen im Editorfenster unberücksichtigt bleiben sollen oder wenn eine Anweisung gegeben werden soll, die nicht in die Anweisungszeile passt.

Die im Folgenden beschriebenen Tastenkombinationen haben nur die genannte Wirkung, wenn für die entsprechenden Funktionstasten bzw. Makros die voreingestellten Definitionen nicht geändert wurden.

---

F1	Dateianfang anzeigen.
F2	Dateiende anzeigen.
F3	Nächste Zeichenfolge suchen.
F4	Editor beenden.
F5	Umgebung des aktuellen Satzes anzeigen.
F6	Liste mit den zuletzt mit dem Editor bearbeiteten Dateien anzeigen.  Soll in eine der aufgeführten Dateien gewechselt werden, so muss der entsprechende Dateiname mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Return-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der linken Maustaste angeklickt werden.  Soll die entsprechende Anweisung in die Anweisungszeile ausgegeben werden, damit sie ggf. in veränderter Form ausgeführt werden kann, muss die Auswahl mit dem Enter-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der rechten Maustaste angeklickt werden.
F7	Aktuellen Satz und Sätze davor anzeigen.
F8	Aktuellen Satz und Sätze danach anzeigen.
F9	In der Liste der ausgeführten Anweisungen zurückblättern.
F10	In der Liste der ausgeführten Anweisungen vorblättern.
F11	In das obere bzw. linke Textfeld wechseln.
F12	In das untere bzw. rechte Textfeld wechseln.
ALT+A	Sucht im Textfeld ab der Cursor-Position vorwärts nach einem Ende-Tag, zu dem im durchsuchten Text kein Anfangs-Tag vorhanden ist, und fügt an der Cursor-Position das entsprechende Anfangs-Tag ein (vgl. ALT+E).
ALT+B	Steht der Cursor in einem Satz, wird auf diesen Satz ein Lesezeichen (bookmark) gesetzt; steht er in der Anweisungszeile, wird auf die aktuelle Satzposition ein Lesezeichen gesetzt.
ALT+C	Zeigt die früheren Inhalte der Zwischenablage an; es werden jedoch nur diejenigen angezeigt, die mit TUSTEP in die Zwischenablage geschrieben bzw. aus der Zwischenablage gelesen wurden.  Soll ein früherer Inhalt wieder in die Zwischenablage geholt werden, so muss dieser mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Return-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der linken Maustaste angeklickt werden (siehe auch RECALL_CB Seite 393).



- ALT+D Zeigt den Anfang aller Sätze mit Lesezeichen an; die Reihenfolge entspricht der Reihenfolge in der die Lesezeichen verwendet wurden (vgl. ALT+I).
- Soll einer der Sätze mit Umgebung angezeigt werden, so muss dieser mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Return-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der linken Maustaste angeklickt werden.
- ALT+E Sucht im Textfeld ab der Cursor-Position rückwärts nach einem Anfangs-Tag, zu dem im durchsuchten Text kein Ende-Tag vorhanden ist, und fügt an der Cursor-Position das entsprechende Ende-Tag ein (vgl. ALT+A).
- ALT+F Zeigt den Satz mit Umgebung an, dessen Lesezeichen als nächstes zuvor (früher) verwendet wurde (vgl. ALT+S).
- ALT+G Wenn der Cursor außerhalb eines Tags steht, wird eine Liste der Tags angezeigt, die an der Cursor-Position in der Tag-Hierarchie erlaubt sind; wenn der Cursor innerhalb eines Tags steht, wird eine Liste der Attribute angezeigt, die in dem betreffenden Tag erlaubt sind.
- Soll eines dieser Tags bzw. Attribute an der Cursor-Position eingefügt werden, so muss dieses mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Return-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der linken Maustaste angeklickt werden (siehe auch SELECT\_TAG Seite 385).
- ALT+H Zeigt das Inhaltsverzeichnis der eingestellten Segment-Datei (siehe Seite 280) an.
- Soll ein Segment in die Editor-Datei geholt werden, so muss dieses mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Return-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der linken Maustaste angeklickt werden (siehe auch FETCH\_SEGM Seite 386).
- ALT+I Zeigt den Anfang aller Sätze mit Lesezeichen an; die Reihenfolge entspricht der Reihenfolge in der Datei (vgl. ALT+D).
- Soll einer der Sätze mit Umgebung angezeigt werden, so muss dieser mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Return-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der linken Maustaste angeklickt werden.
- ALT+J
- bei Dateneingabe (z. B. nach der Anweisung ee): Hängt die aktuelle Zeile mit allen Zeilen von der vorangehenden Leerzeile bis zur nachfolgenden Leerzeile zusammen.
  - sonst: Hängt den Satz, in dem der Cursor steht, mit dem im Textfeld unmittelbar vorangehenden Satz zusammen.

- ALT+K Zeigt die Sätze, die durch Ändern im Textfeld in der Editor-Datei geändert wurden, mit dem Inhalt an, den sie vor der Änderung hatten.
- Soll ein Satz wieder hergestellt werden, so muss dieser mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Return-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der linken Maustaste angeklickt werden (siehe auch `RECALL_MOD` Seite 373).
- ALT+L Zeigt die Sätze, die durch Löschen im Textfeld (z. B. mit `DEL_REC`) in der Editor-Datei gelöscht wurden, mit dem Inhalt an, den sie zuvor hatten.
- Soll ein Satz wieder hergestellt werden, so muss dieser mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Return-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der linken Maustaste angeklickt werden (siehe auch `RECALL_DEL` Seite 372).
- ALT+M Zeigt die früheren Inhalte des Editor-Zwischenspeichers an.
- Soll ein früherer Inhalt wieder in den Editor-Zwischenspeicher geholt werden, so muss dieser mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Return-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der linken Maustaste angeklickt werden (siehe auch `RECALL_MRK` Seite 372).
- ALT+N Die aktuelle Zeile und alle folgenden Zeilen werden um eine Zeile nach unten verschoben. Dadurch entsteht eine Leerzeile, in die anschließend Text eingetragen werden kann. Falls möglich, wird die neue Zeile automatisch mit einer noch freien Satznummer versehen. Der Cursor wird um ebensoviele Leerstellen eingerückt, wie der Text des im Textfeld davor stehenden Satzes eingerückt ist.
- ALT+O Zeigt den Satz mit Umgebung an, der als nächster vor (oberhalb) der aktuellen Satzposition ein Lesezeichen hat (vgl. `ALT=U`).
- ALT+P Löscht den Namen, in dem bzw. hinter dem der Cursor steht, ergänzt den Namen um einen Abkürzungspunkt und ruft das Makro mit diesem Namen auf. Falls der Cursor nicht in oder hinter einem Namen steht, wird das Makro mit dem Namen ».« aufgerufen.
- ALT+Q Zeigt Sonderzeichen und Umlaute an, die auf manchen Tastaturen fehlen oder schwer erreichbar sind.
- Soll eines dieser Zeichen an der Cursor-Position eingefügt werden, so muss es mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Return-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der linken Maustaste angeklickt werden.
- ALT+R Sucht von der Cursor-Position an zum Dateianfang hin (rückwärts) nach dem nächsten Anfangs-Tag, für das in den durchsuchten Daten kein entsprechendes Ende-Tag vorhanden ist, und zeigt den betreffenden Satz an (vgl. `ALT+V`). Steht der Cursor in der Anweisungszeile, so wird von der aktuellen Tag-Position an gesucht (vgl. »Suchen von Tags« Seite 321).

ALT+S	Zeigt den Satz mit Umgebung an, dessen Lesezeichen als nächstes danach (später) verwendet wurde (nur unmittelbar nach ALT+F möglich).
ALT+T	Prüft die Tags vom Dateianfang an bis zur Cursor-Position und zeigt die an der Cursor-Position noch offenen Tags an. Steht der Cursor in der Anweisungszeile, so werden die Tags bis zur aktuellen Satzposition geprüft (vgl. »Anzeigen offener Tags« Seite 318).
ALT+U	Zeigt den Satz mit Umgebung an, der als nächster nach (unterhalb) der aktuellen Satzposition ein Lesezeichen hat (vgl. ALT+O).
ALT+V	Sucht von der Cursor-Position an zum Dateiende hin (vorwärts) nach dem nächsten Ende-Tag, für das in den durchsuchten Daten kein entsprechendes Anfangs-Tag vorhanden ist, und zeigt den betreffenden Satz an (vgl. ALT+R). Steht der Cursor in der Anweisungszeile, so wird von der aktuellen Tag-Position an gesucht (vgl. »Suchen von Tags« Seite 321).
ALT+W	Falls eine Segment-Datei ediert wird und der Cursor in einem zu einem Segment gehörenden Satz steht bzw. der Cursor in der Anweisungszeile steht und die aktuelle Satzposition auf eine zu einem Segment gehörenden Satz zeigt, wird der Name dieses Segments in der Statuszeile angezeigt.
ALT+X	Falls Text markiert ist, wird der markierte Text in die Liste der gemerkten Textteile aufgenommen; ist kein Text markiert, so wird der Anfang aller gemerkten Textteile angezeigt.  Soll einer dieser Textteile an der aktuellen Cursor-Position eingefügt werden, so muss dieser mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Return-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der linken Maustaste angeklickt werden (siehe SELECT_TEXT Seite 359).
ALT+Y	Zeigt den Anfang aller Makros an, deren Namen mit einem Abkürzungspunkt endet (vgl. ALT+P).  Soll eines dieser Makros ausgeführt werden, so muss dieses mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Return-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der linken Maustaste angeklickt werden (siehe auch SELECT_ABBR Seite 377).
ALT+Z	Zeigt die mit der Delete-Taste und/oder Backspace-Taste gelöschten Zeichenfolgen an.  Soll eine dieser Zeichenfolge wieder zurückgeholt werden, so dass sie mit <code>Ctrl+Z</code> wieder eingefügt werden kann, so muss diese mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Return-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der linken Maustaste angeklickt werden (siehe auch RECALL_CHAR Seite 362).

Im Kapitel »Steuerbefehle im Editor« ab Seite 350 sind alle Steuerbefehle beschrieben. Die Steuerbefehle können in Editormakros verwendet werden. Falls ein Steuerbefehl auch durch eine Tastenkombination aufgerufen werden kann, ist sie dort jeweils angegeben.

## Mausaktionen (Auswahl)

Die im Folgenden beschriebenen Mausaktionen haben nur die genannte Wirkung, wenn für die entsprechenden Makros die voreingestellten Definitionen nicht geändert wurden.

Nachdem ein Text mit der linken Maustaste markiert wurde (linke Maustaste am Anfang des zu markierenden Textteils drücken, nicht loslassen, Maus auf das Ende des Textteils ziehen und Maustaste loslassen), werden unten in der Statuszeile die nachfolgend aufgeführten Felder (Schaltflächen) angezeigt. Wird eines dieser Felder mit der linken Maustaste angeklickt, so hat dies jeweils die angegebene Wirkung:

**FIND**

Sucht ab der Cursor-Position zeilenweise nach einem Text im Textfeld des Editorfensters, der mit dem markierten Text übereinstimmt, und positioniert den Cursor an den Anfang des gefundenen Textes. Groß- und Kleinbuchstaben werden bei der Suche nicht unterschieden.

**< COPY**

Kopiert den markierten Text in den Editor-Zwischenspeicher. Falls der Editor-Zwischenspeicher schon Text enthält, wird der markierte Text vor diesen in den Editor-Zwischenspeicher eingeschoben.

**COPY**

Kopiert den markierten Text in den Editor-Zwischenspeicher. Falls der Editor-Zwischenspeicher schon Text enthält, wird dieser vorher gelöscht.

**COPY >**

Kopiert den markierten Text in den Editor-Zwischenspeicher. Falls der Editor-Zwischenspeicher schon Text enthält, wird der markierte Text an diesen im Editor-Zwischenspeicher angehängt.

**DELETE**

Löscht den markierten Text, ohne ihn im Editor-Zwischenspeicher zu merken.

**< MOVE**

Kopiert den markierten Text in den Editor-Zwischenspeicher und löscht den markierten Text im Editorfenster. Falls der Editor-Zwischenspeicher schon Text enthält, wird der markierte Text vor diesen in den Editor-Zwischenspeicher eingeschoben.

**MOVE**

Kopiert den markierten Text in den Editor-Zwischenspeicher und löscht den markierten Text im Editorfenster. Falls der Editor-Zwischenspeicher schon Text enthält, wird dieser vorher gelöscht.

MOVE &gt;

Kopiert den markierten Text in den Editor-Zwischenspeicher und löscht den markierten Text im Editorfenster. Falls der Editor-Zwischenspeicher schon Text enthält, wird der markierte Text an diesen im Editor-Zwischenspeicher angehängt.

IGNORE

Hebt die Markierung auf.

Wird der Text nicht mit der linken Maustaste, sondern mit der rechten markiert, so wird der markierte Text gelöscht und der Inhalt des Editor-Zwischenspeichers an dieser Stelle eingefügt.

Wird mit der linken Maustaste an eine Stelle im Textfeld oder in der Anweisungszeile geklickt, so wird der Cursor an diese Stelle positioniert.

Wird die Shift-Taste niedergehalten und dabei mit der linken Maustaste (z. B. nach einer Zeige-Nur-Anweisung) auf einen Satz im Textfeld geklickt, so werden dieser Satz und die umgebenden Sätze davor und danach angezeigt.

Wird mit der rechten Maustaste an eine Stelle im Textfeld oder in der Anweisungszeile geklickt, so wird an dieser Stelle der Inhalt des Editor-Zwischenspeichers eingefügt; alle in der Zeile folgenden Zeichen werden nach rechts verschoben.

Wird mit der mittleren Maustaste auf ein Tag (eine in spitzen Klammern eingeschlossene Zeichenfolge) geklickt, so wird das Tag in den Editor-Zwischenspeicher kopiert und das Tag im Editorfenster gelöscht. Falls der Editor-Zwischenspeicher schon Text enthält, wird dieser vorher gelöscht.

Mit dem Mausrad kann der Text im Textfeld nach unten bzw. oben verschoben werden. Bei geteiltem Textfeld wird jeweils nur das Textfeld verschoben, in dem der Mauszeiger steht. Wird zusätzlich die Shift-Taste gedrückt, so werden beide Textfelder gleichzeitig verschoben.

Alle Mausaktionen bewirken einen Aufruf eines Editormakros. Welche Makros dies sind, ist im Kapitel »Mausaktionen für Makroaufrufe« auf Seite 344 beschrieben.

## Eingabehilfe

Zur Erleichterung der Eingabe von Zeichen- und Stringgruppen, Häufigkeitsangaben, Verweisen usw. in Such- und Austausch-Tabellen wird für die Tastenkombination `Ctrl+K-x` bzw. `Strg+K-x` (Taste »K« drücken, während die `Ctrl`- bzw. `Strg`-Taste niedergehalten wird, danach Taste »x« drücken) an der Cursor-Position die Zeichenfolge »{x}« ausgegeben. Dabei kann x eines der Zeichen

+ , - , = , ! , ; , @ , % , \ , & , # , [ , ] , | , : oder eine Ziffer

sein. Wird für x »<<« oder »>>« angegeben, so wird die Zeichenfolge »{S:}« bzw. »{C:}« ausgegeben. Wird für x ein Leerzeichen angegeben, so wird eine Liste mit den speziellen Codierungen für Such- und Austausch-Tabellen angezeigt (vgl. »Codierungen mit Sonderfunktion« Seite 704). Die Anzeige kann durch Drücken der Leertaste oder Escape-Taste wieder gelöscht werden.

## Editorfenster

Nach dem Aufruf des Editors wird das Editorfenster angezeigt.

```

Makroleiste mit den selbst definierten Feldern
DATEN*BEISPIEL
*****
1.1      _____ Dateianfang _____
1.2      Kurzer Satz in einer Zeile. Es folgt ein
         langer Satz, der nicht in eine einzige Zeile passt. Für seine
         Anzeige sind deshalb noch zwei Fortsetzungszeilen notwendig.
         Danach folgt
1.3      wieder ein kurzer Satz, der in eine Zeile passt.
         _____ Dateiende _____
*****
.....
.....
*=1.1 Gib Anweisung >
**;01          SPLIT  SCROLL  INSERT  MOUSE

```

Es besteht aus

- einer Menü-Zeile (mit einer Makroleiste), falls beim Aufruf des Editor eine namenlose permanente Makroleiste (»~«) definiert ist; andernfalls entfällt diese Zeile
- einer Kopfzeile, in der angezeigt wird, welche Daten ediert werden.
- einem Textfeld, in dem die Daten angezeigt werden.
- einer Meldungszeile, in der z. B. Fehlermeldungen angezeigt werden.
- einer Anweisungszeile, in der die aktuelle Satzposition angezeigt wird und die Anweisungen angefordert werden.
- einer Statuszeile, in der die Cursor-Position und verschiedene Einstellungen angezeigt werden.

## Darstellung der Sätze im Textfeld

Jeder Satz wird nach Möglichkeit in einer Zeile angezeigt. Sätze, die nicht in eine Zeile passen, werden mit Fortsetzungszeilen angezeigt.

Eine Fortsetzungszeile wird nach Möglichkeit an einem Leerzeichen (Wortzwischenraum) begonnen. Ist kein geeignetes Leerzeichen in den Daten vorhanden, wird die Zeile möglichst voll geschrieben und dann eine Fortsetzungszeile begonnen. Um anzuzeigen, dass am Zeilenumbruch kein Leerzeichen in den Daten vorhanden ist, wird die Zeile mit dem Zeichen Gravis » ` « abgeschlossen.

Wenn die Anzeige der Satznummern nicht (durch eine entsprechende Moduseinstellung) abgeschaltet ist, wird am Anfang jeder ersten Zeile eines Satzes die Satznummer des jeweiligen Satzes angezeigt. Eine Fortsetzungszeile ist daran zu erkennen, dass sie keine Nummer hat (siehe Abbildung im Abschnitt »Editorfenster« Seite 270).

## Nummerierung der Sätze

Jeder Satz in einer TUSTEP-Datei hat eine Satznummer. Wie die Sätze im Programm- bzw. Textmodus nummeriert werden, und wie die Satznummern interpretiert werden, ist unter »Nummerierung im Programmmodus« (Seite 33) bzw. unter »Nummerierung im Textmodus« (Seite 33) beschrieben.

Damit eine Datei mit dem Editor bearbeitet werden kann, müssen noch folgende Bedingungen erfüllt sein:

- Die Satznummer 0/0 bzw. 0.0/0 darf nicht verwendet werden.
- Die Sätze müssen so nummeriert sein, dass alle Satznummern aufsteigend sind.
- Keine Satznummer darf mehrfach vorkommen.

Standardmäßig beginnt die Nummerierung bei 1 bzw. bei 1.1. Es dürfen jedoch auch kleinere Satznummern (z. B. 0/1 bzw. 0.1) verwendet werden, um z. B. vor dem ersten Satz neue Sätze einzufügen.

## Änderungen im Textfeld

Wenn die Daten im Textfeld geändert werden, so werden diese Änderungen nicht sofort in die Datei übertragen, sondern erst wenn

- die Änderungen mit der Enter-Taste oder einem der Steuerbefehle `ENTER` oder `CONFIRM` bestätigt werden,
- eine Funktionstaste gedrückt wird,
- das Textfeld (z. B. mit dem Mausrad) gescrollt wird,
- im Textfeld (z. B. mit einem der Steuerbefehle `SHW_UP`, `SHW_DN` oder einem anderen mit `SHW` beginnenden Steuerbefehl) Daten angezeigt werden,
- eine Anweisung in der Anweisungszeile abgeschickt wird,
- bei geteiltem Textfeld (z. B. mit einem Mausklick) in das andere (nicht-aktive) Textfeld gewechselt wird.

Außerdem wird ein im Textfeld geänderter Satz in die Datei übertragen, wenn er aus dem Textfeld hinausgeschoben werden muss, weil an anderer Stelle im Textfeld zusätzliche Daten eingefügt werden.

Sollen die Änderungen im Textfeld nicht in die Datei übertragen werden, so kann das Textfeld mit dem Steuerbefehl `CLEAR` (siehe Seite 365) gelöscht werden.

## Aufruf der Online-Hilfe

Die Online-Hilfe kann mit dem Kommando `#HILFE` (siehe Seite 162) aufgerufen werden.

Innerhalb des des Editors kann sie mit der Anweisung `hilfe` oder mit dem Steuerbefehl `HELP` (Tastenkombination `Ctrl+O` bzw. `Strg+O`) aufgerufen werden. Wird die Online-Hilfe mit dem Steuerbefehl `HELP` aufgerufen, so ist der Editor nach dem Be-

enden der Online-Hilfe im selben Zustand wie vor dem Aufruf der Online-Hilfe (so als wäre die Online-Hilfe nicht aufgerufen worden).

## Reorganisieren der Dateien

Dateien, die mit dem Editor geändert werden, sollten ab und zu reorganisiert werden, wie dies unter »Reorganisieren von Dateien« (Seite 40) beschrieben ist.

## Hinweise

Die im Folgenden verwendeten Kürzel `pos`, `ber`, `begr`, `stab`, `vtab`, `atab`, `rtab` sind ab Seite 328 erklärt.

Ist die Editor-Datei nur zum Lesen angemeldet, wird beim Versuch, die Daten zu ändern, eine entsprechende Fehlermeldung angezeigt und es wird nachgefragt, ob der Editor für diese Datei Schreiberlaubnis erhalten soll. Wird diese Frage bejaht, arbeitet der Editor bis zum Verlassen dieser Datei so, als wäre sie zum Schreiben angemeldet.

Bei allen Anweisungen, die mit einem Buchstaben beginnen, kann das erste Komma weggelassen werden, wenn nicht unmittelbar danach ein weiteres Komma folgt.

Anweisungen, die länger als eine Zeile sind, können in das Textfeld geschrieben werden. Falls dort Daten stehen, können diese zuvor mit `CLEAR` (siehe Seite 365) gelöscht werden. Werden für die Anweisung Fortsetzungszeilen benötigt, so brauchen diese nicht gekennzeichnet zu werden. Zu beachten ist, dass bei Anweisungen Leerzeichen an den Zeilenenden ignoriert werden.



## Einfache Anweisungen

### Beenden

Der Editor muss entweder mit einer der beiden folgenden Anweisungen oder mit der Escape-Taste beendet werden! Andernfalls wird eine zum Schreiben angemeldete Datei nicht ordnungsgemäß abgeschlossen.

Daten aus nicht abgeschlossenen Dateien können jedoch in der Regel mit dem Kommando #RETTE (siehe Seite 216) in eine andere Datei kopiert und damit wieder zugänglich gemacht werden.

**b** Beenden des Editors.

**b, komm** Beenden des Editors und Ausführen des Kommandos `komm`.

Die TUSTEP-Kommandos `komm` werden unmittelbar nach dem Beenden des Editors ausgeführt. Für die Angabe der Kommandos gelten die gleichen Regeln wie bei der Anweisung **x** (siehe Seite 303).

### Eintragen, Einfügen

**ee** Eintragen in eine leere Datei oder ans Ende der Datei.

Jede Zeile im Textfeld ergibt einen Satz in der Datei. Das Eintragen kann durch eine leere Eingabe mit `Enter` oder mit einer Funktionstaste beendet werden.

Hinweis: Falls mit dem Steuerbefehl `CHG_SETTINGS` (Seite 395) der `ENTER`-Modus eingestellt wurde, wirkt der Steuerbefehl `CR` während der Dateneingabe wie der Steuerbefehl `LF`; d. h. der Cursor springt an den Anfang der nächsten Zeile. Um die Daten abzuschicken, muss der Steuerbefehl `ENTER` benutzt werden.

**e, pos#n** Einfügen von `n` Sätzen, beginnend mit der Satzposition `pos` (falls noch kein Satz mit der Satznummer `pos` vorhanden ist) bzw. nach der Satzposition `pos` (falls schon ein Satz mit der Satznummer `pos` vorhanden ist).

`n` kann ein geschätztes Maximum sein. Falls `n=1` ist, kann `#n` wegfallen. Jede Zeile im Textfeld ergibt einen Satz in der Datei. Sollen weniger Sätze als angegeben eingefügt werden, kann das Eintragen durch eine leere Eingabe mit `Enter` oder mit einer Funktionstaste beendet werden.

Hinweis: Falls mit dem Steuerbefehl `CHG_SETTINGS` (Seite 395) der `ENTER`-Modus eingestellt wurde, wirkt der Steuerbefehl `CR` während der Dateneingabe wie der Steuerbefehl `LF`; d. h. der Cursor springt an den Anfang der nächsten Zeile. Um die Daten abzuschicken, muss der Steuerbefehl `ENTER` benutzt werden.

Wie die eingefügten Sätze nummeriert werden, hängt davon ab, wieviele Satznummern an dieser Stelle frei sind. Als Schrittweite der Nummerierung wird die größtmögliche der Schrittweiten `1/0`, `0/1`, `0/01`, `0/001` (letz-

tere nur im Textmodus) gewählt. Sollen die Sätze mit einer anderen Schrittweite nummeriert werden, kann diese hinter `pos` angegeben werden. `pos` und Schrittweite müssen durch einen Strichpunkt getrennt werden (z. B. `23;0/2#30`). Als Schrittweite darf eine Zeilennummer und/oder eine Unterscheidungsnummer angegeben werden. Ist die Nummerierung in der angegebenen Schrittweite nicht möglich, weil nicht genügend Satznummern frei sind, so wird die angegebene Schrittweite ignoriert und eine kleinere gewählt.

**ez, pos** Einfügen der Daten aus der Zwischenablage, beginnend mit der Satzposition `pos` (falls noch kein Satz mit der Satznummer `pos` vorhanden ist) bzw. nach der Satzposition `pos` (falls schon ein Satz mit der Satznummer `pos` vorhanden ist).

Die eingefügten Sätze werden in gleicher Weise wie bei der zuvor beschriebenen Anweisung `e, pos#n` nummeriert.

**ez, (pos1, pos2)** Eintragen der Sätze von der Satzposition `pos1` bis zur Satzposition `pos2` in die Zwischenablage. Enthält die Zwischenablage schon Daten, werden sie überschrieben.

**ei, text, ber:spa** Einfügen der Zeichenfolge `text` in jedem Satz des Bereichs `ber`.

Die Zeichenfolge `text` muss durch ein Begrenzungszeichen, das frei wählbar ist, am Anfang und am Ende begrenzt werden. Dieses Begrenzungszeichen darf innerhalb der Zeichenfolge nicht vorkommen. Für die Codierung der Zeichenfolge `text` gelten die gleichen Regeln wie für Daten (siehe »Codierung der Zeichen« Seite 259).

Die Angabe `:spa` kann fehlen. Das bedeutet, dass die Zeichenfolge `text` jeweils am Ende eines Satzes eingefügt wird. Wird für `spa n` angegeben, so wird die Zeichenfolge jeweils ab der Position `n` eines Satzes eingefügt (d. h. die Zeichen von der Position `n` bis zum Ende des Satzes werden nach rechts geschoben). Die Angabe `+n` bewirkt, dass jeweils nach dem `n`-ten Zeichen vom Satzanfang an eingefügt wird. Entsprechend bewirkt die Angabe `-n`, dass jeweils vor dem `n`-ten Zeichen vom Satzende her eingefügt wird.

Wird für `spa n1-n2` angegeben, so wird die Zeichenfolge in den Spalten `n1` bis `n2` durch die Zeichenfolge `text` ersetzt. Die Zeichen nach der Spalte `n2` bis zum Satzende werden dabei nach links bzw. nach rechts verschoben, falls die Zeichenfolge `text` kürzer bzw. länger ist als die Zeichenfolge in den Spalten `n1` bis `n2`. Soll nur das Zeichen in Spalte `n` ersetzt werden, so ist dafür `n-n` (nicht `n`) anzugeben.

**ev, ber1, ber2** Einfügen der Sätze des Bereichs `ber1` vor jedem Satz des Bereichs `ber2`.

**en, ber1, ber2** Einfügen der Sätze des Bereichs `ber1` nach jedem Satz des Bereichs `ber2`.

Enthält der Bereich, in dem eingefügt werden soll, mehr als 10 Sätze, so wird vor dem Einfügen zurückgefragt, ob bei soundsovielen Sätzen ein-

gefügt werden soll. Als Antwort kann j (ja) oder n (nein) oder eine neue Anweisung (auch in diesem Fall wird nicht eingefügt) gegeben werden. Die Anzahl der Sätze, bis zu der diese Anfrage unterbleiben soll, kann, durch ein Nummernzeichen getrennt, hinter `ber:spa` bzw. `ber2` angegeben werden (z. B. `(23, -1) #30`).

## Zeigen

<b>za</b>	Zeigen der Sätze vom Dateianfang an.
<b>ze</b>	Zeigen der Sätze vom Dateende her.
<b>za, pos</b>	Zeigen der Sätze ab der Satzposition <code>pos</code> .
<b>zu, pos</b>	Zeigen der Sätze um die Satzposition <code>pos</code> .
<b>zb, pos</b>	Zeigen der Sätze bis zu der Satzposition <code>pos</code> .
<b>z, pos</b>	Zeigen der Sätze ab der Satzposition <code>pos</code> / der Sätze um die Satzposition <code>pos</code> / der Sätze bis zu der Satzposition <code>pos</code> je nachdem, welche der Anweisungen <b>za, pos</b> / <b>zu, pos</b> / <b>zb, pos</b> zuletzt gegeben wurde. Wurde noch keine von diesen Anweisungen gegeben, so wirkt <b>z, pos</b> wie <b>zu, pos</b> .

Nach den Zeige-Anweisungen sind folgende Eingaben möglich:

leere Eingabe: Weiterblättern.

- + Vorwärts weiterblättern; danach bewirkt eine leere Eingabe jeweils ein Weiterblättern zum Dateende hin.
- Rückwärts weiterblättern; danach bewirkt eine leere Eingabe jeweils ein Weiterblättern zum Dateianfang hin.

andere Anweisung: Das Zeigen wird unterbrochen und diese Anweisung ausgeführt.

## Korrigieren

Jeder vom Editor (z. B. durch eine Zeige-Anweisung) mit einer Satznummer angezeigte Satz kann unmittelbar zur Korrektur verwendet werden. Es genügt, ihn nach dem Ändern abzuschicken. Es können auch mehrere Sätze geändert und zusammen abgeschickt werden. Trotzdem ist in vielen Fällen eine eigene Anweisung zum Korrigieren sinnvoll:

**ber** Zeigen eines Satzes bzw. eines Bereichs zur Korrektur.

Die Verwendung dieser Anweisung bietet gegenüber einer Zeige-Anweisung folgende Vorteile:

- Soll z. B. nur ein Satz korrigiert werden, dessen Satznummer bekannt ist, so genügt es, mit dieser Anweisung diesen einen Satz anzeigen zu lassen, statt mit einer Zeige-Anweisung ein ganzes Fenster voll anzeigen zu lassen.
- Wenn ein größerer Bereich korrigiert werden soll, so müssten mit ei-

ner weiteren Zeige-Anweisung jeweils die nächsten Sätze angefordert werden, während mit dieser Anweisung die nachfolgenden Sätze automatisch angezeigt werden, nachdem die korrigierten Sätze abgeschickt wurden.

In der Datei werden jeweils nur die nach der Änderung abgeschickten Sätze geändert; nicht abgeschickte Sätze werden nicht geändert! Insbesondere bewirkt ein Löschen eines Satzes samt der dazugehörigen Satznummer im Textfeld kein Löschen dieses Satzes in der Datei.

Soll einer oder mehrere der im Textfeld gezeigten Sätze gelöscht werden, so kann der hinter der jeweiligen Satznummer stehende senkrechte Strich durch ein Minuszeichen ersetzt werden; der zu dem Satz gehörende Text muss im Textfeld gelöscht werden (vgl. dazu Steuerbefehl `DEL_REC` Seite 364). In der Datei werden solche Sätze erst gelöscht, nachdem sie (ggf. zusammen mit anderen geänderten Sätzen) abgeschickt wurden.

Sollen zwei oder mehrere im Textfeld aufeinander folgende Sätze zusammengehängt werden, so muss der hinter der Satznummer des zweiten Satzes und der ggf. folgenden Sätze stehende senkrechte Strich durch ein Minuszeichen ersetzt werden (vgl. dazu Steuerbefehl `JOIN` Seite 398). In der Datei werden solche Sätze erst zusammengehängt, nachdem sie (ggf. zusammen mit anderen geänderten Sätzen) abgeschickt wurden.

Sollen zu den im Textfeld gezeigten Sätzen ein oder mehrere Sätze hinzugefügt werden, so können diese neuen Sätze einschließlich der neuen Satznummern im Textfeld eingefügt werden; zwischen Satznummer und Text steht in diesem Fall ein Pluszeichen statt des senkrechten Striches (vgl. dazu Steuerbefehl `INS_LINE` Seite 360 und `SPLIT` Seite 360). In der Datei werden solche Sätze erst eingefügt, nachdem sie (ggf. zusammen mit anderen geänderten Sätzen) abgeschickt wurden.

Bei Korrekturen kann an Stelle des senkrechten Striches zwischen Satznummer und Text auch das Gleichheitszeichen verwendet werden.

Nachfolgend sind einige Beispiele angegeben, bei denen die im Textfeld vorgenommenen Änderungen möglicherweise nicht die beabsichtigte Wirkung haben. Zuvor noch ein wichtiger Hinweis:

Entscheidend für die Wirkung der Änderungen ist nur der Inhalt des Textfeldes zum Zeitpunkt der Bestätigung der Änderungen. Der Inhalt des Textfeldes, den er vor den Änderungen hatte, hat keine Bedeutung mehr; es findet auch kein Vergleich zwischen den Fensterinhalten vor und nach den Änderungen statt, um die Unterschiede festzustellen und die Datei entsprechend zu ändern.

Vor der Änderung:	Nach der Änderung:	Ergebnis in der Datei:
1.1   eins	1.2   zwei	1.1 eins
1.2   zwei	drei	1.2 zwei drei
drei		

Satz 1.1 bleibt unverändert. Grund: Dieser Satz (genauer: diese Satznummer) steht

nicht (mehr) im Textfeld, folglich bleibt er unverändert wie alle anderen Sätze der Datei, die auch nicht im Textfeld stehen. Satz 1.2 bleibt unverändert. Grund: Dieser Satz enthält keine Änderungen. Falls beabsichtigt ist, den Satz 1.1 samt Inhalt zu löschen, müsste (z. B. mit dem Steuerbefehl `DEL_REC`, siehe Seite 364) die Satznummer mit einem »-« gekennzeichnet werden und der zum Satz gehörende Text im Textfeld gelöscht werden; eine weitere Möglichkeit wäre, diesen Satz mit der Anweisung »!1.1« zu löschen.

Vor der Änderung:	Nach der Änderung:	Ergebnis in der Datei:
1.1   eins	1.1   eins	1.1 eins zwei drei
1.2   zwei	zwei	1.2 zwei drei
drei	drei	

Satz 1.1 wird um den Text »zwei drei« verlängert. Grund: Vor »zwei« und vor »drei« steht keine Satznummer, folglich sind diese Zeilen jeweils Fortsetzungszeilen der vorangehenden Zeile. Satz 1.2 bleibt unverändert. Grund: Dieser Satz (genauer: diese Satznummer) steht nicht (mehr) im Textfeld, folglich bleibt er unverändert wie alle anderen Sätze der Datei, die auch nicht im Textfeld stehen. Falls beabsichtigt ist, den Inhalt von Satz 1.2 an den von Satz 1.1 anzuhängen und den Satz 1.2 zu löschen, müsste die Satznummer 1.2 mit einem »-« gekennzeichnet werden.

Vor der Änderung:	Nach der Änderung:	Ergebnis in der Datei:
1.1   eins	1.1   eins	1.1 eins zwei drei
1.2   zwei	1.2 - zwei	
drei	drei	

Satz 1.2 wird gelöscht. Grund: Die Satznummer ist mit einem »-« gekennzeichnet. Satz 1.1 wird um den Text »zwei drei« verlängert. Grund: Die Satznummer 1.2 ist mit einem »-« gekennzeichnet und dieser Satz wird gelöscht. Deshalb gilt der noch dahinter stehende Text »zwei« als Fortsetzung der vorangehenden Zeile, so als enthielte die Zeile mit dem Text »zwei« keine Satznummer. Vor »drei« steht keine Satznummer, folglich ist diese Zeile Fortsetzungszeile der vorangehenden Zeile. Falls beabsichtigt ist, den Satz 1.2 samt Inhalt zu löschen, müsste (z. B. mit dem Steuerbefehl `DEL_REC`, siehe Seite 387) die Satznummer mit »-« gekennzeichnet werden und der dazugehörige Text »zwei« und »drei« im Textfeld gelöscht werden.

## Löschen

**![,ber** Löschen eines Satzes bzw. eines Bereichs.

Enthält der zu löschende Bereich mehr als 10 Sätze, so wird, bevor die Sätze gelöscht werden, zurückgefragt, ob soundsoviele Sätze gelöscht werden sollen. Als Antwort kann `j` (ja) oder `n` (nein) oder eine neue Anweisung (auch in diesem Fall wird nicht gelöscht) gegeben werden. Die Anzahl der Sätze, bis zu der diese Anfrage unterbleiben soll, kann, durch ein Nummernzeichen getrennt, hinter `ber` angegeben werden (z. B. `(23,-1)#30`).

**1!!** wie **1!** (ohne Bereichsangabe); außer dem Dateiinhalte werden zusätzlich die eingestellten Namen für `datei` und `segment` (vgl. »Namen abfragen/einstellen/löschen« Seite 280) sowie der Dateititel gelöscht (vgl. »Dateititel abfragen/einstellen/löschen« Seite 286).

## Kopieren

**k,ber,pos** Kopieren eines Satzes bzw. eines Bereichs.

**k,datei,ber,pos** Kopieren eines Satzes bzw. eines Bereichs aus einer anderen Datei.

**k,datei,segment,pos** Kopieren eines Segments aus einer anderen Datei.

Der Satz bzw. der Bereich `ber` bzw. das Segment `segment` wird so kopiert, dass er beginnend mit der Satzposition `pos` (falls noch kein Satz mit der Satznummer `pos` vorhanden ist) bzw. nach der Satzposition `pos` (falls schon ein Satz mit der Satznummer `pos` vorhanden ist) zusätzlich in der Editor-Datei steht. Falls ans Dateieende kopiert werden soll, kann `pos` wegfallen. Im Textmodus erhält in diesem Fall der erste neue Satz die Nummer `n.1`, wobei `n` die um 1 erhöhte Seitennummer des letzten Satzes der Datei ist.

Wie die neuen Sätze nummeriert werden, hängt davon ab, wieviele Satznummern an dieser Stelle frei sind. Als Schrittweite der Nummerierung wird die größtmögliche der Schrittweiten `1/0`, `0/1`, `0/01`, `0/001` (letztere nur im Textmodus) gewählt. Sollen die Sätze mit einer anderen Schrittweite nummeriert werden, kann diese hinter `pos` angegeben werden. `pos` und Schrittweite müssen durch einen Strichpunkt getrennt werden (z. B. `23;0/2`). Als Schrittweite darf eine Zeilennummer und/oder eine Unterscheidungsnummer angegeben werden. Ist die Nummerierung in der angegebenen Schrittweite nicht möglich, weil nicht genügend Satznummern frei sind, so wird die angegebene Schrittweite ignoriert und eine kleinere gewählt.

Falls das Kopieren nicht möglich ist, weil nicht genügend Satznummern frei sind, wird nachgefragt, ob nachfolgende Sätze umnummeriert werden dürfen, damit genügend Satznummern frei werden. Wird diese Frage bejaht, werden zusätzlich entsprechend viele nachfolgende Sätze umnummeriert, andernfalls unterbleibt das Kopieren.

Enthält der zu kopierende Bereich bzw. das zu kopierende Segment mehr als 10 Sätze, so wird, bevor die Sätze kopiert werden, zurückgefragt, ob soundsoviele Sätze kopiert werden sollen. Als Antwort kann `j` (ja) oder `n` (nein) oder eine neue Anweisung (auch in diesem Fall wird nicht kopiert) gegeben werden. Die Anzahl der Sätze, bis zu der diese Anfrage unterbleiben soll, kann, durch ein Nummernzeichen getrennt, hinter `ber` bzw. `segment` angegeben werden (z. B. `(23,-1)#30`).

## Umnummerieren, Umstellen

**u!** Umnummerieren aller Sätze der aktuellen Seite, im Programmmodus aller Sätze der Datei. Die neue Nummerierung beginnt mit 1. Die aktuelle Satzposition wird beibehalten und die Satznummer angepasst.

**u,ber,pos** Umnummerieren und ggf. Umstellen eines Satzes bzw. eines Bereichs.

Der Satz bzw. der Bereich `ber` wird so umnummeriert und ggf. umgestellt, dass er beginnend mit der Satzposition `pos` (falls noch kein Satz mit der Satznummer `pos` vorhanden ist) bzw. nach der Satzposition `pos` (falls schon ein Satz mit der Satznummer `pos` vorhanden ist) in der Datei steht. Falls ans Dateiende umgestellt werden soll, kann `pos` wegfallen. Im Textmodus erhält in diesem Fall der erste umnummerierte/umgestellte Satz die Nummer `n.1`, wobei `n` die um 1 erhöhte Seitennummer des letzten Satzes der Datei ist. Wird jedoch `ber` weggelassen, weil alle Sätze der Datei umnummeriert werden sollen, so muss `pos` angegeben werden.

Wie die Sätze nummeriert werden, hängt davon ab, wieviele Satznummern an dieser Stelle frei sind. Als Schrittweite der Nummerierung wird die größtmögliche der Schrittweiten `1/0`, `0/1`, `0/01`, `0/001` (letztere nur im Textmodus) gewählt. Sollen die Sätze mit einer anderen Schrittweite nummeriert werden, kann diese hinter `pos` angegeben werden. `pos` und Schrittweite müssen durch einen Strichpunkt getrennt werden (z. B. `23;0/2`). Als Schrittweite darf eine Zeilennummer und/oder eine Unterscheidungsnummer angegeben werden. Ist die Nummerierung in der angegebenen Schrittweite nicht möglich, weil nicht genügend Satznummern frei sind, so wird die angegebene Schrittweite ignoriert und eine kleinere gewählt.

Falls das Umnummerieren bzw. Umstellen nicht möglich ist, weil nicht genügend Satznummern frei sind, wird nachgefragt, ob nachfolgende Sätze umnummeriert werden dürfen, damit genügend Satznummern frei werden. Wird diese Frage bejaht, werden zusätzlich entsprechend viele nachfolgende Sätze umnummeriert, andernfalls unterbleibt das Umnummerieren bzw. Umstellen.

Enthält der umzunummerierende bzw. umzustellende Bereich mehr als 100 bzw. 10 Sätze, so wird, bevor die Sätze umnummeriert/umgestellt werden, zurückgefragt, ob soundsoviele Sätze umnummeriert/umgestellt werden sollen. Als Antwort kann `j` (ja) oder `n` (nein) oder eine neue Anweisung (auch in diesem Fall wird nicht umnummeriert/umgestellt) gegeben werden. Die Anzahl der Sätze, bis zu der diese Anfrage unterbleiben soll, kann, durch ein Nummernzeichen getrennt, hinter `ber` angegeben werden (z. B. `(23,-1)#30`).

## Organisatorische Anweisungen

Der Editor ist so ausgelegt, dass bei TUSTEP-Dateien normalerweise in der Originaldatei ediert werden kann. Soll in einer Kopie ediert werden, so kann die Originaldatei außerhalb des Editors in eine Arbeitsdatei kopiert werden, die ggf. nach dem Edieren zurückkopiert wird. Beide Kopiervorgänge können jedoch auch innerhalb des Editors mit Hilfe der Hole- und Rette-Anweisungen vorgenommen werden. Auf diese Weise kann auch der Inhalt von ASCII-Dateien ediert werden.

Daten, die mit der Hole-Anweisung in eine Arbeitsdatei geholt wurden, können mit dem Makro \*VEMO mit den Originaldaten verglichen werden. Wenn nichts anderes angegeben ist, werden die Unterschiede auf dem Bildschirm angezeigt. Weitere Informationen über das Makro werden mit dem Kommando #INFORMIERE, \*VEMO ausgegeben.

### Namen abfragen/einstellen/löschen

Für `datei` und `segment` können Namen eingestellt werden, die bei nachfolgenden Hole- und Rette-Anweisungen eingesetzt werden, falls sie dort weggelassen werden. Werden sie bei Hole- und Rette-Anweisungen angegeben, so werden jeweils diese Namen eingestellt und damit ggf. die mit der Name-Anweisung `n` eingestellten Namen ersetzt. Beim Wechseln der Editor-Datei (vgl. Datei-Anweisung `d`) werden die eingestellten Namen ebenfalls ersetzt.

Bei den folgenden Anweisungen können an Stelle eines Dateinamens für die Angabe `datei` auch ein, zwei oder drei Dollarzeichen angegeben werden. Dies ist gleichbedeutend mit der Angabe des Namens der Datei, die mit dem Kommando #DEFINIERE (siehe Seite 132) als erste, zweite bzw. dritte Makro-Datei definiert ist.

**n** Abfragen der eingestellten Namen für `datei` und `segment`.

**n, datei, -** Einstellen des Namens für `datei` und Löschen des eingestellten Namens für `segment`.

**n, datei, segment** Einstellen der Namen für `datei` und `segment`.

Wird `datei` weggelassen, also nur `segment` angegeben, so wird nur der neue Segmentname eingestellt; wird nur `datei` angegeben, so wird nur der neue Dateiname eingestellt.

**n! , , segment** Einstellen des Namens für `segment` und Umbenennen des Segments mit dem zuvor eingestellten Namen in der Datei mit dem aktuell eingestellten Namen.

**n!** Eingestellte Namen für `datei` und `segment` löschen.



## Holen

Die Datei, aus der mit einer der folgenden Anweisungen Daten geholt werden sollen, muss zum Lesen oder Schreiben angemeldet sein. Ist sie noch nicht angemeldet, kann als dritte Angabe der Hole-Anweisung nach einem Komma ein Träger angegeben werden, damit sie auf diesem Träger zum Lesen angemeldet wird. Für den Träger sind die gleichen Angaben möglich wie zur Spezifikation `TRAEGER` beim Kommando `#ANMELDE` (siehe Seite 124).

Bei den folgenden Anweisungen können an Stelle eines Dateinamens für die Angabe `datei` auch ein, zwei oder drei Dollarzeichen angegeben werden. Dies ist gleichbedeutend mit der Angabe des Namens der Datei, die mit dem Kommando `#DEFINIERE` (siehe Seite 132) als erste, zweite bzw. dritte Makro-Datei definiert ist.

### a) Holen aller Daten einer Datei

**h, datei, -** Kopieren der TUSTEP- bzw. ASCII-Datei `datei` in die Editor-Datei.

Wird `datei` weggelassen, so wird dafür der eingestellte Name (vgl. Name-Anweisung `n`) eingesetzt. Ist für `datei` ein Name und für `segment` kein Name eingestellt, so kann `, datei, -` weggelassen werden.

Falls in der Editor-Datei Daten stehen, die aus einer anderen Datei geholt und seit der letzten Änderung noch nicht gerettet wurden, wird eine entsprechende Fehlermeldung angezeigt und es wird nachgefragt, ob die Daten trotzdem geholt werden sollen. Um diese Frage zu vermeiden, kann die folgende Anweisung verwendet werden:

**h!, datei, -** Kopieren der TUSTEP- bzw. ASCII-Datei `datei` in die Editor-Datei. Dabei werden die Daten in der Editor-Datei in überschrieben.

Wird `datei` weggelassen, so wird dafür der eingestellte Name (vgl. Name-Anweisung `n`) eingesetzt. Ist für `datei` ein Name und für `segment` kein Name eingestellt, so kann `, datei, -` weggelassen werden.

### b) Holen eines Segments aus einer Segment-Datei

**h, datei, segment** Kopieren des Segments `segment` aus der Segment-Datei `datei` in die Editor-Datei.

Wird `datei` oder `, datei, segment` weggelassen, so wird dafür jeweils der eingestellte Name (vgl. Name-Anweisung `n`) eingesetzt.

Für `segment` kann auch die Seitennummer des Segments angegeben werden, unter der das Segment in der Segment-Datei gespeichert ist.

Falls in der Editor-Datei Daten stehen, die aus einer anderen Datei geholt und seit der letzten Änderung noch nicht gerettet wurden, wird eine entsprechende Fehlermeldung angezeigt und es wird nachgefragt, ob die Daten trotzdem geholt werden sollen. Um diese Frage zu vermeiden, kann die folgende Anweisung verwendet werden:

**h!, datei, segment** Kopieren des Segments `segment` aus der Segment-Datei `datei` in die Editor-Datei. Dabei werden die Daten in der Editor-Datei überschrieben.

Wird `datei` oder `,datei,segment` weggelassen, so wird dafür jeweils der eingestellte Name (vgl. Name-Anweisung `n`) eingesetzt.

Für `segment` kann auch die Seitennummer des Segments angegeben werden, unter der das Segment in der Segment-Datei gespeichert ist.

### c) Holen des Inhaltsverzeichnisses aus einer Segment-Datei

**h, datei, ?** Kopieren des Inhaltsverzeichnisses aus der Segment-Datei `datei` in die Editor-Datei.

Wird `datei` weggelassen, so wird dafür der eingestellte Dateiname (vgl. Name-Anweisung `n`) eingesetzt.

Falls in der Editor-Datei Daten stehen, die aus einer anderen Datei geholt und seit der letzten Änderung noch nicht gerettet wurden, wird eine entsprechende Fehlermeldung angezeigt und es wird nachgefragt, ob die Daten trotzdem geholt werden sollen. Um diese Frage zu vermeiden, kann die folgende Anweisung verwendet werden:

**h!, datei, ?** Kopieren des Inhaltsverzeichnisses aus der Segment-Datei `datei` in die Editor-Datei. Dabei werden die Daten in der Editor-Datei überschrieben.

Wird `datei` weggelassen, so wird dafür der eingestellte Dateiname (vgl. Name-Anweisung `n`) eingesetzt.

## Retten

Sollen die Daten der Editor-Datei mit einer der folgenden Anweisungen gerettet werden, so muss die Editor-Datei zum Schreiben angemeldet sein (obwohl die Daten dabei nur gelesen werden), damit danach in der Editor-Datei (automatisch) eingetragen werden kann, wohin die Daten gerettet wurden.

Die Datei, in die mit einer der folgenden Anweisungen Daten gerettet werden sollen, muss zum Schreiben angemeldet sein. Ist sie nicht zum Schreiben angemeldet, kann als dritte Angabe der Rette-Anweisung nach einem Komma ein Träger angegeben werden, damit sie auf diesem Träger zum Schreiben angemeldet wird; nach dem Retten der Daten wird die Datei wieder zum Lesen angemeldet. Für den Träger sind die gleichen Angaben möglich wie zur Spezifikation `TRAEGER` beim Kommando `#ANMELDE` (siehe Seite 124).

Bei den folgenden Anweisungen können an Stelle eines Dateinamens für die Angabe `datei` auch ein, zwei oder drei Dollarzeichen angegeben werden. Dies ist gleichbedeutend mit der Angabe des Namens der Datei, die mit dem Kommando `#DEFINIERE` (siehe Seite 132) als erste, zweite bzw. dritte Makro-Datei definiert ist.

### a) Retten der Daten in eine Datei

**r, datei, -** Kopieren der Editor-Datei in die TUSTEP- bzw. ASCII-Datei `datei`.

Wird `datei` weggelassen, so wird dafür der eingestellte Name (vgl. Name-Anweisung `n`) eingesetzt. Ist für `datei` ein Name und für Segment kein Name eingestellt, so kann `,datei,-` weggelassen werden.

Falls in der Datei `datei` schon Daten stehen, wird eine entsprechende Fehlermeldung angezeigt und es wird nachgefragt, ob die Daten trotzdem gerettet werden sollen. Um diese Frage zu vermeiden, kann die folgende Anweisung verwendet werden:

**r!**, `datei`, - Kopieren der Editor-Datei in die TUSTEP- bzw. ASCII-Datei `datei`.

Wird `datei` weggelassen, so wird dafür der eingestellte Name (vgl. Name-Anweisung `n`) eingesetzt. Ist für `datei` ein Name und für Segment kein Name eingestellt, so kann `, datei, -` weggelassen werden.

Falls in der Datei `datei` schon Daten stehen, werden diese überschrieben.

#### b) Retten der Daten in ein Segment einer Segment-Datei

Sollen Daten als Segment in eine Segment-Datei gerettet werden, müssen sie im Programmmodus nummeriert sein.

**r**, `datei`, `segment` Kopieren der Editor-Datei in die Segment-Datei `datei` unter dem Segmentnamen `segment`.

Wird `datei` oder `, datei, segment` weggelassen, so wird dafür jeweils der eingestellte Name (vgl. Name-Anweisung `n`) eingesetzt.

Falls in der Segment-Datei schon ein Segment mit dem angegebenen Namen vorhanden ist, wird eine entsprechende Fehlermeldung angezeigt und es wird nachgefragt, ob die Daten trotzdem gerettet werden sollen. Um diese Frage zu vermeiden, kann die folgende Anweisung verwendet werden:

**r!**, `datei`, `segment` Kopieren der Editor-Datei in die Segment-Datei `datei` unter dem Segmentnamen `segment`.

Wird `datei` oder `, datei, segment` weggelassen, so wird dafür jeweils der eingestellte Name (vgl. Name-Anweisung `n`) eingesetzt.

Falls in der Segment-Datei schon ein Segment mit dem angegebenen Namen vorhanden ist, wird dieses Segment ersetzt.

Falls die Editor-Datei leer ist, wird nachgefragt, ob das Segment in der Segment-Datei ganz gelöscht werden soll. Es sind dann folgende Antworten möglich:

**j** Das Segment wird gelöscht und der Segmentname aus dem Inhaltsverzeichnis entfernt.

**n** Das Segment wird gelöscht (!), der Segmentname bleibt aber im Inhaltsverzeichnis erhalten.

neue Anweisung: Das Segment bleibt in der Segment-Datei unverändert erhalten.

## Löschen eines Segments einer Segment-Datei

Soll ein Segment in einer Segment-Datei mit dem Editor gelöscht werden, so muss die Editor-Datei leer sein. Dann kann die leere Editor-Datei mit der oben beschriebenen Rette-Anweisung `r!`, `datei`, `segment` in dieses Segment »gerettet« werden. Der Editor fragt in diesem Fall nach, ob das Segment in der Segment-Datei ganz gelöscht werden soll. Darauf ist wie oben beschrieben zu antworten: mit `n`, wenn nur der Inhalt des Segments, mit `j`, wenn auch der Eintrag im Inhaltsverzeichnis gelöscht werden soll.

## Editor-Datei abfragen/wechseln

**d** Abfragen des Namens der Datei, die gerade mit dem Editor bearbeitet wird.

**d, datei** Wechseln der Datei, die mit dem Editor bearbeitet werden soll.

**d, datei, traeger** Anmelden und Wechseln der Datei, die mit dem Editor bearbeitet werden soll.

Vor dem Wechseln in die angegebene Datei wird diese auf dem angegebenen Träger zum Lesen angemeldet. Für `traeger` sind die gleichen Angaben möglich wie zur Spezifikation `TRAEGER` beim Kommando `#ANMELDE` (siehe Seite 124).

**d, -std-** Standard-Editor-Datei bearbeiten.

**d, ?** Anzeigen einer Liste mit den zuletzt mit dem Editor bearbeiteten Dateien.

Soll eine dieser Dateien mit dem Editor bearbeitet werden, so kann sie mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Return-Taste bestätigt werden, oder es kann die entsprechende Zeile mit der linken Maustaste angeklickt werden.

Soll die entsprechende Anweisung in die Anweisungszeile ausgegeben werden, damit sie ggf. in veränderter Form ausgeführt werden kann, muss die Auswahl mit dem Enter-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der rechten Maustaste angeklickt werden.

Die Anzeige kann durch Drücken der Leertaste oder Escape-Taste wieder gelöscht werden.

Um die Liste zu reduzieren, kann mit der Delete-Taste der jeweils markierte Dateiname aus der Liste entfernt werden. Wird die Anzeige anschließend mit der Escape-Taste gelöscht, so werden alle mit der Delete-Taste entfernten Dateinamen wieder in die Liste aufgenommen.

## Modus abfragen/einstellen

### a) Nummerierung und Zeichendarstellung

Eine vollständige Modusangabe besteht aus drei Teilwerten:

Der erste Teilwert kann »+« oder »-« sein und gibt an, ob die Sätze (z. B. nach einer Zeige-Anweisung) mit (+) oder ohne (-) Satznummer im Textfeld angezeigt werden sollen. Eine Ausnahme bilden die »Such-Anweisungen für strukturierte Daten«; hier werden die Daten unabhängig von dieser Einstellung immer ohne Satznummer im Textfeld angezeigt. Zu beachten ist, dass Sätze ohne Satznummern nicht korrigiert werden können.

Der zweite Teilwert kann »T« oder »P« sein und gibt an, ob die Sätze im Textmodus (mit Seitennummer) oder im Programmmodus (ohne Seitennummer) nummeriert bzw. angezeigt werden sollen.

Der dritte Teilwert kann »+« oder »-« sein und gibt an, ob Akzentbuchstaben und mit dem Steuerzeichen »#« codierte Zeichen im Textfeld als solche dargestellt (+) oder in der Eingabe-Codierung (-) dargestellt werden sollen, ob also z. B. ein e mit Gravis als è oder als %\e dargestellt werden soll. Diese Einstellung ist jedoch nur von Bedeutung, wenn mit dem Kommando #DEFINIERE (siehe Seite 132) auch ein Code eingestellt wurde, in dem solche Zeichen unterstützt werden.

- m** Abfragen des eingestellten Modus.
- m, . . .** Modus . . . einstellen. Für » . . . « kann eine dreistellige Modusangabe angegeben werden, wie sie oben beschrieben ist, oder nur die erste Stelle oder nur die ersten beiden Stellen einer solchen Modusangabe.
- m, t** Modus T einstellen: Nummerierung der Sätze im Textmodus.
- m, p** Modus P einstellen: Nummerierung der Sätze im Programmmodus.  
 Falls das Umstellen von Modus T auf Modus P nicht möglich ist, weil die Satznummern zu groß sind, wird nachgefragt, ob die Sätze entsprechend unnummeriert werden sollen. Wird diese Frage bejaht, wird unnummeriert und Modus P eingestellt, andernfalls bleibt Modus T eingestellt.

### b) Textfeld

Der Editor verwendet zum Anzeigen der Daten ein Textfeld, wenn nicht mit einer der beiden folgenden Anweisungen zwei Textfelder verlangt werden und damit das obere/untere Fenster bzw. linke/rechte aktiviert wird. In den beiden Fenstern kann die gleiche oder jeweils eine andere Datei (durch Wechseln der Datei mit der Datei-Anweisung d) ediert werden.

- m, 1** Zwei Textfelder verwenden, oberes bzw. linkes Textfeld aktivieren.
- m, 2** Zwei Textfelder verwenden, unteres bzw. rechtes Textfeld aktivieren.
- m, 0** Nur ein Textfeld verwenden.

Mit den beiden folgenden Anweisungen kann bestimmt werden, in welcher Weise das Textfeld geteilt wird:

- m, h**      Textfeld horizontal (in oberes und unteres) teilen.  
**m, v**      Textfeld vertikal (in linkes und rechtes) teilen.

### Dateititel abfragen/einstellen/löschen

- tt**            Dateititel abfragen.
- tt=titeltext**    Definieren des Dateititels. Für `titeltext` kann eine beliebige Zeichenfolge mit bis zu 120 Zeichen angegeben werden.
- Werden die Daten mit der `RETTE`-Anweisung als Segment in eine Segment-Datei gerettet, so wird der Dateititel der Editor-Datei als Segment-Titel mitgespeichert; wird ein Segment aus einer Segment-Datei geholt, so wird der Segment-Titel automatisch als Dateititel in die Editor-Datei übernommen.
- tt=**            Dateititel löschen.

### Tabulator abfragen/einstellen/löschen

Nach dem Initialisieren einer TUSTEP-Sitzung sind die Tabulatorpositionen 11, 21, 31, 41, 51, 61, 71 voreingestellt.

- tab**            Tabulatorzeichen und Tabulatorpositionen abfragen.
- tab, x, n1 n2 n3 . . .**    Definieren des Zeichens `x` (für `x` kann ein beliebiges Zeichen mit Ausnahme der Ziffern gewählt werden) als Tabulatorzeichen und Definieren der Positionen `n1 n2 n3` usw. als Tabulatorpositionen.
- Ein so definiertes Tabulatorzeichen wirkt nur bei der Eingabe von Daten nach den Anweisungen `ee` und `e, pos#n`. Dabei wird jedes Tabulatorzeichen beim Abspeichern der Daten durch so viele Leerzeichen ersetzt, dass die nächsthöhere Tabulatorposition erreicht wird (entspricht der Funktionsweise der Tabulatortaste einer Schreibmaschine). Falls es keine höhere Tabulatorposition gibt, wird das Tabulatorzeichen durch ein Leerzeichen ersetzt.
- Die so definierten Tabulatorpositionen können auch durch den Steuerbefehl `TAB` (vgl. Seite 350) angesprungen werden. Sollen sie nur dafür verwendet werden, kann in der Anweisung das Zeichen `x` (nicht aber die Kommata) weggelassen werden.

- tab!**            Tabulatorzeichen und Tabulatorpositionen löschen.

Tabulatorzeichen und Tabulatorpositionen können auch beim Aufruf des Editors definiert bzw. gelöscht werden (siehe Kommando `#EDIERE` Seite 149). Sollen Tabulatorzeichen und/oder Tabulatorpositionen automatisch beim ersten Aufruf des Editors in einer TUSTEP-Sitzung definiert werden, können sie in das Segment `EDIT` der `INI`-Datei (siehe Seite 89) eingetragen werden.

## Funktionen aufrufen/definieren/löschen/abfragen

Im Editor ist eine Funktion ein Kürzel der Form  $F_n$  ( $n =$  eine Zahl von 1 bis 60) für eine beliebige Editoranweisung oder einen Makroaufruf. Ein solches Kürzel wird Funktion genannt, weil die Anweisung, für die das Kürzel steht, in der Regel durch Drücken der entsprechenden Funktionstaste (siehe Seite 341) aufgerufen und ausgeführt wird.

Von TUSTEP sind folgende Funktionen vordefiniert:

$F1=ZA$	$F5=ZU, *$	$F9=G-$
$F2=ZE$	$F6=D, ?$	$F10=G+$
$F3=Z$	$F7=ZB, *$	$F11=M, 1$
$F4=B$	$F8=ZA, *$	$F12=M, 2$

Sie können jedoch undefiniert werden.

**f** Anzeigen einer Liste mit allen Funktionen.

Soll eine der angezeigten Funktionen unverändert ausgeführt werden, so muss sie mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Return-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der linken Maustaste angeklickt werden; soll die Definition einer Funktion vollständig ins Editorfenster ausgegeben werden, damit sie ggf. verändert werden kann, muss die Auswahl mit der Enter-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der rechten Maustaste angeklickt werden.

Die Anzeige kann durch Drücken der Leertaste oder Escape-Taste wieder gelöscht werden.

Wird eine markierte Funktion mit der Delete-Taste aus der Liste entfernt, so wird auch die Definition dieser Funktion gelöscht. Wird die Anzeige anschließend mit der Escape-Taste gelöscht, so wird das Löschen der mit der Delete-Taste entfernten Funktionen wieder aufgehoben.

**f** $n$ =anweisung Definieren der Funktion  $n$  als Editoranweisung.

Für *anweisung* muss eine Editoranweisung stehen. Diese kann dann durch Drücken der Funktionstaste  $n$  oder durch Eingabe von **f** $n$  als Editoranweisung aufgerufen und ausgeführt werden. Die Definition einer Funktion gilt jeweils so lange, bis sie durch eine neue Definition geändert oder gelöscht wird.

Wird eine so definierte Funktion aufgerufen, so wird zunächst der Steuerbefehl CONFIRM (siehe Seite 398) ausgeführt. Dadurch werden ggf. im Textfeld vorgenommenen Änderungen bestätigt und in die Datei übertragen; eine ggf. in der Anweisungszeile stehende Anweisung wird ausgeführt. Erst danach wird die für die Funktion definierte Editoranweisung ausgeführt.

**f** $n$ = $y, name$  Definieren der Funktion  $n$  als Makroaufruf.

Für *name* muss der Name eines Editormakros stehen. Dieses kann dann

durch Drücken der Funktionstaste *n* ausgeführt werden. Die Definition einer Funktion gilt jeweils so lange, bis sie durch eine neue Definition geändert oder gelöscht wird.

**f<sub>n</sub>=** Löschen der Definition der Funktion *n*.

**f!** Löschen der Definitionen aller Funktionen.

Mit dieser Anweisung werden alle Funktionen gelöscht, auch diejenigen, die von TUSTEP voreingestellt sind. Die von TUSTEP voreingestellten Funktionen sind oben angegeben und müssen bei Bedarf wieder definiert werden.

**f=-std-** Vordefinierte Funktionen wieder einstellen.

Funktionen können auch beim Aufruf des Editors definiert bzw. gelöscht werden (siehe Kommando #EDIERE Seite 149). Sollen Funktionen automatisch beim ersten Aufruf des Editors in einer TUSTEP-Sitzung definiert werden, können sie in das Segment EDIT der INI-Datei (siehe Seite 89) eingetragen werden.

## Makros aufrufen/definieren/löschen/abfragen

Im Editor ist ein Makro eine Folge von Steuerbefehlen und/oder Zeichenfolgen, die mit einer Funktionstaste (siehe Seite 341), einer Tastenkombination (siehe Seite 342) oder der Maus (siehe Seite 344) aufgerufen werden kann. Die Steuerbefehle sind ab Seite 350 beschrieben.

Editormakros können auch durch Eingabe ihres Namens aufgerufen und ausgeführt werden. Eine solche Eingabe wird durch einen doppelten Anschlag der Plus-Taste im Ziffernblock eingeleitet. Danach muss der Name des Makros eingegeben und mit der Return-Taste bestätigt werden. Für Makros, deren Namen nur aus einem einzigen Buchstaben bestehen, ist auch ein verkürzter Aufruf möglich. Es genügt in diesem Fall ein einmaliger Anschlag der Plus-Taste und die Eingabe des Buchstabens (ohne zusätzliche Return-Taste).

Da auf manchen Notebooks die Plus-Taste nur umständlich zu erreichen oder nicht vorhanden ist, kann statt der Plus-Taste grundsätzlich auch die Tastenkombination `Ctrl+B` bzw. `Strg+B` verwendet werden.

Beim Aufruf des Editors (Kommando #EDIERE) kann zur Spezifikation MAKRO der Name eines Makros angegeben werden, das nach dem Start des Editors sofort ausgeführt wird.



Von TUSTEP sind folgende Makros vordefiniert:

Y, A=ADD_STRT_TAG	Y, Q=SELECT_CHAR	Y, M_LC=CLICK	Y, N_0="" 0"
Y, B=DEFINE_BM	Y, R=SHW_STRT_TAG	Y, M_LP=MRK_INI	Y, N_1="" 1"
Y, C=RECALL_CB	Y, S=NEXT_VBM	Y, M_LR=MRK_ASK	Y, N_2="" 2"
Y, D=SELECT_VBM	Y, T=SHOW_TAGS	Y, M_MC=MRK_TAG_DEL	Y, N_3="" 3"
Y, E=ADD_END_TAG	Y, U=NEXT_BM	Y, M_RC=MRK_INS	Y, N_4="" 4"
Y, F=PREV_VBM	Y, V=SHW_END_TAG	Y, M_RP=MRK_INI	Y, N_5="" 5"
Y, G=SELECT_TAG	Y, W=WHICH_SEGM	Y, M_RR=MRK_DEL_INS	Y, N_6="" 6"
Y, H=FETCH_SEGM	Y, X=SELECT_TEXT	Y, S_LC=SHW_CUR	Y, N_7="" 7"
Y, I=SELECT_BM	Y, Y=SELECT_ABBR	Y, S_LP=MRK_INI	Y, N_8="" 8"
Y, J=JOIN	Y, Z=RECALL_CHAR	Y, S_LR=MRK_CB	Y, N_9="" 9"
Y, K=RECALL_MOD		Y, S_RC=INSERT_CB	
Y, L=RECALL_DEL		Y, S_RP=MRK_INI	
Y, M=RECALL_MRK		Y, S_RR=MRK_INS_CB	
Y, N=INS_LINE_IND		Y, M_DN=SCR_DN	
Y, O=PREV_BM		Y, M_UP=SCR_UP	
Y, P=REPL_ABBR		Y, S_DN=SCR_DN_BOTH	
		Y, S_UP=SCR_UP_BOTH	

Sie können jedoch undefiniert werden.

**y** Anzeigen einer Liste mit allen Makros.

Soll eines der angezeigten Makros unverändert ausgeführt werden, so muss es mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Return-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der linken Maustaste angeklickt werden; soll die Definition eines Makros vollständig ins Editorfenster ausgegeben werden, damit sie ggf. verändert werden kann, muss die Auswahl mit der Enter-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der rechten Maustaste angeklickt werden.

Die Anzeige kann durch Drücken der Leertaste oder Escape-Taste wieder gelöscht werden.

Wird ein markiertes Makro mit der Delete-Taste aus der Liste entfernt, so wird auch die Definition dieses Makros gelöscht. Wird die Anzeige anschließend mit der Escape-Taste gelöscht, so wird das Löschen der mit der Delete-Taste entfernten Makros wieder aufgehoben.

**y, name=makroanweisungen** Definieren des Makros name.

Der Name eines Makros kann aus 1 bis 12 Zeichen (Buchstaben, Ziffern und »\_«) bestehen, muss mit einem Buchstaben beginnen und darf nicht mit »\_« enden. Ausnahmen davon sind in der Beschreibung explizit aufgeführt.

Eine Makroanweisung ist entweder ein Steuerbefehl oder eine Zeichenfolge. Die möglichen Steuerbefehle und ihre Wirkung sind ab Seite 350 beschrieben. Eine Zeichenfolge muss mit einem frei wählbaren Begrenzungszeichen (Sonderzeichen außer Komma und Leerzeichen) eingelei-

tet und abgeschlossen werden. Sie wird ab der jeweils aktuellen Cursor-Position so ausgegeben, als wäre sie dort mit der Tastatur eingegeben worden. Die einzelnen Makroanweisungen müssen durch Komma getrennt sein. Vor jeder Makroanweisung kann, durch das Zeichen »\*« getrennt, eine Zahl (von 1 bis 99) angegeben werden, die angibt, wie oft die Makroanweisung ausgeführt werden soll.

**y, name=** Löschen des Makros *name*.

**y!** Löschen aller Makros und Makroleisten.

Mit dieser Anweisung werden alle Editormakros gelöscht, auch diejenigen, die von TUSTEP voreingestellt und für die Verwendung der Maus erforderlich sind. Die von TUSTEP voreingestellten Makros sind oben angegeben und müssen bei Bedarf wieder definiert werden.

**y=-std-** Alle definierten Makros löschen und die vordefinierten Makros wieder einstellen.

**y, name.=makroanweisungen** Definieren des Makros *name*. .

Makros, deren Name mit einem Abkürzungspunkt endet, können nur mit dem Steuerbefehl `REPL_ABBR` (siehe Seite 377) oder über die Steuerbefehl `SELECT_ABBR` (siehe Seite 377) aufgerufen werden.

Diese Makros werden in der Regel verwendet, um Abkürzungen aufzulösen (`REPL_ABBR`) oder kurze Textteile an der Cursor-Position einzufügen (`SELECT_ABBR`).

Beim Aufruf des Editors und beim Wechseln der Editor-Datei (vgl. Datei-Anweisung d) kann automatisch ein Makro aufgerufen werden und damit z. B. eine bestimmte Colorierung der Daten eingestellt werden. Dazu können der Name, der Titel und die ersten beiden Sätze der Datei nach Zeichenfolgen durchsucht werden. Die Zeichenfolgen müssen in einer Such-Tabelle *stab* vorgegeben werden:

**y, fn\_#=stab** Definiert die Zeichenfolgen, nach denen im Dateinamen (einschließlich des Projektnamens, z. B. *daten\*testdatei*) der Datei gesucht werden soll.

**y, fn\_#=** Löscht die Definition.

**y, ft\_#=stab** Definiert die Zeichenfolgen, nach denen im Titel der Datei gesucht werden soll.

**y, ft\_#=** Löscht die Definition.

**y, fc\_#=stab** Definiert die Zeichenfolgen, nach denen im ersten Satz der Datei gesucht werden soll.

**y, fc\_#=** Löscht die Definition.

**y, fx\_#=stab** Definiert die Zeichenfolgen, nach denen im ersten und im zweiten Satz der Datei gesucht werden soll.

**y, fx\_#=** Löscht die Definition.

Wenn beim Aufruf des Editors (Kommando `#EDIERE`) zur Spezifikation `MAKRO` der

Name eines Makros angegeben wird, unterbleibt beim Aufruf eine weitere Prüfung. Andernfalls werden wie beim Wechseln der Editor-Datei folgende drei Prüfungen in der unten angegebenen Reihenfolge ausgeführt. Sobald jedoch ein entsprechendes Makro definiert ist und aufgerufen werden kann, entfallen weitere Prüfungen.

- Wenn  $f_n_\#$  definiert ist: Suchen einer entsprechenden Zeichenfolge im Dateinamen (genauer: in der Dateibezeichnung der Form »Projektname\*Dateiname«). Falls die  $n$ -te Zeichenfolge der Such-Tabelle als erste gefunden wird, erfolgt der Aufruf des Makros  $f_n_\#$ , wobei an Stelle des Nummernzeichens die Zahl  $n$  eingesetzt wird; falls keine entsprechende Zeichenfolge gefunden wird, erfolgt der Aufruf des Makros  $f_{n_0}$  ( $\#$  = Ziffer 0).
- Wenn  $f_t_\#$  definiert ist: Suchen einer entsprechenden Zeichenfolge im Dateititel. Falls die  $n$ -te Zeichenfolge der Such-Tabelle als erste gefunden wird, erfolgt der Aufruf des Makros  $f_t_\#$ , wobei an Stelle des Nummernzeichens die Zahl  $n$  eingesetzt wird; falls keine entsprechende Zeichenfolge gefunden wird, erfolgt der Aufruf des Makros  $f_{t_0}$  ( $\#$  = Ziffer 0).
- Wenn  $f_c_\#$  definiert ist: Suchen einer entsprechenden Zeichenfolge im ersten Satz der Datei. Falls die  $n$ -te Zeichenfolge der Such-Tabelle als erste gefunden wird, erfolgt der Aufruf des Makros  $f_c_\#$ , wobei an Stelle des Nummernzeichens die Zahl  $n$  eingesetzt wird; falls keine entsprechende Zeichenfolge gefunden wird, erfolgt der Aufruf des Makros  $f_{c_0}$  ( $\#$  = Ziffer 0).
- Wenn  $f_x_\#$  definiert ist: Suchen einer entsprechenden Zeichenfolge im ersten und im zweiten Satz der Datei. Falls die  $n$ -te Zeichenfolge der Such-Tabelle als erste gefunden wird, erfolgt der Aufruf des Makros  $f_x_\#$ , wobei an Stelle des Nummernzeichens die Zahl  $n$  eingesetzt wird; falls keine entsprechende Zeichenfolge gefunden wird, erfolgt der Aufruf des Makros  $f_{x_0}$  ( $\#$  = Ziffer 0).

Hinweis: Sollen diese Prüfungen auch erfolgen, wenn beim Aufruf des Editors der Name eines Makros angegeben ist, so können sie in diesem Makro einzeln mit den Steuerbefehlen `CHECK_FN`, `CHECK_FT` und/oder `CHECK_FC` (siehe Seite 381) aufgerufen werden.

Makros können auch beim Aufruf des Editors definiert bzw. gelöscht werden (siehe Kommando `#EDIERE` Seite 149). Sollen Makros automatisch beim ersten Aufruf des Editors in einer TUSTEP-Sitzung definiert werden, können sie in das Segment `EDIT` der INI-Datei (siehe Seite 89) eingetragen werden.

## Makroleisten definieren/löschen/abfragen

**y** Ausgeben einer Übersicht der Makros und Makroleisten.

Eine Makroleiste ist eine Zeile im Editorfenster, die aus einzelnen Feldern besteht. Jedes dieser Felder hat einen Namen. Wird ein Feld mit einer Maustaste angeklickt, so wird ein Makro aufgerufen. In welcher Weise der Name dieses Makros zusammengesetzt wird, ist vom Typ der Makroleiste abhängig.

Die Eigenschaften eines Feldes können mit einer eigenen Anweisung oder innerhalb der Definition einer Makroleiste (s. u.) definiert werden. Namen einer Felddefinition müssen vor dem ersten Buchstaben mit einem `"@"` gekennzeichnet werden. Ein Bei-

spiel, bei dem die Eigenschaften eines Feldes nicht innerhalb der Definition einer Makroleiste definiert werden können, ist am Ende dieses Kapitels (auf Seite 296) angegeben.

**y, @def=fld** Definieren eines Feldes für eine Makroleiste.

**y, @def=** Löschen einer Felddefinition.

Für ein Feld `fld` können bis zu drei durch Apostroph getrennte Namen angegeben werden. Der erste, zweite bzw. dritte wird verwendet, wenn das Feld mit der linken, mittleren bzw. rechten Maustaste angeklickt wird. Falls für eine Maustaste keine Aktion vorgesehen ist, kann an Stelle eines Namens ein Minuszeichen angegeben werden. Wenn mehrere Namen angegeben sind, wird der erste davon in der Makroleiste angezeigt. Soll eine andere Zeichenfolge angezeigt werden, so kann diese nach dem letzten Namen des jeweiligen Feldes durch einen Doppelpunkt getrennt angegeben werden. Diese Zeichenfolge muss außerdem mit einem frei wählbaren Begrenzungszeichen (Sonderzeichen außer Komma und Leerzeichen) eingeleitet und abgeschlossen werden.

Die Felder werden in der Farbe angezeigt, die für die jeweilige Zeile (Menü-Zeile, Meldungszeile oder Statuszeile), in der sie angezeigt werden, eingestellt ist. Soll ein Feld in einer anderen Farbe angezeigt werden, so kann als letzte Angabe des jeweiligen Feldes nach einem Doppelpunkt der entsprechende Hexadezimal-Code angegeben werden. Die möglichen Hexadezimal-Codes können der Tabelle entnommen werden, die durch die Tastenkombination `Ctrl+F` bzw. `Strg+F` angezeigt wird.

Die Felder einer Makroleiste werden alle gleich breit angezeigt. Wieviele Felder maximal angezeigt werden können, hängt von der Zeilenlänge (= Breite des Editorfensters) und dem längsten Feldnamen (bzw. der ersatzweise anzuzeigenden Zeichenfolge) der Makroleiste ab; die Anzahl ergibt sich durch die Formel  $(\text{Zeilenlänge}+1)/(\text{längster Feldname}+1)$ .

#### a) Permanente Makroleisten

Permanente Makroleisten werden in der Menü-Zeile des Editor-Fenster angezeigt. Sie werden so lange angezeigt, bis sie explizit gelöscht oder durch eine andere Makroleiste ersetzt werden. Ersetzt werden können sie durch Definieren einer anderen permanenten Makroleiste oder mit dem Steuerbefehl `CHG_MACROS` (siehe Seite 383).

**y, name~=fld1, fld2, ...** Definieren einer Makroleiste.

Diese Anweisung definiert die permanente Makroleiste `name` mit den Feldern `fld1`, `fld2`, ... und zeigt sie in der Menü-Zeile an.

Der Name der Makroleiste darf maximal 11 Zeichen lang sein.

**y, name~=** Löschen einer Makroleiste.

Diese Anweisung löscht die Definition der permanenten Makroleiste `name`. Wird gerade eine permanente Makroleiste angezeigt, so wird diese Anzeige ebenfalls gelöscht.

Für jedes Feld kann entweder ein mit "@" gekennzeichneteter Name einer Felddefinition (s. o.) angegeben werden oder es kann direkt ein Felddefinition angegeben

werden. Wird der Name einer Felddefinition angegeben, so muss nach dem Namen durch Doppelpunkt getrennt noch ein Hilfstext angegeben werden. Dieser muss mit einem frei wählbaren Begrenzungszeichen (Sonderzeichen außer Komma und Leerzeichen) eingeleitet und abgeschlossen werden. Mit dem Hilfstext wird die Mindestbreite bestimmt, in der das Feld angezeigt wird.

Wird ein Feld einer permanenten Makroleiste mit einer Maustaste angeklickt, wird ein Makro aufgerufen. Der Name dieses Makros setzt sich aus zwei Teilen zusammen. Der erste Namensteil entspricht bei permanenten Makroleisten dem Namen der Makroleiste, der zweite entspricht dem für die verwendete Maustaste angegebenen Namen des angeklickten Feldes. Ergibt sich aus den beiden Namensteilen ein Name mit mehr als zwölf Zeichen, werden nur die ersten zwölf Zeichen berücksichtigt; dieser gekürzte Name darf nicht (zufällig) mit einem Unterstrichsstrich enden.

Neben den permanenten Makroleisten mit Namen gibt es noch eine ohne Namen. Sie kann in gleicher Weise mit den oben angegebenen Anweisungen definiert und gelöscht werden. Dabei entfällt der Name vor der Tilde. Ist diese namenlose Makroleiste definiert, so wird sie nach dem Start des Editors automatisch angezeigt; ist sie beim Starten des Editors nicht definiert, wird die Menü-Zeile nicht angezeigt.

#### b) Optionale Makroleisten

Optionale Makroleisten werden in der Meldungszeile angezeigt. Sie werden so lange angezeigt, bis sie explizit gelöscht oder durch eine andere Makroleiste ersetzt werden. Ersetzt werden können sie durch Definieren einer anderen optionalen Makroleiste oder mit dem Steuerbefehl `CHG_MACROS` (siehe Seite 383).

Wenn jedoch eine Meldung angezeigt werden muss, so wird die Makroleiste vorübergehend verdeckt. Sie wird wieder sichtbar, wenn die Meldung vom Editor gelöscht wird, die Meldung mit der linken Maustaste angeklickt wird oder der Steuerbefehl `SHOW_MACROS` (siehe Seite 383) ausgeführt wird.

**y, name\*=f1d1, f1d2, ...** Definieren einer Makroleiste.

Diese Anweisung definiert die optionale Makroleiste `name` mit den Feldern `f1d1`, `f1d2`, ... und zeigt sie in der Meldungszeile an.

Der Name der Makroleiste darf maximal 11 Zeichen lang sein.

**y, name\*=** Löschen einer Makroleiste.

Diese Anweisung löscht die Definition der optionalen Makroleiste `name`. Wird gerade eine optionale Makroleiste angezeigt, so wird diese Anzeige ebenfalls gelöscht.

Für jedes Feld kann entweder ein mit "@" gekennzeichnete Name einer Felddefinition (s. o.) angegeben werden oder es kann direkt ein Felddefinition angegeben werden. Wird der Name einer Felddefinition angegeben, so muss nach dem Namen durch Doppelpunkt getrennt noch ein Hilfstext angegeben werden. Dieser muss mit einem frei wählbaren Begrenzungszeichen (Sonderzeichen außer Komma und Leerzeichen) eingeleitet und abgeschlossen werden. Mit dem Hilfstext wird die Mindestbreite bestimmt, in der das Feld angezeigt wird.

Wird ein Feld einer optionalen Makroleiste mit einer Maustaste angeklickt, wird ein

Makro aufgerufen. Der Name dieses Makros setzt sich aus zwei Teilen zusammen. Der erste Namensteil entspricht bei optionalen Makroleisten dem Namen der Makroleiste, der zweite entspricht dem für die verwendete Maustaste angegebenen Namen des angeklickten Feldes. Ergibt sich aus den beiden Namensteilen ein Name mit mehr als zwölf Zeichen, werden nur die ersten zwölf Zeichen berücksichtigt; dieser gekürzte Name darf nicht (zufällig) mit einem Unterstrichungsstrich enden.

Neben den optionalen Makroleisten mit Namen gibt es noch eine ohne Namen. Sie kann in gleicher Weise mit den oben angegebenen Anweisungen definiert und gelöscht werden. Dabei entfällt der Name vor dem Stern. Ist diese namenlose Makroleiste definiert, so wird sie nach dem Start des Editors automatisch angezeigt.

### c) Temporäre Makroleisten

Temporäre Makroleisten können innerhalb eines Makros mit dem Steuerbefehl `SWITCH:erg?name` (siehe Seite 375) aufgerufen werden. Dadurch werden der Meldungstext in der Anweisungszeile und die Makroleiste in der Statuszeile angezeigt. Diese Anzeige bleibt nur bis zur nächsten Eingabe mit der Maus oder der Tastatur lesbar; danach wird automatisch wieder der ursprüngliche Inhalt der Anweisungszeile und der Statuszeile angezeigt.

**y**, ?name=text, fld1, fld2, ... Definieren einer Makroleiste.

Diese Anweisung definiert die temporäre Makroleiste `name` mit der Meldung `text` und den Feldern `fld1`, `fld2`, ...

Der Name der Makroleiste darf maximal 11 Zeichen lang sein.

Der Meldungstext muss mit einem frei wählbaren Begrenzungszeichen (Sonderzeichen außer Komma und Leerzeichen) eingeleitet und abgeschlossen werden.

**y**, ?name= Löschen einer Makroleiste.

Diese Anweisung löscht die Definition der temporären Makroleiste `name`.

Der Text wird in der für die Anweisungszeile eingestellten Farbe angezeigt. Soll der Text in einer anderen Farbe angezeigt werden, so kann hinter dem den Text abschließenden Begrenzungszeichen nach einem Doppelpunkt der entsprechende Hexadezimal-Code angegeben werden. Die möglichen Hexadezimal-Codes können der Tabelle entnommen werden, die durch die Tastenkombination `Ctrl+F` bzw. `Strg+F` angezeigt wird.

Für jedes Feld kann entweder ein mit "@" gekennzeichnete Name einer Felddefinition (s. o.) angegeben werden oder es kann direkt ein Felddefinition angegeben werden. Wird der Name einer Felddefinition angegeben, so muss nach dem Namen durch Doppelpunkt getrennt noch ein Hilfstext angegeben werden. Dieser muss mit einem frei wählbaren Begrenzungszeichen (Sonderzeichen außer Komma und Leerzeichen) eingeleitet und abgeschlossen werden. Mit dem Hilfstext wird die Mindestbreite bestimmt, in der das Feld angezeigt wird.

Wird ein Feld einer temporären Makroleiste mit einer Maustaste angeklickt, wird ein Makro aufgerufen. Der Name dieses Makros setzt sich aus zwei Teilen zusammen. Der erste Namensteil entspricht bei temporären Makroleisten der im Steuerbefehl

`SWITCH:erg?name` angegebenen Namensergänzung, der zweite entspricht dem für die verwendete Maustaste angegebenen Namen des angeklickten Feldes. Ergibt sich aus den beiden Namensteilen ein Name mit mehr als zwölf Zeichen, werden nur die ersten zwölf Zeichen berücksichtigt; dieser gekürzte Name darf nicht (zufällig) mit einem Unterstrichsstrich enden.

Wird eine temporäre Makroleiste angezeigt und kein Feld angeklickt, sondern mit der Tastatur ein Zeichen oder eine Tastenkombination für einen Steuerbefehl eingegeben oder eine Funktionstaste gedrückt, so wird ebenfalls ein Makro aufgerufen. Der Name dieses Makros setzt sich aus zwei Teilen zusammen. Der erste Namensteil entspricht der im Steuerbefehl `SWITCH:erg?name` angegebenen Namensergänzung, der zweite entspricht dem Zeichen bzw. dem Namen des Steuerbefehls bzw. dem Namen der Funktionstaste. Eine Ausnahme bilden Leerzeichen, Fragezeichen und Tilde; wird eines dieser Zeichen eingegeben, so wird `BLANK`, `HELP` bzw. `TILDE` als zweiter Namensteil eingesetzt. Ergibt sich aus den beiden Namensteilen ein Name mit mehr als zwölf Zeichen, werden nur die ersten zwölf Zeichen berücksichtigt.

Ergeben die beiden Namensteile den Namen eines definierten Makros, werden die auf den Steuerbefehl `SWITCH` folgenden Makroanweisungen nicht mehr ausgeführt, sondern es wird das Makro mit dem entsprechenden Namen aufgerufen und ausgeführt.

Wurde ein Zeichen eingegeben und ist kein entsprechendes Makro definiert, wird als zweiter Namensteil `CHAR` eingesetzt. Ist auch mit diesem Namen kein Makro definiert, wird als zweiter Namensteil `KEY` eingesetzt.

Wurde eine Funktionstaste gedrückt oder eine Tastenkombination für einen Steuerbefehl eingegeben und ist kein entsprechendes Makro definiert, wird als zweiter Namensteil `KEY` eingesetzt.

Ist kein Makro mit passendem Namen definiert, wird ein Signalton (`BEEP`) ausgegeben; dann werden die auf den Steuerbefehl `SWITCH` folgenden Makroanweisungen ausgeführt. Von diesen Makroanweisungen aus kann mit dem Steuerbefehl `RETRY` (siehe Seite 376) zum Steuerbefehl `SWITCH` zurückgesprungen werden (um dann vielleicht eine vorgesehene Eingabe, für die ein Makro definiert ist, zu erhalten).

Hinweis: Solange eine temporäre Makroleiste angezeigt wird, werden Mauseaktionen und Tastatureingaben nicht in der üblichen Weise interpretiert. Es wird jeweils auf einen Mausklick auf ein Feld der temporären Makroleiste oder genau eine Tastatureingabe gewartet, um dann einen Namen zu erzeugen und nach Möglichkeit das Makro mit diesem Namen aufzurufen.

#### d) Imaginäre Makroleisten

Imaginäre Makroleisten unterscheiden sich von temporären nur dadurch, dass sie keine Felder haben. Wird eine imaginäre Makroleiste mit dem Steuerbefehl `SWITCH` aufgerufen, wird der Meldungstext in der Anweisungszeile angezeigt, die Statuszeile bleibt unverändert. Damit entfällt die Möglichkeit, mit der Maus ein Feld anzuklicken.

**y**, ?name=text Definieren einer Makroleiste.

Diese Anweisung definiert die imaginäre Makroleiste `name` mit der Meldung `text`; sie hat keine Felder.

Der Meldungstext muss mit einem frei wählbaren Begrenzungszeichen (Sonderzeichen außer Komma und Leerzeichen) eingeleitet und abgeschlossen werden.

Beispiel mit getrennt definierten Feldern für Makroleisten

```
y,*=@licht:"LICHT ???",@tür:"TÜR ???"
```

```
y,@licht=licht_aus:"Licht AUS"
```

```
y,@tür=tür_zu:"Tür ZU"
```

```
y,licht_aus=CLR_CMD_LINE, /Y,@licht=licht_an:"Licht AN"/, ENTER
```

```
y,licht_an=CLR_CMD_LINE, /Y,@licht=licht_aus:"Licht AUS"/, ENTER
```

```
y,tür_zu =CLR_CMD_LINE, /Y,@tür=tür_auf:"Tür AUF"/, ENTER
```

```
y,tür_auf=CLR_CMD_LINE, /Y,@tür=tür_zu : "Tür ZU"/, ENTER
```

Dieses (nicht sehr praxisnahe) Beispiel zeigt eine Makroleiste mit zwei Feldern an, wobei die Beschriftung des ersten Feldes bei jedem Anklicken zwischen »Licht AUS« und »Licht AN« wechselt; das zweite Feld wechselt zwischen »TÜR ZU« und »TÜR AUF«. Erst durch Einfügen weiterer Steuerbefehle würde eine darüber hinausgehende Wirkung erreicht.

## Zeichen- und Stringgruppen definieren/löschen/abfragen

Eine Zeichen- bzw. eine Stringgruppe ist eine Zusammenfassung von einzelnen Zeichen bzw. von Zeichenfolgen (Strings), auf die in nachfolgenden Anweisungen innerhalb von Vergleichs-Tabellen `vtab` (siehe Seite 331), Such-Tabellen `stab` (siehe Seite 331) und Austausch-Tabellen `atab` (siehe Seite 332) durch Angabe der dazugehörigen Gruppenkennung Bezug genommen werden kann.

Die Definition einer Gruppe gilt jeweils für alle nachfolgenden Anweisungen, bis diese Gruppe neu definiert oder gelöscht wird.

In der Definition einer Zeichengruppe für den Editor darf keine Kennung einer selbst definierten Gruppe vorkommen.

**z:xy=zeichen** Definieren der Zeichengruppe `xy`.

Die möglichen Angaben zu `zeichen` entsprechen denen einer Zeichengruppe (Parameterart V). Diese ist für die Parameter-Konvention `»{ }«` ab Seite 709 und für die Parameter-Konvention `»<>«` ab Seite 741 beschrieben. Nach welcher Konvention die Parameter interpretiert werden, kann mit dem Kommando `#PARAMETER` (siehe Seite 207) eingestellt werden.

**c:xy=zeichen** Alternative Schreibweise für **z:xy=zeichen**

**z:xy=** Löschen der Zeichengruppe `xy`.

**c:xy=** Alternative Schreibweise für **z:xy=**

**s:xy=strings** Definieren der Stringgruppe `xy`.



Die möglichen Angaben zu `strings` entsprechen denen einer Stringgruppe (Parameterart V). Diese ist für die Parameter-Konvention `»{ }«` ab Seite 709 und für die Parameter-Konvention `»<>«` ab Seite 741 beschrieben. Nach welcher Konvention die Parameter interpretiert werden, kann mit dem Kommando `#PARAMETER` (siehe Seite 207) eingestellt werden.

**s** : `xy`= Löschen der Stringgruppe `xy`.

**i** Anzeigen einer Liste mit allen Zeichen- und Stringgruppen.

Soll die Definition einer angezeigten Zeichen- oder Stringgruppe vollständig ins Editorfenster ausgegeben werden, damit sie ggf. verändert werden kann, so muss sie mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Enter-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der rechten Maustaste angeklickt werden.

Die Anzeige kann durch Drücken der Leertaste oder Escape-Taste wieder gelöscht werden.

Wird eine markierte Zeichen- oder Stringgruppe mit der Delete-Taste aus der Liste entfernt, so wird auch die Definition dieser Zeichen- bzw. Stringgruppen gelöscht. Wird die Anzeige anschließend mit der Escape-Taste gelöscht, so wird das Löschen der mit der Delete-Taste entfernten Zeichen- und Stringgruppen wieder aufgehoben.

Zeichen- und Stringgruppen können auch beim Aufruf des Editors definiert bzw. gelöscht werden (siehe Kommando `#EDIERE` Seite 149). Sollen Zeichen- und/oder Stringgruppen automatisch beim ersten Aufruf des Editors in einer TUSTEP-Sitzung definiert werden, können sie in das Segment `EDIT` der INI-Datei (siehe Seite 89) eingetragen werden.

Falls mit dem Kommando `#PARAMETER,MODUS=...` (siehe Seite 207) ein anderer Modus als `»{ }«` eingestellt wurde, gilt folgendes zusätzlich:

Zur Definition von Zeichen- und Stringgruppen und zur Bezugnahme auf die so definierten Gruppen stehen für Zeichengruppen Gruppenkennungen der Form `>[xy]` und für Stringgruppen Gruppenkennungen der Form `<[xy]` zur Verfügung. Dabei ist `xy` ein aus zwei Zeichen bestehender Name, wobei `x` ein Buchstabe und `y` eine Buchstabe oder eine Ziffer sein muss. Groß- und Kleinschreibung wird nicht unterschieden.

Daneben sind noch Gruppenkennungen für Zeichen- und Stringgruppen der Form `>n` und `<n` möglich, wobei `n` jeweils durch eine Ziffer zu ersetzen ist. Es können auf diese Weise also bis zu 20 Zeichengruppen und bis zu 20 Stringgruppen zusätzlich definiert werden.

`>[xy]=zeichen` Definieren der Zeichengruppe `xy`.

Eine Beschreibung der zu `zeichen` möglichen Angaben befindet sich unter »Parameterart V« (Seite 741 ff.).

`>[xy]=` Löschen der Zeichengruppe `xy`.

`<[xy]=strings` Definieren der Stringgruppe `xy`.

Eine Beschreibung der zu `strings` möglichen Angaben befindet sich unter »Parameterart V« (Seite 741 ff.).

`<[xy]=` Löschen der Stringgruppe `xn`.

`xnz=zeichen` Definieren der Zeichengruppe `xn`.

Für `x` muss `>><` oder `><<`, für `n` muss die entsprechende Ziffer angegeben werden (z. B. `>2z=aeiou` zur Definition der Zeichengruppe `>2`, die die Vokale `a, e, i, o, u` enthalten soll).

Eine Beschreibung der zu `zeichen` möglichen Angaben befindet sich unter »Parameterart V« (Seite 741 ff.).

`xnz=` Löschen der Zeichengruppe `xn`.

Für `x` muss `>><` oder `><<`, für `n` muss die entsprechende Ziffer angegeben werden.

`xns=strings` Definieren der Stringgruppe `xn`.

Für `x` muss `>><` oder `><<` und für `n` muss die entsprechende Ziffer angegeben werden (z. B. `>2s='%/'%\'%<<'%:'%;'` zur Definition der Stringgruppe `>2`, die die in französischen Texten vorkommenden Akzente enthalten soll).

Eine Beschreibung der zu `strings` möglichen Angaben befindet sich unter »Parameterart V« (Seite 741 ff.).

`xns=` Löschen der Stringgruppe `xn`.

Für `x` muss `>><` oder `><<`, für `n` muss die entsprechende Ziffer angegeben werden.

## Colorierung definieren/wechseln/löschen/abfragen

Wie die Farbe eingestellt werden kann, in der Daten im Textfeld angezeigt werden, ist im Kapitel »Einstellen der Farben« auf Seite 333 beschrieben. Die im Folgenden beschriebenen Anweisungen erlauben es darüber hinaus, bestimmte Zeichenfolgen (z. B. Steuerzeichen) in einer vorgegebenen Farbe anzuzeigen und gegen Änderungen zu schützen.

Um verschiedenartigen Daten (z. B. Texten oder Programmen) gerecht zu werden, können bis zu neun Farbgruppen definiert werden. Sie werden mit `c1` bis `c9` bezeichnet. In jeder Farbgruppe können ein Kommentar und bis zu neun Farben mit den dazugehörigen Zeichenfolgen definiert werden. Diese neun Farben werden von 1 bis 9 nummeriert.

Bei der Colorierung der angezeigten Daten werden zuerst die Zeichenfolgen der Farbe 1 in der entsprechenden Farbe angezeigt, dann diejenigen der Farbe 2, 3 usw. Falls sich Zeichenfolgen von verschiedenen Farben überlappen, hat also die Farbe mit der höheren Nummer Vorrang.

`cn, =kommentar` Definiert für die Farbgruppe `n` einen Kommentar.

Die Definition des Kommentars gilt so lange, bis sie durch eine neue Definition geändert oder gelöscht wird.

- c<sub>n</sub>, =**        Löscht in der Farbgruppe *n* den Kommentar.
- c<sub>n</sub>, m=xx:stab** Definiert für die Farbgruppe *n* die *m*-te Farbe. Sie wird durch *xx* festgelegt; für *xx* sind die gleichen Angaben vorgesehen wie für den Steuerbefehl `MRK_CHG:xx` (siehe Seite 368). An Stelle von *stab* muss eine Such-Tabelle *stab* (siehe Seite 331) angegeben werden.
- c<sub>n</sub>, m=xx, %:stab** Definiert für die Farbgruppe *n* die *m*-te Farbe wie die zuvor beschriebene Anweisung, jedoch werden beim Suchen der Zeichenfolgen, die coloriert werden sollen, bestimmte Zeichenfolgen übergangen (siehe »Ignorieren von Akzent-Codierungen« 330)
- c<sub>n</sub>, m=xx, %%:stab** Definiert für die Farbgruppe *n* die *m*-te Farbe wie die zuvor beschriebene Anweisung, jedoch werden beim Suchen der Zeichenfolgen, die coloriert werden sollen, bestimmte Zeichenfolgen übergangen (siehe »Ignorieren von Auszeichnungen und Schriftumschaltungen« Seite 330)
- c<sub>n</sub>, m=xx, \_:stab** Definiert für die Farbgruppe *n* die *m*-te Farbe wie oben beschrieben, jedoch werden nur ganze Wörter coloriert (siehe »Nur ganze Wörter suchen/austauschen« Seite 330)
- c<sub>n</sub>, m=xx, \_\_:stab** Definiert für die Farbgruppe *n* die *m*-te Farbe wie oben beschrieben, jedoch werden nur ganze Wörter coloriert (siehe »Nur ganze Wörter suchen/austauschen« Seite 330)
- c<sub>n</sub>, m=xx, #:stab** Definiert für die Farbgruppe *n* die *m*-te Farbe wie oben beschrieben, jedoch werden nur Satznummern coloriert, die eine angegebene Zeichenfolge enthalten.

Wird bei den vorgenannten Anweisungen unmittelbar hinter der Farbangabe *xx* ein Stern `»*«` eingefügt, so werden alle Zeichenfolgen (z. B. Tags), die in dieser Farbe angezeigt werden (nicht nur die mit *stab* angegebenen Zeichenfolgen!), gegen Änderungen im Textfeld geschützt. Diese Zeichenfolgen können jedoch (z. B. mit dem Steuerbefehl `MRK_TAG_DEL`, siehe Seite 367) komplett gelöscht werden. Die Zeichenkombination `><` bildet eine Ausnahme: Zwischen die beiden Klammern dürfen Daten eingefügt werden, auch wenn beide Klammern geschützt sind. Damit ist es möglich, zwischen zwei unmittelbar hintereinander stehenden geschützten Tags Daten einzufügen.

Die Definition einer Farbe gilt so lange, bis sie durch eine neue Definition geändert oder gelöscht wird.

- c<sub>n</sub>, m=**        Löscht in der Farbgruppe *n* die *m*-te Farbe.
- c<sub>n</sub>=**            Löscht alle Farbdefinitionen der Farbgruppe *n*.
- c!**             Löscht alle Farbdefinitionen in allen Farbgruppen.
- c<sub>n1</sub>=c<sub>n2</sub>**    Löscht alle Farbdefinitionen der Farbgruppe *n1* und kopiert dann alle Farbdefinitionen der Farbgruppe *n2* in die Farbgruppe *n1*.

Nachdem Farben definiert sind, wird beim Anzeigen von Daten im Textfeld jeweils die aktuelle Farbgruppe verwendet. Als aktuelle Farbgruppe gilt diejenige,

- in der zuletzt eine Farbe definiert oder gelöscht wurde,
- die zuletzt durch die Anweisung `cn` ( $n = 1$  bis  $9$ ) angezeigt wurde,
- die zuletzt in der mit der Anweisung `c` angezeigten Liste ausgewählt wurde,
- die zuletzt mit dem Steuerbefehl `COLOR` (siehe Seite 399) eingestellt wurde.

`c0` Schaltet die Colorierung aus. Mit dem Steuerbefehl `COLOR` und durch jede andere mit `c` beginnende Anweisung kann sie wieder eingeschaltet werden.

Mit den beiden nachfolgend beschriebenen Anweisungen kann jeweils eine Liste mit Farbdefinitionen angezeigt werden. Soll eine der angezeigten Farbgruppen eingestellt werden, so muss eine Zeile davon mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Return-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der linken Maustaste angeklickt werden; soll eine Farbdefinition vollständig ins Editorfenster ausgegeben werden, damit sie ggf. verändert werden kann, so muss die entsprechende Zeile mit Pfeil nach unten/oben ausgewählt werden und die Auswahl mit der Enter-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der rechten Maustaste angeklickt werden.

`c` Zeigt eine Liste aller Farbdefinitionen an, beginnend mit der aktuellen Farbgruppe. Mit dieser Anweisung kann also auch festgestellt werden, welches die aktuelle Farbgruppe ist.

`cn` Zeigt eine Liste der Farbdefinitionen der Farbgruppe  $n$  an. Für  $n$  ist die entsprechende Ziffer einzusetzen.

Die Anzeige kann durch Drücken der Leertaste oder Escape-Taste wieder gelöscht werden.

Wird eine mit Pfeil nach unten/oben ausgewählte Farbdefinition mit der Delete-Taste aus der Liste entfernt, so wird auch die Definition dieser Farbdefinition gelöscht. Wird die Anzeige anschließend mit der Escape-Taste gelöscht, so wird das Löschen der mit der Delete-Taste entfernten Farbdefinitionen wieder aufgehoben.

Colorierungen können auch beim Aufruf des Editors definiert bzw. gelöscht werden (siehe Kommando `#EDIERE` Seite 149). Sollen Colorierungen automatisch beim ersten Aufruf des Editors in einer TUSTEP-Sitzung definiert werden, können sie in das Segment `EDIT` der INI-Datei (siehe Seite 89) eingetragen werden.

## Tag-Prüfung definieren/wechseln/löschen/abfragen

Mit den Anweisungen `tp`, `tpv` und `tpR` (siehe »Suchen und Prüfen von Tags« Seite 317) wird die korrekte Schachtelung und die Syntax der Tags überprüft. Soll darüber hinaus auch geprüft werden, ob nur vorgesehene Tag-Namen, Attribut-Namen und Attribut-Werte verwendet wurden, und ob ein Tag bzw. Text an der jeweiligen Stelle in der Tag-Hierarchie erlaubt ist, müssen zuvor alle erlaubten Tags mit den im Folgenden beschriebenen Anweisungen definiert werden.

Wenn eine Datei mit korrekten Tags (z. B. eine valide oder wohlgeformte XML-Datei) zur Verfügung steht, können die entsprechenden Tag-Definitionen auch mit dem

Makro `*TADE` erstellt und in eine Datei ausgegeben werden. Diese Definitionen müssen i.d.R. noch von Hand nachgearbeitet und präzisiert werden. Weitere Informationen über das Makro werden mit dem Kommando `#INFORMIERE, *TADE` ausgegeben.

Um verschiedenartigen Daten gerecht zu werden, können bis zu neun Tag-Gruppen definiert werden. Sie werden mit `T1` bis `T9` bezeichnet. In jeder Tag-Gruppe können ein Kommentar und beliebig viele Tags definiert werden.

`tn,=kommentar` Definiert für die Tag-Gruppe `n` einen Kommentar.

Die Definition des Kommentars gilt so lange, bis sie durch eine neue Definition geändert oder gelöscht wird.

`tn,=` Löscht in der Tag-Gruppe `n` den Kommentar.

`tn,name=attr;pfad;komm` Definiert für die Tag-Gruppe `n` das Tag mit dem Namen `name`.

Wenn unmittelbar hinter dem Namen des Tags ein `»/«` angegeben wird, darf das Tag in den Daten nur als leeres Tag vorkommen; wenn ein `»&«` angegeben wird, darf das Tag als leeres Tag und als Anfangs- und Ende-Tag vorkommen; wenn keines der beiden Zeichen angegeben ist, darf das Tag nur als Anfangs- und Ende-Tag vorkommen.

Soll überprüft werden, ob in den Daten an der jeweiligen Stelle in der Tag-Hierarchie Text erlaubt ist, müssen an Stelle eines Tag-Namens `name` drei Punkte, für `attr` ein Minuszeichen und für `pfad` die entsprechenden Pfade (s. u.) angegeben werden.

Mit `attr` kann angegeben werden, ob und ggf. welche Attribute obligat bzw. optional sind: Ein Minuszeichen bedeutet »keine Attribute«, ein Pluszeichen bedeutet »beliebige Attribute«, durch Kommata getrennte Namen bedeuten, dass diese Attribute obligat sind. Optionale Attribute müssen mit einem `»?«` unmittelbar vor dem Namen gekennzeichnet werden. Sind für ein Attribut nur bestimmte Attribut-Werte erlaubt, können diese jeweils durch Apostroph getrennt hinter dem Attribut-Namen angegeben werden. Attribut-Name und Attribut-Werte müssen durch ein Gleichheitszeichen getrennt sein. Wird als Attribut-Wert ein Zahlenpaar `n-m` angegeben, bedeutet dies, dass in den Daten als Attribut-Wert die Zahlen von `n` bis `m` erlaubt sind.

Mit `pfad` kann angegeben werden, an welchen Stellen in der Tag-Hierarchie das Tag erlaubt ist. Ein Pfad kann sich aus folgenden drei Angaben zusammensetzen: `<tag>` für ein Tag mit dem Namen `tag`, `<*>` für ein Tag mit einem beliebigen Namen, `*` für null oder beliebig viele Tags mit jeweils beliebigem Namen; falls ein Tag nur auf der obersten Hierarchiestufe erlaubt ist, muss `<>` als Pfad angegeben werden. Unmittelbar hinter den Angaben `<tag>` und `<*>` kann jeweils noch eine Zahl `n` in eckigen Klammern angegeben werden. Sie beschränkt das davor stehende Tag auf der aktuellen Hierarchiestufe auf das `n`-te Tag mit dem betreffenden Namen. Die einzelnen Pfade müssen durch ein frei wählbares Begrenzungszeichen voneinander getrennt sein. Vor der ersten und nach der letzten

Pfadangabe muss ebenfalls das Begrenzungszeichen angegeben werden. Es können erlaubte Pfade und unerlaubte Pfade (Ausnahmen) angegeben werden. Durch Angabe von zwei aufeinander folgenden Begrenzungszeichen kann beliebig oft zwischen erlaubten und unerlaubten Pfaden gewechselt werden. Beim Prüfen, ob ein Tag an einer bestimmten Stelle in den Daten erlaubt ist, werden die Pfade in der angegebenen Reihenfolge überprüft, bis eine Übereinstimmung gefunden wird. Falls keine Übereinstimmung mit einem erlaubten Pfad gefunden wurde, ist das Tag an dieser Stelle in den Daten nicht erlaubt.

Für `komm` kann ein Kommentar angegeben werden. Wird kein Kommentar angegeben, entfällt auch der Strichpunkt davor. Werden auch keine Pfade angegeben, entfallen beide Strichpunkte.

Die Definition eines Tags gilt so lange, bis sie durch eine neue Definition geändert oder gelöscht wird.

`tn, name=` Löscht in der Tag-Gruppe `n` das Tag `name`.

`tn=` Löscht alle Tag-Definitionen der Tag-Gruppe `n`.

`tn1=tn2` Löscht alle Tag-Definitionen der Tag-Gruppe `n1` und kopiert dann alle Tag-Definitionen der Tag-Gruppe `n2` in die Tag-Gruppe `n1`.

Nachdem Tags definiert sind, wird beim Prüfen von Tags mit den Anweisungen `tp`, `tpV` und `tpr` jeweils die aktuelle Tag-Gruppe verwendet. Als aktuelle Tag-Gruppe gilt diejenige,

- in der zuletzt ein Tag definiert oder gelöscht wurde,
- die zuletzt durch die Anweisung `tn` (`n = 1` bis `9`) angezeigt wurde,
- die zuletzt in der mit der Anweisung `t` angezeigten Liste ausgewählt wurde,
- die zuletzt mit dem Steuerbefehl `TAGS` (siehe Seite 399) eingestellt wurde.

Hinweis: Wenn alle Tags mit ihren Attributen definiert sind, kann mit dem Steuerbefehl `SELECT_TAG` (siehe Seite 385) eine Liste mit den an der jeweiligen Cursor-Position erlaubten Tags bzw. Attributen angezeigt werden.

`t0` Schaltet für die Tag-Prüfung die zusätzliche Verwendung der in den Tag-Gruppen definierten Regeln aus. Mit dem Steuerbefehl `TAGS` und jeder der Anweisungen `t1` bis `t9` kann die zusätzliche Verwendung einer Tag-Gruppe für die Tag-Prüfung wieder eingeschaltet werden.

Mit den beiden nachfolgend beschriebenen Anweisungen kann jeweils eine Liste mit Tag-Definitionen angezeigt werden. Soll eine der angezeigten Tag-Gruppen eingestellt werden, so muss eine Zeile davon mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Return-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der linken Maustaste angeklickt werden; soll eine Tag-Definition vollständig ins Editorfenster ausgegeben werden, damit sie ggf. verändert werden kann, so muss die entsprechende Zeile mit Pfeil nach unten/oben ausgewählt werden und die Auswahl mit der Enter-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der rechten Maustaste angeklickt werden.

- t** Zeigt eine Liste aller Tag-Definitionen an, beginnend mit der aktuellen Tag-Gruppe. Mit dieser Anweisung kann also auch festgestellt werden, welches die aktuelle Tag-Gruppe ist.
- tn** Zeigt eine Liste der Tag-Definitionen der Tag-Gruppe *n* an und stellt diese als aktuelle Tag-Gruppe ein. Für *n* ist die entsprechende Ziffer einzusetzen.
- Die Anzeige kann durch Drücken der Leertaste oder Escape-Taste wieder gelöscht werden.
- Wird eine mit Pfeil nach unten/oben ausgewählte Tag-Definition mit der Delete-Taste aus der Liste entfernt, so wird auch die Definition dieser Tag-Definition gelöscht. Wird die Anzeige anschließend mit der Escape-Taste gelöscht, so wird das Löschen der mit der Delete-Taste entfernten Tag-Definitionen wieder aufgehoben.

Tags können auch beim Aufruf des Editors definiert bzw. gelöscht werden (siehe Kommando #EDIERE Seite 149). Sollen Tags automatisch beim ersten Aufruf des Editors in einer TUSTEP-Sitzung definiert werden, können sie in das Segment EDIT der INI-Datei (siehe Seite 89) eingetragen werden.

## Kommandos ausführen

Der Editor kann nur Editoranweisungen und Editormakros ausführen. Sollen TUSTEP-Kommandos vom Editor aus aufgerufen werden, so kann dazu die folgende Editoranweisung verwendet werden. Mit ihr wird der Editor automatisch beendet und nach Ausführung der Kommandos wieder gerufen.

**x, komm** Ausführen des Kommandos *komm*.

Das Kommando *komm* muss durch ein Begrenzungszeichen, das frei wählbar ist, am Anfang und am Ende begrenzt werden. Sollen mehrere Kommandos ausgeführt werden, so sind diese durch dieses Begrenzungszeichen voneinander zu trennen. Besteht ein Kommando aus mehr als einer Zeile (z. B. bei Kommandos mit Parametern), so müssen die einzelnen Zeilen ebenfalls durch dieses Begrenzungszeichen voneinander getrennt werden. Als Begrenzungszeichen ist ein Zeichen zu wählen, das im Kommando nirgends vorkommt.

Die Kommandos müssen vollständig sein; d. h. bei Kommandos, auf die Daten (z. B. Parameter) folgen, müssen diese, einschließlich dem abschließenden \*EOF, mit angegeben werden.

Kommt in den Kommandos *komm* die Zeichenfolge <editor> vor, so wird diese Zeichenfolge (einschließlich der spitzen Klammern) durch den Namen der Editor-Datei ersetzt. Entsprechend wird die Zeichenfolge <datei> durch den definierten Dateinamen und die Zeichenfolge <segment> durch den definierten Segmentnamen ersetzt. Dateinamen und Segmentnamen können z. B. durch die Name-Anweisung *n* (siehe Seite 280) definiert werden.

Soll der Editor nach der Ausführung der Kommandos nicht wieder gerufen werden, kann die Beende-Anweisung `b, komm` (siehe Seite 273) verwendet werden.

## Ausgeben und Wiederholen von Anweisungen

Mit den nachfolgend beschriebenen Anweisungen können zuvor gegebene Anweisungen wieder ins Editorfenster ausgegeben werden. Sie können dann ggf. in veränderter Form oder auch unverändert wieder ausgeführt werden.

Es werden jeweils die letzten 40 Anweisungen gemerkt, die

- formal richtig sind,
- aus mehr als nur dem Anweisungskürzel bestehen und
- sich von der vorangehenden Anweisung unterscheiden.

Hinweis: Mit dem Steuerbefehl `MEM_OFF` (siehe Seite 400) kann das Merken der Anweisungen ausgeschaltet und mit dem Steuerbefehl `MEM_ON` wieder eingeschaltet werden.

- g** Ausgeben der letzten gemerkten Anweisung
- g-** Ausgeben der Anweisung, die vor der zuletzt mit `g`, `g-` oder `g+` ausgegebenen gemerkt wurde. Wurde seit der letzten Anweisung (außer `g`, `g-` und `g+`) noch keine der Anweisungen `g`, `g-` oder `g+` gegeben, so wird die zuletzt gemerkte Anweisung ausgegeben.
- g+** Ausgeben der Anweisung, die nach der zuletzt mit `g`, `g-` oder `g+` ausgegebenen gemerkt wurde. Wurde seit der letzten Anweisung (außer `g`, `g-` und `g+`) noch keine der Anweisungen `g`, `g-` oder `g+` gegeben, so wird die älteste gemerkte Anweisung ausgegeben.

Mit den Anweisungen `g+` bzw. `g-` kann also in den gemerkten Anweisungen »vor- bzw. zurückgeblättert« werden. Es empfiehlt sich, für diese Anweisungen jeweils eine Funktionstaste zu verwenden.

Eine durch `g-` oder `g+` ausgegebene Anweisung wird zur erneuten Ausführung bereitgestellt. Wenn sie jedoch nicht ausgeführt werden soll, sondern mit `g-` oder `g+` weitergeblättert werden soll, so muss jedesmal zuerst die eben ausgegebene Anweisung überschrieben bzw. gelöscht werden. Dies kann vermieden werden, indem die Anweisungen `g-` und `g+` durch Definieren von zwei Funktionen (siehe Seite 287) auf Funktionstasten gelegt werden. Dabei ist zu beachten, dass für jede der beiden Anweisungen jeweils nur eine einzige Funktion definiert sein darf. Die Funktionen `F9` und `F10` sind standardmäßig als `g-` und `g+` definiert. Soll eine andere Funktion mit `g-` bzw. `g+` definiert werden, so muss also erst `F9` bzw. `F10` undefiniert werden. Ist die Anweisung `g-` oder `g+` auf eine Funktionstaste gelegt, so wird beim Drücken der Funktionstaste eine im Editorfenster stehende Anweisung (im Gegensatz zu allen anderen Funktionstasten, wenn sie nicht als Makroaufruf definiert sind) nicht ausgeführt, sondern gleich die Anweisung, die auf diese Funktionstaste gelegt ist (also `g-` und `g+`). Dadurch erübrigt sich das Löschen der Anweisung, die zuvor mit `g`, `g-` oder `g+` ausgegeben wurde.



**g<sub>x</sub>** Ausgeben der letzten organisatorischen oder erweiterten Anweisung, die mit dem Buchstaben *x* beginnt. Für *x* ist der entsprechende Anfangsbuchstabe einzusetzen.

Soll eine Anweisung ausgeführt werden, die in ähnlicher Form als Funktion definiert ist, so kann diese mit folgender Anweisung ins Editorfenster ausgegeben werden, um sie dann zu ändern und ausführen zu lassen:

**g<sub>n</sub>** Ausgeben der Anweisung, die als Funktion *f<sub>n</sub>* definiert ist. Für *n* muss die Nummer der entsprechenden Funktion angegeben werden.

Soll die Definition einer Funktion, eines Makros, einer Zeichengruppe, einer Stringgruppe, eines Parameters, des Dateititels oder der leeren Tags geändert werden, die in ähnlicher Form schon definiert ist, so kann sie mit einer der nachfolgend genannten Anweisungen zur Änderung ins Editorfenster ausgegeben werden:

**g<sub>f<sub>n</sub></sub>** Ausgeben der Definition der Funktion *f<sub>n</sub>*. Für *n* ist die Nummer der entsprechenden Funktion einzusetzen (z. B. *gf2*).

**g<sub>y, name</sub>** Ausgeben der Definition des Makros *name*.

**g<sub>p<sub>n</sub>, xxx</sub>** Ausgeben der Definition des Parameters *p<sub>n</sub>, xxx*.

Für *n* ist die entsprechende Ziffer und für *xxx* die entsprechende Kennung einzusetzen (z. B. *gp2, zv3*).

**g, tt** Ausgeben der Definition des Dateititels.

**g, t<sub>pl</sub>** Ausgeben der Definition der leeren Tags.

**g, t<sub>sl</sub>** wie *g, t<sub>pl</sub>*.

**g, t<sub>pn</sub>** Ausgeben der Definition der Tag- und Attribut-Namen, die in jedem Fall legal sind, auch wenn sie den Bedingungen für diese Namen nicht entsprechen.

**g, t<sub>sn</sub>** wie *g, t<sub>pn</sub>*.

Mit den beiden nachfolgend beschriebenen Anweisungen kann jeweils eine Liste mit Anweisungen angezeigt werden. Soll eine der angezeigten Anweisungen unverändert ausgeführt werden, so muss sie mit Pfeil nach unten/oben ausgewählt und die Auswahl mit dem der Return-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der linken Maustaste angeklickt werden; soll eine Anweisung vollständig ins Editorfenster ausgegeben werden, damit sie ggf. in veränderter Form wieder ausgeführt werden kann, muss die Auswahl mit der Enter-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der rechten Maustaste angeklickt werden.

**gg** Zeigt die letzten gemerkten Anweisungen an.

**gg<sub>x</sub>** Zeigt die letzten gemerkten Anweisungen, die mit dem Buchstaben *x* beginnen, an. Für *x* ist der entsprechende Anfangsbuchstabe einzusetzen.

Die Anzeige kann durch Drücken der Leertaste oder Escape-Taste wieder gelöscht werden.

Um die Liste zu reduzieren, kann mit der Delete-Taste die jeweils markier-

te Anweisung aus der Liste entfernt werden. Wird die Anzeige anschließend mit der Escape-Taste gelöscht, so werden alle mit der Delete-Taste entfernten Anweisungen wieder in die Liste aufgenommen.

**gz**:xy Ausgeben der Definition der Zeichengruppe xy.

**gc**:xy Alternative Schreibweise für **gz**:xy.

**gs**:xy Ausgeben der Definition der Stringgruppe xy.

Falls mit dem Kommando #PARAMETER,MODUS=... (siehe Seite 207) ein anderer Modus als »{ }« eingestellt wurde, gilt folgendes zusätzlich:

**g>**[xy] Ausgeben der Definition der Zeichengruppe xy.

**g<**[xn] Ausgeben der Definition der Stringgruppe xy.

**gxnz** Ausgeben der Definition der Zeichengruppe xn.

Für x ist »>« oder »<« und für n ist die entsprechende Ziffer einzusetzen (z. B. g>2z).

**gxns** Ausgeben der Definition der Stringgruppe xn.

Für x ist »>« oder »<« und für n ist die entsprechende Ziffer einzusetzen (z. B. g>2s).

## Erweiterte Anweisungen

### Suchen und Zeigen

- za**, *ber*, *begr*, *stab* Suchen der Sätze eines Bereichs, die die Bedingungen *begr*, *stab* erfüllen, und jeweils Zeigen ab dem gefundenen Satz; Zeichenfolgen, die die Bedingungen im jeweils gefundenen Satz erfüllen, hervorgehoben anzeigen.
- zu**, *ber*, *begr*, *stab* Suchen der Sätze eines Bereichs, die die Bedingungen *begr*, *stab* erfüllen, und jeweils Zeigen des gefundenen Satzes mit Umgebung; Zeichenfolgen, die die Bedingungen im jeweils gefundenen Satz erfüllen, hervorgehoben anzeigen.
- zb**, *ber*, *begr*, *stab* Suchen der Sätze eines Bereichs, die die Bedingungen *begr*, *stab* erfüllen, und jeweils Zeigen bis zu dem gefundenen Satz; Zeichenfolgen, die die Bedingungen im jeweils gefundenen Satz erfüllen, hervorgehoben anzeigen.
- z**, *ber*, *begr*, *stab* Suchen der Sätze eines Bereichs, die die Bedingungen *begr*, *stab* erfüllen, und jeweils Zeigen entsprechend der Anweisung, die als letzte von den drei vorgenannten Anweisungen gegeben wurde. Wurde noch keine von diesen Anweisungen gegeben, so wirkt **z**, *ber*, *begr*, *stab* wie **zu**, *ber*, *begr*, *stab*.
- zn**, *ber*, *begr*, *stab* Suchen der Sätze eines Bereichs, die die Bedingungen *begr*, *stab* erfüllen, und Zeigen nur der gefundenen Sätze; Zeichenfolgen, die die Bedingungen erfüllen, hervorgehoben anzeigen.
- zf**, *ber*, *begr*, *stab* Zeigen aller Sätze eines Bereichs (fortlaufend); alle Zeichenfolgen, die die Bedingungen erfüllen, hervorgehoben anzeigen.

Bei jeder der sechs vorgenannten Zeige-Anweisungen kann durch Einfügen des Buchstabens *v* bzw. *r* nach dem ersten Buchstaben der Anweisung bestimmt werden, ob der angegebene Bereich vorwärts (d. h. zum Dateiende hin) oder rückwärts (d. h. zum Dateianfang hin) durchsucht werden soll. Nach einer so ergänzten Anweisung durchsucht jede der obigen fünf Anweisungen ohne Ergänzung den jeweils angegebenen Bereich ebenfalls vor- bzw. rückwärts. Wurde vor einer Zeige-Anweisung ohne Ergänzung noch keine mit einem *v* oder *r* ergänzte Anweisung gegeben, wird der Bereich vorwärts durchsucht.

Wird bei diesen Zeige-Anweisungen statt einer Bereichsangabe *ber* eine Positionsangabe *pos* verwendet, so wird von dieser Satzposition an vor- bzw. rückwärts gesucht.

Nach diesen Zeige-Anweisungen sind folgende Eingaben möglich:

leere Eingabe: In der gleichen Richtung weiter suchen und zeigen.

- w** Weiter suchen, aber gefundene Sätze nicht mehr zeigen, sondern nur am Ende die Anzahl der gefundenen Zeichenfolgen und die Anzahl der betroffenen Sätze anzeigen.

- zr** Rückwärts (d. h. zum Dateianfang hin) weiter suchen und zeigen.
- zv** Vorwärts (d. h. zum Dateiende hin) weiter suchen und zeigen.
- +** Suchen unterbrechen und vorwärts weiterblättern; danach bewirkt eine leere Eingabe jeweils ein Weiterblättern zum Dateiende hin (nicht möglich nach **zn**).
- Suchen unterbrechen und rückwärts weiterblättern; danach bewirkt eine leere Eingabe jeweils ein Weiterblättern zum Dateianfang hin (nicht möglich nach **zn**).

neue Anweisung: Das Suchen wird unterbrochen. In diesem Fall, wie auch bei den Eingaben **+** und **-**, kann später mit der Anweisung **z** (oder einer der Anweisungen **zr** und **zv**) das Suchen und Zeigen an der Unterbrechungsstelle fortgesetzt werden, falls der Editor inzwischen nicht verlassen oder mit der Anweisung **d** in eine andere Datei gewechselt wurde.

## Einfügen

**ei**, *text*, *ber:spa*, *begr*, *stab* Einfügen der Zeichenfolge *text* in jedem Satz des Bereichs *ber*, der die Bedingungen *begr*, *stab* erfüllt.

Die Zeichenfolge *text* muss durch ein Begrenzungszeichen, das frei wählbar ist, am Anfang und am Ende begrenzt werden. Dieses Begrenzungszeichen darf innerhalb der Zeichenfolge nicht vorkommen. Für die Codierung der Zeichenfolge *text* gelten die gleichen Regeln wie für Daten (siehe »Codierung der Zeichen« Seite 259).

Die Angabe *:spa* kann fehlen. Das bedeutet, dass die Zeichenfolge *text* jeweils am Ende eines Satzes eingefügt wird. Wird für *spa n* angegeben, so wird die Zeichenfolge jeweils ab der Position *n* eines Satzes eingefügt (d. h. die Zeichen von der Position *n* bis zum Ende des Satzes werden nach rechts geschoben). Die Angabe *+n* bewirkt, dass jeweils nach dem *n*-ten Zeichen vom Satzanfang an eingefügt wird. Entsprechend bewirkt die Angabe *-n*, dass jeweils vor dem *n*-ten Zeichen vom Satzende her eingefügt wird.

Wird für *spa n1-n2* angegeben, so wird die Zeichenfolge in den Spalten *n1* bis *n2* durch die Zeichenfolge *text* ersetzt. Die Zeichen nach der Spalte *n2* bis zum Satzende werden dabei nach links bzw. nach rechts verschoben, falls die Zeichenfolge *text* kürzer bzw. länger ist als die Zeichenfolge in den Spalten *n1* bis *n2*. Soll nur das Zeichen in Spalte *n* ersetzt werden, so ist dafür *n-n* (nicht *n*) anzugeben.

**ev**, *ber1*, *ber2*, *begr*, *stab* Einfügen der Sätze des Bereichs *ber1* vor jedem Satz des Bereichs *ber2*, der die Bedingungen *begr*, *stab* erfüllt.

**en**, *ber1*, *ber2*, *begr*, *stab* Einfügen der Sätze des Bereichs *ber1* nach jedem Satz des Bereichs *ber2*, der die Bedingung *begr*, *stab* erfüllt.

Bei der Ausführung dieser Anweisungen wird jeder Satz, der die Bedingungen erfüllt, mit Umgebung angezeigt; gleichzeitig wird gefragt, ob eingefügt werden soll. Es sind dann folgende Antworten möglich:

leere Eingabe: Es wird eingefügt und der nächste Satz gesucht, der die Bedingungen erfüllt.

- j** Es wird eingefügt und dann das Einfügen unterbrochen.
- n** Es wird nicht eingefügt, sondern der nächste Satz gesucht, der die Bedingungen erfüllt.
- w** Es wird eingefügt und im Folgenden ohne vorherige Anfrage eingefügt.

neue Anweisung: Das Einfügen wird unterbrochen, ohne dass in/vor/nach dem aktuellen Satz eingefügt wird. In diesem Fall, wie auch bei der Antwort **j**, kann später mit der Anweisung **e** das Einfügen an der Unterbrechungsstelle fortgesetzt werden, falls der Editor inzwischen nicht verlassen oder mit der Datei-Anweisung **d** in eine andere Datei gewechselt wurde.

## Löschen

**l!**, *ber, begr, stab* Löschen bestimmter Sätze eines Bereichs.

Es wird jeder Satz, der die Bedingungen *begr, stab* erfüllt, mit Umgebung angezeigt; gleichzeitig wird gefragt, ob er gelöscht werden soll. Es sind dann folgende Antworten möglich:

leere Eingabe: Der Satz wird gelöscht und der nächste gesucht.

- j** Der Satz wird gelöscht und dann das Löschen unterbrochen.
- n** Der Satz wird nicht gelöscht, sondern der nächste gesucht.
- w** Der Satz wird gelöscht, und alle folgenden Sätze, die die Bedingungen erfüllen, werden ohne Anfrage gelöscht.

neue Anweisung: Das Löschen wird unterbrochen, ohne dass der aktuelle Satz gelöscht wird. In diesem Fall, wie auch bei der Antwort **j**, kann später mit der Anweisung **l** das Löschen an der Unterbrechungsstelle fortgesetzt werden, falls der Editor inzwischen nicht verlassen oder mit der Datei-Anweisung **d** in eine andere Datei gewechselt wurde.

## Kopieren

**k**, *ber, pos, begr, stab* Kopieren bestimmter Sätze eines Bereichs.

**k**, *datei, ber, pos, begr, stab* Kopieren bestimmter Sätze aus einem Bereich einer anderen Datei.

**k**, *datei, segment, pos, begr, stab* Kopieren bestimmter Sätze aus einem Segment einer anderen Datei.

Es wird jeder Satz, der die Bedingungen *begr, stab* erfüllt, mit Umgebung angezeigt; gleichzeitig wird gefragt, ob er kopiert werden soll. Es sind dann folgende Antworten möglich:

leere Eingabe: Der Satz wird kopiert und der nächste gesucht.

- j** Der Satz wird kopiert und dann das Kopieren unterbrochen.

- n** Der Satz wird nicht kopiert, sondern der nächste gesucht.
  - w** Der Satz wird kopiert, und alle folgenden Sätze, die die Bedingungen erfüllen, werden ohne Anfrage kopiert.
- neue Anweisung: Das Kopieren wird beendet, ohne dass der aktuelle Satz kopiert wird. In diesem Fall, wie auch bei der Antwort **j**, kann später mit der Anweisung **k** das Kopieren an der Unterbrechungsstelle fortgesetzt werden, falls der Editor inzwischen nicht verlassen oder mit der Datei-Anweisung **d** in eine andere Datei gewechselt wurde.

Die Sätze werden so kopiert, dass sie beginnend mit der Satzposition  $pos$  (falls noch kein Satz mit der Satznummer  $pos$  vorhanden ist) bzw. nach der Satzposition  $pos$  (falls schon ein Satz mit der Satznummer  $pos$  vorhanden ist) zusätzlich in der Editor-Datei stehen. Falls sie ans Dateiende kopiert werden sollen, kann  $pos$  wegfallen. Im Textmodus erhält in diesem Fall der erste neue Satz die Nummer  $n.1$ , wobei  $n$  die um 1 erhöhte Seitennummer des letzten Satzes der Datei ist.

## Umstellen

**u, ber, pos, begr, stab** Umstellen bestimmter Sätze eines Bereichs.

Es wird jeder Satz, der die Bedingungen **begr, stab** erfüllt mit Umgebung angezeigt; gleichzeitig wird gefragt, ob er umgestellt werden soll. Es sind dann folgende Antworten möglich:

- leere Eingabe: Der Satz wird umgestellt und der nächste gesucht.
- j** Der Satz wird umgestellt und dann das Umstellen unterbrochen.
  - n** Der Satz wird nicht umgestellt, sondern der nächste gesucht.
  - w** Der Satz wird umgestellt, und alle folgenden Sätze, die die Bedingungen erfüllen, werden ohne Anfrage umgestellt.
- neue Anweisung: Das Umstellen wird beendet, ohne dass der aktuelle Satz umgestellt wird. In diesem Fall, wie auch bei der Antwort **j**, kann später mit der Anweisung **u** das Umstellen an der Unterbrechungsstelle fortgesetzt werden, falls der Editor inzwischen nicht verlassen oder mit der Datei-Anweisung **d** in eine andere Datei gewechselt wurde.

Die Sätze werden so umgestellt und unnummeriert, dass sie beginnend mit der Satzposition  $pos$  (falls noch kein Satz mit der Satznummer  $pos$  vorhanden ist) bzw. nach der Satzposition  $pos$  (falls schon ein Satz mit der Satznummer  $pos$  vorhanden ist) in der Datei stehen. Falls sie ans Dateiende umgestellt werden sollen, kann  $pos$  wegfallen. Im Textmodus erhält in diesem Fall der erste umgestellte Satz die Nummer  $n.1$ , wobei  $n$  die um 1 erhöhte Seitennummer des letzten Satzes der Datei ist.

## Austauschen

**a**, **ber**, **begr**, **atab** Austauschen der unter **atab** angegebenen Zeichenfolgen in einem Bereich.

Bevor eine Zeichenfolge ausgetauscht wird, wird der Satz in der ursprünglichen und in der durch den Austausch entstehenden neuen Form und die Umgebung angezeigt; gleichzeitig wird gefragt, ob ausgetauscht werden soll. Es sind dann folgende Antworten möglich:

leere Eingabe: Es wird ausgetauscht und die nächste Zeichenfolge gesucht, die ausgetauscht werden soll.

**j** Die Zeichenfolge wird ausgetauscht und dann das Austauschen unterbrochen.

**n** Es wird nicht ausgetauscht, sondern die nächste Zeichenfolge gesucht, die ausgetauscht werden soll.

**w** Die Zeichenfolge wird ausgetauscht und im Folgenden ohne vorherige Anfrage ausgetauscht.

**jf** Die Zeichenfolge wird ausgetauscht und im Rest des aktuellen Satzes keine mehr ausgetauscht, sondern vom nachfolgenden Satz an die nächste Zeichenfolge gesucht, die ausgetauscht werden soll.

**nf** Es wird nicht ausgetauscht, sondern vom nachfolgenden Satz an die nächste Zeichenfolge gesucht, die ausgetauscht werden soll.

**wf** Die Zeichenfolge wird ausgetauscht und im Rest des aktuellen Satzes ohne vorherige Anfrage ausgetauscht. Vom nachfolgenden Satz an wird wieder nachgefragt, ob ausgetauscht werden soll.

neue Anweisung: Das Austauschen wird unterbrochen, ohne dass die aktuelle Zeichenfolge ausgetauscht wird. In diesem Fall, wie auch bei der Antwort **j**, kann später mit der Anweisung **a** das Austauschen an der Unterbrechungsstelle fortgesetzt werden, falls der Editor inzwischen nicht verlassen oder mit der Datei-Anweisung **d** in eine andere Datei gewechselt wurde.

## Such-Anweisungen für strukturierte Daten

Mit den erweiterten Anweisungen zum Suchen und Zeigen (siehe Seite 307) können Sätze gesucht werden, die bestimmte Zeichenfolgen enthalten. Dabei werden die gefundenen Sätze entweder einzeln mit ihrer Umgebung angezeigt (Anweisung `zu`), oder es werden nur die gefundenen Sätze angezeigt (Anweisung `zn`). Da die Sätze mit ihrer Satznummer und mit unverändertem Text angezeigt werden, können sie ggf. korrigiert werden.

Bei der in diesem Kapitel beschriebenen Such-Anweisung, die eine datenbankähnliche Abfrage ermöglicht, muss die Suchbedingung und das Zeigeformat mit speziell dafür vorgesehenen Anweisungen zuvor definiert werden. Suchbedingung und Zeigeformat setzen sich aus mehreren Parametern zusammen.

Für die Suchbedingung kann definiert werden,

- welche Sätze zu einer logischen Einheit (z. B. einem bibliographischen Eintrag), im Folgenden Texteinheit genannt, zusammengehören,
- in welchen Textteilen (z. B. Rubriken) einer Texteinheit welche Zeichenfolge vorkommen muss, damit die Suchbedingung erfüllt ist,
- welche Zeichenfolgen (z. B. Akzente) in diesen Textteilen eliminiert oder durch andere ersetzt werden sollen, bevor geprüft wird, ob eine vorgegebene Zeichenfolge vorkommt.

Für das Zeigeformat kann definiert werden,

- welche Textteile der Texteinheit gezeigt werden sollen,
- welche Zeichenfolgen (z. B. Rubrik-Kennzeichen) in diesen Textteilen durch andere ersetzt oder eliminiert werden sollen,
- an welchen Stellen (z. B. bei jeder Rubrik) bei der Ausgabe ins Textfeld eine neue Zeile begonnen werden soll.

Erfüllt beim Suchen eine Texteinheit die Suchbedingung, so wird sie ohne Satznummer (und ohne Umgebung) entsprechend den Vorgaben des Zeigeformats im Textfeld angezeigt. Dieses Zeigeformat kann nicht zur Korrektur der Daten verwendet werden. Sollen die angezeigten Daten korrigiert werden, so müssen sie (z. B. mit der Anweisung `za, *`) zur Korrektur angefordert werden.

Alle Parameter, die zu einer Suchbedingung und einem Zeigeformat gehören, bilden eine Parametergruppe. Es können gleichzeitig bis zu neun Parametergruppen definiert sein. Sie werden mit `P1` bis `P9` bezeichnet.

## Definieren und Löschen der Parameter – allgemein

Eine Anweisung zur Definition eines Parameters hat die Form:

```
pn, xxx=vtab|stab|atab
```

Mit dieser Anweisung wird für die Parametergruppe `pn` der Parameter mit der Kennung `xxx` definiert. Für `n` ist die entsprechende Ziffer und für `xxx` die entsprechende Parameterkennung einzusetzen. An Stelle von `vtab|stab|atab` muss in



Abhängigkeit vom jeweiligen Parameter entweder eine Vergleichs-Tabelle `vtab` (siehe Seite 331), eine Such-Tabelle `stab` (siehe Seite 331) oder eine Austausch-Tabelle `atab` (siehe Seite 332) angegeben werden.

Die Definition eines Parameters gilt so lange, bis sie durch eine neue Definition geändert oder gelöscht wird. Eine Anweisung zum Löschen eines Parameters hat die Form:

`pn, xxx=`

Mit dieser Anweisung wird in der Parametergruppe `pn` der Parameter mit der Kennung `xxx` gelöscht. Gegenüber der Definition wird nur die Angabe `vtab|stab|atab` nach dem Gleichheitszeichen weggelassen.

## Definieren der Parameter für die Suchbedingung

Parameter können auch beim Aufruf des Editors definiert bzw. gelöscht werden (siehe Kommando `#EDIERE` Seite 149). Sollen Parameter automatisch beim ersten Aufruf des Editors in einer TUSTEP-Sitzung definiert werden, können sie in das Segment `EDIT` der INI-Datei (siehe Seite 89) eingetragen werden.

### Zusammenfassen von mehreren Sätzen zu einer Texteinheit

Falls jeder Satz schon eine vollständige Texteinheit enthält, sollte keiner der beiden folgenden Parameter definiert werden. Andernfalls können mit den Parametern `aa` und/oder `ae` jeweils mehrere Sätze zu einer Texteinheit zusammengefasst werden.

`pn, aa=vtab` Zeichenfolgen, die am Satzanfang den Anfang einer Texteinheit kennzeichnen.

`pn, ae=vtab` Zeichenfolgen, die am Satzende das Ende einer Texteinheit kennzeichnen.

### Auswahl der Texteinheiten durch Suchbedingungen

Für jede Parametergruppe kann eine Suchbedingung aus bis zu 9 Teilbedingungen definiert werden. Für jede Teilbedingung kann zunächst ein Textteil, der durch eine Anfangs- und eine Endekennung gekennzeichnet ist, aus der Texteinheit ausgewählt werden. In diesem Textteil wird dann, ggf. nachdem noch Zeichenfolgen ersetzt wurden, geprüft, ob eine der vorgegebenen Zeichenfolgen vorkommt. Wenn eine solche vorkommt, ist die Teilbedingung erfüllt, andernfalls ist sie nicht erfüllt.

Damit die Suchbedingung erfüllt ist (nur in diesem Fall wird die Texteinheit gezeigt), müssen alle angegebenen Teilbedingungen erfüllt sein. Dies entspricht einem logischen UND. Eine Verknüpfung mit einem logischen ODER kann z. Z. nur dadurch erreicht werden, dass zu einer Teilbedingung mehr als eine Zeichenfolge angegeben wird. Darüber hinausgehende logische Verknüpfungen sind noch nicht möglich.

- pn, avm=stab** Zeichenfolgen, die den Anfang des Textteils für die Teilbedingung  $m$  ( $m = 1$  bis  $9$ ) kennzeichnen. Enthält die Texteinheit keine der angegebenen Zeichenfolgen, wird kein Textteil ausgewählt; die Teilbedingung ist in diesem Fall nicht erfüllt. Wird dieser Parameter nicht definiert, beginnt der Textteil am Anfang der Texteinheit.
- pn, evm=stab** Zeichenfolgen, die das Ende des Textteils für die Teilbedingung  $m$  ( $m = 1$  bis  $9$ ) kennzeichnen. Enthält die Texteinheit, ggf. nach der Anfangskennung für den jeweiligen Textteil, keine der angegebenen Zeichenfolgen oder wird dieser Parameter nicht definiert, endet der Textteil am Ende der Texteinheit.
- pn, xvm=atab** Zeichenfolgenpaare (und Ausnahmezeichenfolgen), deren jeweils erste Zeichenfolge durch die jeweils zweite Zeichenfolge im Textteil für die Teilbedingung  $m$  ( $m = 1$  bis  $9$ ) ersetzt werden soll.
- pn, zvm=stab** Zeichenfolgen, von denen mindestens eine im (ggf. modifizierten) Textteil für die Teilbedingung  $m$  ( $m = 1$  bis  $9$ ) vorkommen muss.

## Definieren der Parameter für das Zeigeformat

### Auswahl der Textteile zur Ausgabe

Falls eine Texteinheit vollständig angezeigt werden soll, braucht keiner der beiden folgenden Parameter definiert zu werden. Andernfalls können mit den Parametern  $azm$  und  $ezm$  ( $m = 1$  bis  $9$ ) bis zu neun Textteile, die im Textfeld angezeigt werden sollen, aus der Texteinheit ausgewählt werden. Die Textteile werden in der Reihenfolge ihrer Nummer ( $m$ ) angeordnet. In der Nummerierung können Lücken gelassen werden. In jedem so ausgewählten Textteil können mit dem Parameter  $xzm$  vor der Ausgabe noch Zeichenfolgen ersetzt werden.

- pn, azm=stab** Zeichenfolgen, die den Anfang des Textteils  $m$  ( $m = 1$  bis  $9$ ) kennzeichnen. Enthält die Texteinheit keine der angegebenen Zeichenfolgen, wird für den Textteil nichts ausgewählt. Wird dieser Parameter nicht definiert, beginnt der Textteil am Anfang der Texteinheit, falls der entsprechende Parameter für die Endekennung definiert ist.
- pn, ezm=stab** Zeichenfolgen, die das Ende des Textteils  $m$  ( $m = 1$  bis  $9$ ) kennzeichnen. Enthält die Texteinheit, ggf. nach der Anfangskennung für den jeweiligen Textteil, keine der angegebenen Zeichenfolgen oder wird dieser Parameter nicht definiert, endet der Textteil am Ende der Texteinheit.
- pn, xzm=atab** Zeichenfolgenpaare (und Ausnahmezeichenfolgen), deren jeweils erste Zeichenfolge durch die jeweils zweite Zeichenfolge im Textteil  $m$  ( $m = 1$  bis  $9$ ) vor der Ausgabe ersetzt werden soll.

## Zeileneinteilung bei der Anzeige

Mit den folgenden drei Parametern kann bestimmt werden, an welchen Stellen bei der Anzeige im Textfeld eine neue Zeile begonnen werden soll.

**pn, za=stab** Zeichenfolgen, vor denen jeweils eine neue Zeile begonnen werden soll (Zeilenanfänge).

**pn, ze=stab** Zeichenfolgen, nach denen jeweils eine neue Zeile begonnen werden soll (Zeilenenden).

**pn, zw=stab** Zeichenfolgen, die nicht angezeigt werden, sondern nur einen Zeilenwechsel bewirken sollen.

## Suchen und Zeigen

Nachdem die Parameter definiert sind, folgt die eigentliche Suche. Bei jeder der folgenden Such-Anweisungen wird die jeweils aktuelle Parametergruppe verwendet. Als aktuelle Parametergruppe gilt diejenige, in der zuletzt ein Parameter definiert oder gelöscht wurde bzw. deren Parameter zuletzt durch eine der Anweisungen **p** oder **pn** ( $n = 1$  bis  $9$ ) angezeigt wurden.

**s, ber** Suchen in einem Bereich nach Texteinheiten, die die Suchbedingung der aktuellen Parametergruppe erfüllen.

Wird für die Bereichsangabe **ber** die Form  $(pos, pos)$  verwendet, so kann die erste Positionsangabe größer als die zweite sein; in diesem Fall wird dieser Bereich rückwärts durchsucht.

**s, pos** Suchen ab einer bestimmten Satzposition nach Texteinheiten, die die Suchbedingung der aktuellen Parametergruppe erfüllen.

**sa** Suchen vom Dateianfang an nach Texteinheiten, die die Suchbedingung der aktuellen Parametergruppe erfüllen.

**se** Suchen vom Dateiende her nach Texteinheiten, die die Suchbedingung der aktuellen Parametergruppe erfüllen.

Nach diesen Such-Anweisungen sind folgende Eingaben möglich:

leere Eingabe: In der gleichen Richtung weitersuchen.

**w** Weiter suchen, aber gefundene Einheiten nicht mehr zeigen, sondern nur am Ende die Anzahl der gefundenen Einheiten anzeigen.

**sr** Rückwärts (d. h. zum Dateianfang hin) weitersuchen.

**sv** Vorwärts (d. h. zum Dateiende hin) weitersuchen.

neue Anweisung: Das Suchen wird unterbrochen. In diesem Fall kann später mit der Anweisung **s** (oder einer der Anweisungen **sr** und **sv**) das Suchen an der Unterbrechungsstelle fortgesetzt werden, falls inzwischen nicht die Editor-Datei gewechselt oder der Editor verlassen wurde.

## Parameter abfragen/löschen/kopieren

- p** Zeigt eine Übersicht der Parameter der einzelnen Parametergruppen, beginnend mit der aktuellen Parametergruppe, nacheinander im Editorfenster an. Mit dieser Anweisung kann also auch festgestellt werden, welches die aktuelle Parametergruppe ist.
- Wenn nicht alle Parameter einer Parametergruppe zusammen ins Editorfenster passen, wird nach einer leeren Eingabe die nächste Portion angezeigt. Andernfalls werden nach einer leeren Eingabe die Parameter der nächsten Parametergruppe angezeigt.
- p<sub>n</sub>** Zeigt eine Übersicht der Parameter der Parametergruppe *n* im Editorfenster an. Für *n* ist die entsprechende Ziffer einzusetzen.
- Wenn nicht alle Parameter zusammen ins Editorfenster passen, wird nach einer leeren Eingabe die nächste Portion angezeigt.
- p<sub>n</sub>=** Löscht alle Parameter der Parametergruppe *n*.
- Wie einzelne Parameter einer Parametergruppe gelöscht werden können, ist weiter oben (auf Seite 313) beschrieben.
- p!** Löscht alle Parameter in allen Parametergruppen.
- p<sub>n1</sub>=p<sub>n2</sub>** Löscht alle Parameter der Parametergruppe *n1* und kopiert dann alle Parameter der Parametergruppe *n2* in die Parametergruppe *n1*.

## Suchen und Prüfen von Tags

Ein Tag ist eine in spitzen Klammern eingeschlossene Zeichenfolge. Es gibt Anfangs-Tags, Ende-Tags und leere Tags:

Anfangs-Tag:

```
<name>
<name attribut1=wert1, attribut2=wert2, ... >
```

Ende-Tag:

```
</name>
```

Leeres Tag:

```
<name/>
<name attribut1=wert1, attribut2=wert2, ... />
```

Damit die nachfolgend beschriebenen Anweisungen die angegebene Wirkung haben, müssen folgende Bedingungen erfüllt sein:

- Die spitzen Klammern dürfen nur als Anfangs- bzw. Endekennung von Tags, in Akzent-Codierungen (z. B. »%<e«) oder in den Codes für doppelte Anführungszeichen (»#. <« und »#. >«) vorkommen; andere spitze Klammern müssen z. B. mit ^< bzw. ^> codiert sein.
- Anfangs- und Endekennung eines Tags (beide spitze Klammern) müssen im selben Satz stehen.
- Tag- und Attribut-Namen dürfen aus Buchstaben, Ziffern, Minuszeichen, Punkt, Doppelpunkt und »\_« bestehen; das erste Zeichen darf kein Minuszeichen und kein Punkt sein. Tag- und Attribut-Namen, die dieser Bedingung nicht entsprechen, können mit der Anweisung `tpn` (siehe Seite 323) als legale Namen definiert werden.

Leere Tags werden beim Prüfen auf Paarigkeit übergangen. Enthalten die Daten Anfangs-Tags, zu denen grundsätzlich keine Ende-Tags vorhanden sind, können diese Tags mit den Anweisungen `tp1` und `ts1` (siehe Seite 323) als leere Tags definiert werden, damit sie beim Prüfen auf Paarigkeit ebenfalls übergangen werden.

Prüfen von Tags:

Mit den im folgenden beschriebenen Anweisungen wird die korrekte Schachtelung und die Syntax der Tags überprüft. Soll darüber hinaus auch geprüft werden, ob nur vorgesehene Tag-Namen, Attribut-Namen und Attribut-Werte verwendet wurden, und ob die Tags bzw. Text an der jeweiligen Stelle in der Tag-Hierarchie erlaubt sind, müssen zuvor alle erlaubten Tags in einer Tag-Gruppe definiert bzw. es muss zuvor eine Tag-Gruppe eingestellt werden. Die Anweisungen dafür sind im Kapitel »Tag-Prüfung definieren/...« (siehe Seite 300) beschrieben.

Wird ein Tag mit einem Syntaxfehler gefunden, so wird das Prüfen beendet und der entsprechende Satz mit Umgebung angezeigt; wird ein Tag ohne entsprechenden Partner gefunden wird nur eine entsprechende Fehlermeldung angezeigt.

**tp** ist gleichbedeutend mit **tpv**

**tpv**, *ber*, *mds* Tag-Prüfung vorwärts im angegebenen Bereich

Wird statt einer Bereichsangabe *ber* eine Positionsangabe *pos* verwendet, so wird von dieser Satzposition an vorwärts bis zum Dateiende geprüft.

Wenn für *mds* ein Minuszeichen angegeben wird, werden XML-Kommentare, die in »<!--« und »-->« eingeschlossen sind, beim Prüfen übergangen; wenn ein Pluszeichen angegeben wird, werden die Zeichenfolgen »<!--« und »-->« auf Paarigkeit überprüft und die dazwischen stehenden Daten geprüft; wenn für *mds* nichts angegeben ist, werden »<!--« und »-->« wie Anfangs- und Ende-Tags behandelt.

**tpr**, *ber*, *mds* Tag-Prüfung rückwärts im angegebenen Bereich

Wie die zuvor beschriebene Anweisung **tpv**, jedoch wird nicht vorwärts (d. h. zum Dateiende hin), sondern rückwärts (d. h. zum Dateianfang hin) geprüft.

Anzeigen offener Tags

**tz**, *ber*, *mds* Tag-Zeige

Sucht im angegebenen Bereich nach offenen Tags. Die in den durchsuchten Daten gefundenen Tags werden auf Paarigkeit geprüft. Wird ein Ende-Tag ohne entsprechenden Partner gefunden, so wird die Suche beendet. Der Satz mit dem Tag ohne Partner wird zur aktuellen Satzposition.

Wird statt einer Bereichsangabe *ber* eine Positionsangabe *pos* verwendet, so werden die Tags von dieser Satzposition an vorwärts bis vor die aktuelle Satzposition geprüft.

Wird keine Bereichsangabe *ber* angegeben, so werden die Tags vom Dateianfang an bis zur aktuellen Satzposition geprüft.

Falls sich die gemerkte Tag-Position im Satz mit der aktuellen Satzposition befindet, werden die Tags bis zu dieser Tag-Position geprüft.

Eine Tag-Position besteht aus der Satznummer des entsprechenden Satzes und einer Zahl, die angibt, das wievielte Tag es in diesem Satz ist. Eine Tag-Position kann mit dem Steuerbefehl `SET_TAG_POS` (siehe Seite 387) gemerkt werden.

Falls nicht zu allen Anfangs-Tags die entsprechenden Ende-Tags gefunden wurden, werden in der Meldungszeile statt einer entsprechenden Fehlermeldung (wie bei den zuvor beschriebenen Anweisungen) die Namen dieser Anfangs-Tags angezeigt. Hinter jedem Tag wird in eckigen Klammern angegeben, das wievielte gleichnamige Tag es ist, das jeweils dieselben (!) übergeordneten Tags hat; beim jeweils ersten Tag wird diese Angabe (»[1]«) unterdrückt. Wird einer dieser Namen mit der Maus angeklickt, so wird die Umgebung des entsprechenden Tags im Textfeld angezeigt.

Wenn für `mds` ein Minuszeichen angegeben wird, werden XML-Kommentare, die in »<!--« und »-->« eingeschlossen sind beim Prüfen übergangen; wenn ein Pluszeichen angegeben wird, werden die Zeichenfolgen »<!--« und »-->« auf Paarigkeit überprüft und die dazwischen stehenden Daten geprüft; wenn für `mds` nichts angegeben ist, werden »<!--« und »-->« wie Anfangs- und Ende-Tags behandelt.

#### Hinweise:

Um die Namen der Tags anzuzeigen, die an einer bestimmten Stelle eines Satzes offen sind, ist es sinnvoll, den Cursor an diese Stelle zu positionieren und den Steuerbefehl `SHOW_TAGS` (siehe Seite 388) aufzurufen. Er setzt die Tag-Position und die Satzposition auf die der Cursor-Position entsprechenden Werte und führt dann die Anweisung `tz` aus.

Nach der Ausführung der Anweisung `tz` können mit dem Steuerbefehl `XPATH_MRK` (siehe Seite 388) die Namen der offenen Tags in den Editor-Zwischenspeicher kopiert werden.

#### Suchen von Pfaden

`tz,ber,mds,pfad` Tag-Zeige

Sucht im angegebenen Bereich nach einem mit `pfad` angegebenen Pfad, d. h. nach einem Tag, das in den Daten an einer bestimmten Stelle in der Tag-Hierarchie steht. Wird ein entsprechendes Tag gefunden, wird die Suche beendet und die Umgebung des Tags angezeigt.

Die in den durchsuchten Daten gefundenen Tags werden auf Paarigkeit geprüft. Wird ein Ende-Tag ohne entsprechenden Partner gefunden, so wird die Suche beendet. Der Satz mit dem Tag ohne Partner wird zur aktuellen Satzposition.

Für `pfad` kann mit Hilfe von Pfaden angegeben werden, welche Tags gesucht werden sollen. Ein Pfad kann sich aus folgenden drei Angaben zusammensetzen: `<tag>` für ein Tag mit dem Namen `tag`, `<*>` für ein Tag mit einem beliebigen Namen, `*` für null oder beliebig viele Tags mit jeweils beliebigem Namen. Unmittelbar hinter den Angaben `<tag>` und `<*>` kann jeweils noch eine Zahl `n` in eckigen Klammern angegeben werden. Sie beschränkt das davor stehende Tag auf der aktuellen Hierarchiestufe auf das `n`-te Tag mit dem betreffenden Namen. Die einzelnen Pfade müssen durch ein frei wählbares Begrenzungszeichen voneinander getrennt sein. Vor der ersten und nach der letzten Pfadangabe muss ebenfalls das Begrenzungszeichen angegeben werden. Es können erlaubte Pfade und unerlaubte Pfade (Ausnahmen) angegeben werden. Durch Angabe von zwei aufeinander folgenden Begrenzungszeichen kann beliebig oft zwischen erlaubten und unerlaubten Pfaden gewechselt werden. Beim Prüfen, ob ein Tag an einer bestimmten Stelle in den Daten erlaubt ist, werden die Pfade in der angegebenen Reihenfolge überprüft, bis eine Übereinstimmung gefunden wird.

Wenn für `mds` ein Minuszeichen angegeben wird, werden XML-Kom-

mentare, die in »<!--« und »-->« eingeschlossen sind, beim Prüfen übergangen; wenn ein Pluszeichen angegeben wird, werden die Zeichenfolgen »<!--« und »-->« auf Paarigkeit überprüft und die dazwischen stehenden Daten geprüft; wenn für `mds` nichts angegeben ist, werden »<!--« und »-->« wie Anfangs- und Ende-Tags behandelt.

Wenn für `mds` ein Schrägstrich angegeben wird, ist dies gleichbedeutend mit der Angabe eines Minuszeichens, jedoch gilt für die Pfadangaben folgende Regelung: Es kann nur ein einziger Pfad angegeben werden. Er kann sich nur aus Angaben der Form `/tag` für ein Tag mit dem Namen `tag` zusammensetzen. Unmittelbar hinter der Angabe `/tag` kann jeweils noch eine Zahl `n` in eckigen Klammern angegeben werden. Sie beschränkt das davor stehende Tag auf der aktuellen Hierarchiestufe auf das `n`-te Tag mit dem betreffenden Namen.

Falls ein gesuchtes Tag gefunden wurde, kann mit der Anweisung `z` oder mit einer »leeren Eingabe« die Suche fortgesetzt werden.

### Anzeigen einer Tag-Liste

Mit den drei nachfolgend beschriebenen Anweisungen kann jeweils eine Liste der in den Daten vorkommenden Tags angezeigt werden. Soll eines der angezeigten Tags in den Daten gesucht und die Umgebung angezeigt werden, so muss die entsprechende Zeile davon mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Return-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der linken Maustaste angeklickt werden; soll eine entsprechende Anweisung ins Editorfenster ausgegeben werden, damit sie ggf. verändert werden kann, so muss die entsprechende Zeile mit Pfeil nach unten/oben ausgewählt werden und die Auswahl mit der Enter-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der rechten Maustaste angeklickt werden.

#### `t1,ber,mds,pfad` Tag-Liste

Sucht im angegebenen Bereich nach Tags und zeigt sie in einer Liste an.

Die in den durchsuchten Daten gefundenen Tags werden auf Paarigkeit geprüft. Wird ein Ende-Tag ohne entsprechenden Partner gefunden, so wird die Suche beendet. Der Satz mit dem Tag ohne Partner wird zur aktuellen Satzposition.

Wenn für `mds` ein Minuszeichen angegeben wird, werden XML-Kommentare, die in »<!--« und »-->« eingeschlossen sind, beim Prüfen übergangen; wenn ein Pluszeichen angegeben wird, werden die Zeichenfolgen »<!--« und »-->« auf Paarigkeit überprüft und die dazwischen stehenden Daten geprüft; wenn für `mds` nichts angegeben ist, werden »<!--« und »-->« wie Anfangs- und Ende-Tags behandelt.

Für `pfad` kann mit Hilfe von Pfaden angegeben werden, welche Tags in die Liste aufgenommen werden sollen. Ein Pfad kann sich aus folgenden drei Angaben zusammensetzen: `<tag>` für ein Tag mit dem Namen `tag`, `<*>` für ein Tag mit einem beliebigen Namen, `*` für null oder beliebig viele Tags mit jeweils beliebigem Namen. Unmittelbar hinter den An-



gaben `<tag>` und `<*>` kann jeweils noch eine Zahl `n` in eckigen Klammern angegeben werden. Sie beschränkt das davor stehende Tag auf der aktuellen Hierarchiestufe auf das `n`-te Tag mit dem betreffenden Namen. Ein Pfad darf nicht mit `»*«` enden. Die einzelnen Pfade müssen durch ein frei wählbares Begrenzungszeichen voneinander getrennt sein. Vor der ersten und nach der letzten Pfadangabe muss ebenfalls das Begrenzungszeichen angegeben werden. Es können erlaubte Pfade und unerlaubte Pfade (Ausnahmen) angegeben werden. Durch Angabe von zwei aufeinander folgenden Begrenzungszeichen kann beliebig oft zwischen erlaubten und unerlaubten Pfaden gewechselt werden. Beim Prüfen, ob ein Tag an einer bestimmten Stelle in den Daten in die Liste aufgenommen werden soll, werden die Pfade in der angegebenen Reihenfolge überprüft, bis eine Übereinstimmung gefunden wird.

Die Anzeige kann durch Drücken der Leertaste oder Escape-Taste wieder gelöscht werden.

**t1h**, *ber, mds, pfad* Tag-Liste hierarchisch

wie **t1**, jedoch werden in der Tag-Liste zu jedem Tag auch noch die übergeordneten Tags angezeigt. Die Liste wird beim hierarchisch obersten Tag beginnend alphabetisch sortiert.

**t1i**, *ber, mds, pfad* Tag-Liste invers

wie **t1h**, jedoch wird die Tag-Liste beim hierarchisch untersten Tag beginnend alphabetisch sortiert.

Suchen von Tags:

Wird ein gesuchtes Tag gefunden, so wird die Suche beendet. Die Position des gefundenen Tags wird gemerkt; sie gilt zugleich als aktuelle Tag-Position.

Eine Tag-Position besteht aus der Satznummer des entsprechenden Satzes und einer Zahl, die angibt, das wievielte Tag es in diesem Satz ist. Eine Tag-Position kann auch mit dem Steuerbefehl `SET_TAG_POS` (siehe Seite 387) gemerkt werden.

Wird kein gesuchtes Tag gefunden, so wird die zuvor gemerkte Tag-Position gelöscht.

Die in den durchsuchten Daten gefundenen Tags werden auf Paarigkeit geprüft. Wird ein Tag ohne entsprechenden Partner gefunden, so wird die Suche beendet. Der Satz mit dem Tag ohne Partner wird zur aktuellen Satzposition; die Position des Tags wird nicht gemerkt, die zuvor gemerkte Tag-Position bleibt unverändert.

**tsv**, *ber, mds* Tag-Suche vorwärts

Sucht im angegebenen Bereich vorwärts nach einem Ende-Tag, zu dem in den durchsuchten Daten kein entsprechendes Anfangs-Tag gefunden wurde.

Wird statt einer Bereichsangabe *ber* eine Positionsangabe *pos* verwendet, so wird von dieser Satzposition an vorwärts bis zum Dateiende gesucht.

Die Suche beginnt am Anfang des Satzes. Soll die Suche mitten im Satz

beginnen (weil der Satz mehrere Tags enthält), so kann mit dem Steuerbefehl `SET_TAG_POS` (siehe Seite 387) die entsprechende Position gemerkt werden und mit einer Anweisung `tsv` ohne Bereichsangabe die Suche ab dieser Position gestartet werden.

Falls ein gesuchtes Tag gefunden wurde, kann mit der nachfolgend beschriebenen Anweisung `tsv` oder mit einer »leeren Eingabe« die Suche fortgesetzt werden.

Wenn für `mds` ein Minuszeichen angegeben wird, werden XML-Kommentare, die in `<!--` und `-->` eingeschlossen sind beim Suchen übergangen; wenn ein Pluszeichen angegeben wird, werden die Zeichenfolgen `<!--` und `-->` auf Paarigkeit überprüft und die dazwischen stehenden Daten durchsucht; wenn für `mds` nichts angegeben ist, werden `<!--` und `-->` wie Anfangs- und Ende-Tags behandelt.

**Hinweis:** Um von einer bestimmten Stelle innerhalb eines Satzes aus vorwärts das nächste Ende-Tag zu suchen, ist es sinnvoll, den Cursor an diese Stelle zu positionieren und den Steuerbefehl `SHW_END_TAG` (siehe Seite 388) aufzurufen. Er setzt die Tag-Position und die Satzposition auf die der Cursor-Position entsprechenden Werte und führt dann die Anweisung `tsv` aus.

**t<sub>sr</sub>, ber, mds** Tag-Suche rückwärts

Sucht im angegebenen Bereich rückwärts nach einem Anfangs-Tag, zu dem in den durchsuchten Daten kein entsprechendes Ende-Tag gefunden wurde.

Wird statt einer Bereichsangabe `ber` eine Positionsangabe `pos` verwendet, so wird von dieser Satzposition an rückwärts bis zum Dateianfang gesucht.

Die Suche beginnt am Ende des Satzes. Soll die Suche mitten im Satz beginnen (weil der Satz mehrere Tags enthält), so kann mit dem Steuerbefehl `SET_TAG_POS` (siehe Seite 387) die entsprechende Position gemerkt werden und mit einer Anweisung `tsr` ohne Bereichsangabe die Suche ab dieser Position gestartet werden.

Falls ein gesuchtes Tag gefunden wurde, kann mit der nachfolgend beschriebenen Anweisung `tsr` oder mit einer leeren Eingabe die Suche fortgesetzt werden.

Wenn für `mds` ein Minuszeichen angegeben wird, werden XML-Kommentare, die in `<!--` und `-->` eingeschlossen sind, beim Prüfen übergangen; wenn ein Pluszeichen angegeben wird, werden die Zeichenfolgen `<!--` und `-->` auf Paarigkeit überprüft und die dazwischen stehenden Daten geprüft; wenn für `mds` nichts angegeben ist, werden `<!--` und `-->` wie Anfangs- und Ende-Tags behandelt.

**Hinweis:** Um von einer bestimmten Stelle innerhalb eines Satzes aus rückwärts das nächste Anfangs-Tag zu suchen, ist es sinnvoll, den Cursor an diese Stelle zu positionieren und den Steuerbefehl `SHW_STRT_TAG` (siehe Seite 388) aufzurufen. Er setzt die Tag-Position und die Satzposition auf die der Cursor-Position entsprechenden Werte und führt dann die Anweisung `tsr` aus.

Definieren zusätzlicher Tag- und Attribut-Namen:

Beim Prüfen und Suchen von Tags mit den zuvor beschriebenen Anweisungen dürfen Tag- und Attribut-Namen aus Buchstaben, Ziffern, Minuszeichen, Punkt, Doppelpunkt und »\_« bestehen; das erste Zeichen darf kein Minuszeichen und kein Punkt sein. Mit der folgenden Anweisung können Tag- und Attribut-Namen, die dieser Bedingung nicht entsprechen, als legale Namen definiert werden.

**tpn=rtab** Tag-Prüfung, Namen definieren

Tags, die einen in der Tabelle `rtab` angegebenen Namen haben, werden beim Prüfen und Suchen von Tags mit den zuvor beschriebenen Anweisungen als legale Namen angesehen.

Groß- und Kleinbuchstaben werden unterschieden.

**tpn=** Tag-Prüfung, Definition der Namen löschen.

Ob und ggf. welche Namen als zulässig definiert sind, kann mit der Anweisung `g, tpn` angezeigt werden.

Definieren der leeren Tags:

**tpl=rtab** Tag-Prüfung, leere Tags definieren

**tsl=rtab** Tag-Suche, leere Tags definieren

Beide Anweisungen sind gleichwertig und definieren eine Tabelle mit Namen von »leeren« Tags. Die Tabelle gilt jeweils sowohl für das Prüfen als auch für das Suchen von Tags.

Anfangs-Tags, die einen in der Tabelle `rtab` angegebenen Namen haben, werden beim Prüfen und Suchen von Tags mit den zuvor beschriebenen Anweisungen wie leere Tags behandelt und beim Prüfen der Tags auf Paarigkeit übergangen.

Ende-Tags, die einen in der Tabelle `rtab` angegebenen Namen haben, werden beim Prüfen und Suchen von Tags mit den zuvor beschriebenen Anweisungen wie Ende-Tags behandelt. Sollen Ende-Tags wie leere Tags behandelt werden, müssen deren Namen jeweils mit einem vorangestellten Schrägstrich zusätzlich angegeben werden.

Groß- und Kleinbuchstaben werden unterschieden.

**tpl=** Tag-Prüfung, Definition der leeren Tags löschen

**tsl=** Tag-Suche, Definition der leeren Tags löschen

Beide Anweisungen sind gleichwertig und löschen die Tabelle mit Namen von »leeren« Tags.

Ob und ggf. welche leeren Tags definiert sind, kann mit den Anweisung `g, tpl` und `g, tsl` angezeigt werden.

## Prüfen von Klammern

Mit den folgenden Anweisungen können Klammern auf Paarigkeit und auf zulässige Verschachtelung überprüft werden. Außerdem kann geprüft werden, ob bestimmte Zeichenfolgen nur außerhalb bzw. innerhalb von bestimmten Klammern vorkommen.

Klammern dürfen nur paarweise vorkommen. Steht eine öffnende Klammer innerhalb eines anderen Klammerpaares, so muss die dazugehörige schließende Klammer ebenfalls innerhalb des selben Klammerpaares stehen; es ist also bei entsprechenden Angaben z. B. »(...[...])...« erlaubt, keinesfalls aber »(...[...])...«.

Wenn »(...[...])...« erlaubt ist, ist damit auch automatisch »(...)...« erlaubt; wenn »{...(...[...])...}« erlaubt ist, ist auch »{...(...)...}« und »{...}« erlaubt; usw.

Welche Zeichen bzw. Zeichenfolgen als öffnende und schließende Klammern gelten, und welche Zeichen bzw. Zeichenfolgen innerhalb welcher Klammern vorkommen dürfen, muss in den Anweisungen angegeben werden.

**tpv**, ber, ( ), atab

Prüft die Klammern und Zeichenfolgen in den Sätzen des angegebenen Bereichs, beginnend mit dem ersten Satz des Bereichs.

Wird statt einer Bereichsangabe *ber* eine Positionsangabe *pos* verwendet, so wird von dieser Satzposition an vorwärts bis zum Dateiende geprüft.

**tpr**, ber, ( ), atab

Prüft die Klammern und Zeichenfolgen in den Sätzen des angegebenen Bereichs, beginnend mit dem letzten Satz des Bereichs.

Wird statt einer Bereichsangabe *ber* eine Positionsangabe *pos* verwendet, so wird von dieser Satzposition an rückwärts bis zum Dateianfang geprüft.

Für *atab* werden analog zu einer Austausch-Tabelle (siehe Seite 332) paarweise Zeichenfolgen angegeben, deren jeweils erste Zeichenfolge als Klammer bzw. als zu prüfende Zeichenfolge gilt, deren Eigenschaften durch die jeweils zweite Zeichenfolge in der Form  $x_n$  festgelegt wird. Außerdem können ggf. Zeichenfolgen (z. B. »%{1-2}{%}« für Akzent-Codierungen, die Klammern enthalten können, oder »#{1-4}{!}« für entsprechend codierte Sonderzeichen) angegeben werden, die übergangen werden sollen (Ausnahmezeichenfolgen).

In der jeweils zweiten Zeichenfolge steht *x* für eine oder mehrere gleiche Klammern oder für einen oder mehrere Klammeraffen; *n* steht für eine Zahl von 1 bis 99.

Mit Klammern wird festgelegt, ob es sich um eine öffnende oder schließende Klammer handelt, und auf welcher Klammerstufe die jeweilige Klammer vorkommen darf. » (« und ») « bezeichnen öffnende/schließende Klammern, die nur auf Klammerstufe 1 (also nicht innerhalb eines Klammerpaares) vorkommen dürfen; » ( (« bzw. ») ) « bezeichnen öffnende/schließende Klammern, die nur auf Klammerstufe 2 (also nur innerhalb eines Klammerpaares) vorkommen dürfen; » ( ( (« bzw. ») ) ) « bezeichnen

öffnende/schließende Klammern, die nur auf Klammerstufe 3 (also nur innerhalb zweier Klammerpaare) vorkommen dürfen; usw.

Mit Klammeraffen wird festgelegt, dass es sich um eine zu prüfende Zeichenfolge handelt, und auf welcher Klammerstufe sie vorkommen darf. »@« bezeichnet eine Zeichenfolge, die nur auf Klammerstufe 1 (also nicht innerhalb eines Klammerpaares) vorkommen darf; »@@« bezeichnet eine Zeichenfolge, die nur auf Klammerstufe 2 (also nur innerhalb eines Klammerpaares) vorkommen darf; »@@@« bezeichnet eine Zeichenfolge, die nur auf Klammerstufe 3 (also nur innerhalb zweier Klammerpaare) vorkommen darf; usw.

Bei Klammern wird mit der Zahl festgelegt, welche Klammern zusammengehören und welche Klammern innerhalb welcher Klammern vorkommen dürfen. Zusammengehörende Klammern müssen mit der gleichen Zahl gekennzeichnet sein. Eine Klammer darf nur innerhalb einer anderen Klammer vorkommen, wenn sie auf der nächst höheren Klammerstufe vorkommen darf und mit der gleichen Zahl gekennzeichnet ist.

Bei Klammeraffen (zu prüfenden Zeichenfolgen) wird mit der Zahl festgelegt, innerhalb welcher Klammern die Zeichenfolgen vorkommen dürfen. Bei Zeichenfolgen, die auf Klammerstufe 1 (nur außerhalb von Klammern) vorkommen dürfen, ist die für  $n$  angegebene Zahl bedeutungslos; sie muss aber trotzdem angegeben werden.

Gegebenenfalls können mehrere Angaben der Form  $x_n$  durch Komma getrennt angegeben werden.

Neben den oben beschriebenen normalen Klammern gibt es zusätzlich übergeordnete Klammern. Für sie gelten die gleichen Regeln. Sie werden jedoch statt mit runden Klammern mit eckigen gekennzeichnet. Mit den übergeordneten Klammern können Klammerbereiche gebildet werden. Mit jeder übergeordneten öffnenden Klammer wird der jeweils aktuelle Klammerbereich unterbrochen und ein neuer begonnen; mit jeder übergeordneten schließenden Klammer wird der jeweils aktuelle Klammerbereich beendet und der zuvor unterbrochene Klammerbereich fortgesetzt. Wird statt eines Klammeraffen ein senkrechter Strich angegeben, so wird mit der entsprechenden Zeichenfolge der aktuelle Klammerbereich beendet und ein neuer begonnen. In jedem Klammerbereich wird die Paarigkeit und die Verschachtelung der normalen Klammern getrennt geprüft.

Beispiele:

Prüfen von runden und eckigen Klammern, wobei die eckigen Klammern auch innerhalb von runden vorkommen dürfen:

... (... ) ... (... [ ... ] ... ) ... [ ... ] ...

Da (... [ ... ] ...) automatisch auch (... ) beinhaltet, genügt es

... (... [ ... ] ...) ... [ ... ] ... zu deklarieren.

Um die Angaben zu `atab` zusammenzustellen, empfiehlt sich, folgendes Vorgehen: Jede Klammerkombination wird in eine eigene Zeile geschrieben, dahinter die Deklarationen der einzelnen Klammern. Die erste öffnende Klammer wird mit »(« de-

klariert, die zweite mit »((« usw. Die letzte schließende Klammer wird mit »))« deklariert, die zweitletzte mit »))« usw. In der ersten Zeile wird bei allen Klammern die Zahl 1 angegeben, in der zweiten die Zahl 2 usw.

```
( [ ] )      ( = (1, [ = ((1, ] = ))1, ) = )1
[ ]          [ = (2, ] = )2
```

Diese Deklarationen ergeben zusammengefasst:

```
tpv,, ( ) , ' ( ' (1'\ [ ' ( (1, (2'\ ] '))1, )2' ) ' )1'
```

oder

```
tpv,, ( ) , ' ( ' (1' ) ' )1'\ [ ' (2, ((1'\ ] ' )2, ) )1'
```

Prüfen von runden und eckigen Klammern und darin erlaubten bzw. nicht erlaubten Zeichenfolgen: Die eckigen Klammern dürfen auch innerhalb von runden vorkommen. Der senkrechte Strich darf nur innerhalb von eckigen Klammern vorkommen. Der Stern darf nur innerhalb von runden Klammern vorkommen, nicht aber innerhalb von eckigen Klammern.

```
... (...*...) ... (...*... [... | ...] ...*...) ... [... | ...] ...
```

Da (... [...] ...) automatisch auch (...) beinhaltet

und ...\*... [...] auch [...] ...\*... beinhaltet, genügt es

```
... (...*... [... | ...] ...) ... [... | ...] ... zu deklarieren.
```

Mit den Klammern wird wie im vorangehenden Beispiel beschrieben verfahren. Die zu prüfenden Zeichenfolgen werden mit so vielen Klammeraffen deklariert, wie eine öffnende Klammer an derselben Stelle mit Klammern deklariert würde.

```
( * [ | ] )      ( = (1, * = @@1, [ = ((1, | = @@@1, ] = ))1, ) = )1
[ | ]           [ = (2, | = @@2, ] = )2
```

Diese Deklarationen ergeben zusammengefasst:

```
tpv,, ( ) , ' ( ' (1'\ * '@@1'\ [ ' ( (1, (2'\ | '@@@1, @@2'\ ] '))1, )2' ) ' )1'
```

oder

```
tpv,, ( ) , ' ( ' (1' ) ' )1'\ [ ' (2, ((1'\ ] ' )2, ) )1'\ * '@@1' | '@@2, @@@1'
```

Prüfen von »Klammern« in eigenständigen Klammerbereichen: Die Zeichenfolge #/+ gilt als öffnende Klammer, die Zeichenfolge #- als schließende. Mit @F+ wird ein Klammerbereich unterbrochen und ein neuer begonnen, mit @F- endet ein Klammerbereich und der zuvor unterbrochene wird fortgesetzt. Mit &! wird ein Klammerbereich beendet und ein neuer begonnen, sowohl innerhalb von @F+ und @F- als auch außerhalb davon.

Wenn @F+ für Fußnotenanzug, @F- für Fußnotenende und &! für neuer Abschnitt steht, müssen die Codierungen für Anfang (#/+) und Ende (#/-) der kursiven Schrift sowohl innerhalb der Fußnoten als auch innerhalb jedes Abschnitts paarig sein.

```
... #/+ ... @F+ ... &! ... #/+ ... #/- ... @F- ... #/- ... &! ... #/+ ... #/- ...
```

Mit den übergeordneten Klammern für die Kennzeichnung der Klammerbereiche

wird genauso verfahren wie mit den normalen Klammern, mit den Zeichenfolgen zur Unterteilung in Klammerbereiche genauso wie mit den zu prüfenden Zeichenfolgen.

```
#/+ #-          #/+ = (1, #- = ) 1
@F+ &! @F- &!   @F+ = [2, &! = ||2, @F- = ]2, &! = |2
```

Diese Deklarationen ergeben zusammengefasst:

```
tpv,, ( ) , '#/+' (1 '#/-' ) 1 '@F+' [2 '&!' ||2, |2 '@F-' ] 2'
```

## Angabe von Satzpositionen `pos` und Dateibereichen `ber`

a) Programmmodus:

`ber:` `pos` | `(pos,pos)` | `(pos#n)` | `([z]/)`

`pos:` `bez` | `bez+n` | `bez-n` | `+n` | `-n`

`bez:` `z` | `[z]/u` | `*`

b) Textmodus:

`ber:` `pos` | `(pos,pos)` | `(pos#n)` | `([s].)` | `([[s.]z]/)`

`pos:` `bez` | `bez+n` | `bez-n` | `+n` | `-n`

`bez:` `[s.]z` | `[[s.]z]/u` | `*`

Dabei bedeutet (im Programm- und Textmodus):

`*` Satz mit der aktuellen Satzposition.

`bez+n` `n`-ter Satz nach der Satzposition `bez`

`bez-n` `n`-ter Satz vor der Satzposition `bez`

`+n` `n`-ter Satz vom Anfang der Datei an

`-n` `n`-ter Satz vom Ende der Datei her

`(pos1,pos2)` Dateibereich, der mit der Satzposition `pos1` beginnt und mit der Satzposition `pos2` endet

`(pos#n)` Dateibereich, der mit der Satzposition `pos` beginnt und insgesamt `n` Sätze umfasst

`z` Satz mit der Zeilennummer `z` und der Unterscheidungsnummer Null

`z/u` Satz mit der Zeilennummer `z` und der Unterscheidungsnummer `u`

`(z/)` Dateibereich, der alle Sätze mit der Zeilennummer `z` umfasst, unabhängig von der Unterscheidungsnummer `u`

`s.z` Satz mit der Seitennummer `s`, der Zeilennummer `z` und der Unterscheidungsnummer Null

`s.z/u` Satz mit der Seitennummer `s`, der Zeilennummer `z` und der Unterscheidungsnummer `u`

`(s.)` Dateibereich, der alle Sätze mit der Seitennummer `s` umfasst, unabhängig von der Zeilennummer `z` und der Unterscheidungsnummer `u`

`(s.z/)` Dateibereich, der alle Sätze mit der Seitennummer `s` und der Zeilennummer `z` umfasst, unabhängig von der Unterscheidungsnummer `u`

Die Angabe `ber` kann weggelassen werden, wenn der zu bearbeitende Bereich alle Sätze der Datei umfasst.



Die in [ ] stehenden Angaben einer Satzposition können weggelassen werden. In diesem Fall wird der weggelassene Wert von der Satzposition \* genommen. Eine Ausnahme bildet die zweite Positionsangabe einer Bereichsangabe `ber` der Form `(pos, pos)`; hier werden die weggelassenen Angaben von der ersten Positionsangabe übernommen, falls sie dort angegeben sind.

Bei der Unterscheidungsnummer müssen führende Nullen geschrieben werden, abschließende Nullen können weggelassen werden (`2/3` ist also gleichbedeutend mit `2/30`).

## Begrenzung auf bestimmte Spalten `begr`

Die Angabe `begr` kann weggelassen werden. Das bedeutet, dass jeweils im ganzen Satz nach einer Zeichenfolge gesucht wird, die unter `stab` angegeben ist, bzw. im ganzen Satz die Zeichenfolgen ausgetauscht werden, die unter `atab` angegeben sind. Soll jeweils nur in bestimmten Spalten gesucht bzw. ausgetauscht werden, so sind für `begr` folgende Angaben möglich:

<code>n1-n2</code>	von Spalte <code>n1</code> bis Spalte <code>n2</code>
<code>n1+n2</code>	<code>n2</code> Spalten weit ab Spalte <code>n1</code>
<code>+n</code>	in den ersten <code>n</code> Spalten
<code>-n</code>	in den letzten <code>n</code> Spalten
<code>n</code>	nur in Spalte <code>n</code>

An Stelle der Angabe `begr` zur Begrenzung auf bestimmte Spalten sind auch folgende Angaben möglich:

- ein Nummernzeichen
- ein oder zwei Unterstreichungsstriche
- ein oder zwei Prozentzeichen

Die Bedeutung ist in den drei folgenden Abschnitten beschrieben.

## Begrenzung auf Satznummer

Wird in einer Editoranweisung für `begr` (bzw. bei der Anweisung für die Colorierung vor dem Doppelpunkt, vom Hexadezimal-Code der Farbangabe durch Komma getrennt) ein Nummernzeichen angegeben, so wird nur in den Satznummern gesucht.

## Nur ganze Wörter suchen/austauschen

Werden in einer Editoranweisung für `begr` (bzw. bei der Anweisung für die Colorierung vor dem Doppelpunkt, vom Hexadezimal-Code der Farbangabe durch Komma getrennt) ein oder zwei Unterstreichungsstriche angegeben, so werden nur ganze Wörter gesucht bzw. ausgetauscht.

Eine Zeichenfolge (sie kann auch Worttrenner enthalten) gilt dabei als ganzes Wort, wenn sie zwischen Worttrennern bzw. am Satzanfang und/oder Satzende steht. Werden zwei Unterstreichungsstriche angegeben, so gelten alle Zeichen außer Buchstaben und Ziffern als Worttrenner; wird nur einer angegeben, so gilt der Unterstreichungsstrich nicht als Worttrenner (z. B. für Variablenamen, die Unterstreichungsstriche enthalten können).

## Ignorieren von Akzent-Codierungen

Wird in einer Editoranweisung für `begr` (bzw. bei der Anweisung für die Colorierung vor dem Doppelpunkt, vom Hexadezimal-Code der Farbangabe durch Komma getrennt) ein Prozentzeichen angegeben, so gelten beim Suchen bzw. Austauschen von Zeichenfolgen folgende zusätzlichen Regelungen:

- Akzent-Codierungen
- die Zeichenfolgen »^«, »\«, »#.«, »#,«, »#'«
- Punktierung-Codes für hebräische Schrift

werden im durchsuchten Text übergangen, falls sie nicht explizit in der Such-Tabelle `stab` bzw. Austausch-Tabelle `atab` angegeben sind.

In der Such-Tabelle bzw. Austausch-Tabelle sind weder in der Suchzeichenfolge noch in der Ersatzzeichenfolge Verweise erlaubt.

Beispiele:

Mit »:e1eve:« wird »élève« (= »%/e1%/eve«) und »élève« (= "%/e1%\eve«) gefunden, mit »:e1%/eve:« nur »élève« (= »%/e1%/eve«).

Mit »:Zloty:« wird »Zloty« und »Złoty« (= »Z#.loty«) gefunden.

## Ignorieren von Auszeichnungen/Schriftumschaltungen

Werden in einer Editoranweisung für `begr` (bzw. bei der Anweisung für die Colorierung vor dem Doppelpunkt, vom Hexadezimal-Code der Farbangabe durch Komma getrennt) zwei Prozentzeichen angegeben, so gelten beim Suchen bzw. Austauschen von Zeichenfolgen folgende zusätzlichen Regelungen:

- Akzent-Codierungen
- die Zeichenfolgen »^«, »\«, »#.«, »#,«, »#'«
- Punktierung-Codes für hebräische Schrift

- das Minuszeichen
- runde und eckige Klammern
- Codierungen der Form »#x+« und »#x-«
- die Codierungen »#h:«, »#t:«, »#o:«, »#u:«, »#g:«
- Tags deren Name aus einem Buchstaben besteht, z. B. <i>...</i>

werden im durchsuchten Text übergangen, falls sie nicht explizit in der Such-Tabelle `stab` bzw. Austausch-Tabelle `atab` angegeben sind.

Werden in Suchzeichenfolgen Umlaute oder das scharfe s (ä, ö, ü, Ä, Ö, Ü und ß) angegeben, so dürfen im durchsuchten Text an ihrer Stelle auch ihre Ersatzzeichen (ae, oe, ue, Ae, AE, Oe, OE, Ue, UE, ss) stehen.

In der Such-Tabelle bzw. Austausch-Tabelle sind weder in der Suchzeichenfolge noch in der Ersatzzeichenfolge Verweise erlaubt.

## Vergleichs-Tabelle `vtab`

Die Angaben zu einer Vergleichs-Tabelle entsprechen denen zu einer Such-Tabelle. Es sind jedoch keine Häufigkeits- und keine Umgebungsbedingungen erlaubt.

Die möglichen Angaben zu `vtab` entsprechen denen einer Vergleichs-Tabelle (Parameterart VIII). Diese ist für die Parameter-Konvention »{ }« ab Seite 713 bzw. und für die Parameter-Konvention »<>« ab Seite 744 beschrieben.

## Such-Tabelle `stab`

Es werden die Zeichenfolgen angegeben, nach denen gesucht werden soll (Suchzeichenfolgen), und ggf. Zeichenfolgen, die beim Suchen übergangen werden sollen (Ausnahmezeichenfolgen). Die einzelnen Sätze werden jeweils von links nach rechts nach den angegebenen Zeichenfolgen durchsucht. Sind verschieden lange Zeichenfolgen angegeben, so haben die jeweils längeren Zeichenfolgen Vorrang; bei gleich langen Zeichenfolgen entspricht die Rangfolge der Reihenfolge ihrer Angabe.

Die Zeichenfolgen werden durch ein frei wählbares Begrenzungszeichen (außer Backslash und Klammern) voneinander getrennt. Das Begrenzungszeichen ist das erste Zeichen der Angabe `stab`. Suchzeichenfolgen und Ausnahmezeichenfolgen werden durch zwei aufeinander folgende Begrenzungszeichen voneinander getrennt. Durch Angabe von zwei aufeinander folgenden Begrenzungszeichen kann beliebig oft zwischen Suchzeichenfolgen und Ausnahmezeichenfolgen gewechselt werden. Das letzte Zeichen der Angabe `stab` muss ebenfalls ein Begrenzungszeichen sein.

Die möglichen Angaben zu `stab` entsprechen denen einer Such-Tabelle (Parameterart IX). Diese ist für die Parameter-Konvention »{ }« ab Seite 714 und für die Parameter-Konvention »<>« ab Seite 746 beschrieben. Nach welcher Konvention die Parameter interpretiert werden, kann mit dem Kommando `#PARAMETER` (siehe Seite 207) eingestellt werden.

## Austausch-Tabelle `atab`

Es werden paarweise Zeichenfolgen angegeben, deren jeweils erste Zeichenfolge die Suchzeichenfolge ist, die durch die jeweils zweite Zeichenfolge, die Ersatzzeichenfolge, ersetzt werden soll. Außerdem können ggf. Zeichenfolgen angegeben werden, die beim Ersetzen übergangen werden sollen (Ausnahmezeichenfolgen).

Für die Suchzeichenfolgen gilt das im Abschnitt »Zeichenfolgen `stab`« Gesagte. Die Buchstaben der Ersatzzeichenfolge werden unverändert als Groß- und Kleinbuchstaben eingesetzt.

Such- und Ersatzzeichenfolgen dürfen verschieden lang sein. Sie werden durch ein frei wählbares Begrenzungszeichen (außer Backslash und Klammern) voneinander getrennt. Das Begrenzungszeichen ist das erste Zeichen der Angabe `atab`. Das Umschalten von Zeichenfolgenpaaren auf Ausnahmezeichenfolgen geschieht dadurch, dass an einer Stelle, an der eine Suchzeichenfolge erwartet wird, nochmals ein Begrenzungszeichen steht. Das letzte Zeichen der Angabe `atab` muss ebenfalls ein Begrenzungszeichen sein.

Die möglichen Angaben zu `atab` entsprechen denen einer Austausch-Tabelle (Parameterart X). Diese ist für die Parameter-Konvention »{ }« ab Seite 721 und für die Parameter-Konvention »<>« ab Seite 752 beschrieben. Nach welcher Konvention die Parameter interpretiert werden, kann mit dem Kommando `#PARAMETER` (siehe Seite 207) eingestellt werden.

## Recherchier-Tabelle `rtab`

Es werden Zeichenfolgen angegeben, die als zulässig gelten sollen. Sie werden durch ein frei wählbares Begrenzungszeichen (außer Backslash und Klammern) voneinander getrennt. Das Begrenzungszeichen ist das erste Zeichen in der Angabe `rtab`. Das letzte Zeichen muss ebenfalls ein Begrenzungszeichen sein.

Das Fragezeichen und der Stern haben eine spezielle Bedeutung. Ein Fragezeichen steht stellvertretend für ein beliebiges TUSTEP-Zeichen und ein Stern für null bis beliebig viele beliebige TUSTEP-Zeichen.

An Stelle eines einzelnen Zeichens einer Zeichenfolge kann auch eine in eckige Klammern eingeschlossene Gruppe von Zeichen angegeben werden. Die Zeichen innerhalb der eckigen Klammern werden als eine Zeichengruppe mit den angegebenen Zeichen interpretiert; sie steht stellvertretend für irgend eines der Zeichen.

Die möglichen Angaben zu `rtab` entsprechen denen einer Recherchier-Tabelle (Parameterart XII). Diese ist für die Parameter-Konvention »{ }« ab Seite 724 und für die Parameter-Konvention »<>« ab Seite 756 beschrieben. Nach welcher Konvention die Parameter interpretiert werden, kann mit dem Kommando `#PARAMETER` (siehe Seite 207) eingestellt werden.

## Anpassen des Editors (Optionen)

Die vom Editor voreingestellten Farben sind so gewählt, dass sich möglichst auf allen Bildschirmen ein brauchbares Bild ergibt. Durch die Verschiedenheit der Bildschirme und deren Einstellungen können sich jedoch auf manchen Bildschirmen ungünstige oder gar unbrauchbare Darstellungen ergeben. Deshalb gibt es die Möglichkeit, die verwendeten Farben einzustellen.

Das Editorfenster ist standardmäßig gleich groß wie das TUSTEP-Fenster. Es gibt jedoch die Möglichkeit, für das Editorfenster eine andere Größe einzustellen.

Für den Cursor gibt es ebenfalls Einstellmöglichkeiten. Für Einfüge- und Überschreibmodus können unterschiedliche Formen des Cursors gewählt werden. Der Cursor kann auch unterdrückt werden (z. B. weil sein Blinken stört). In diesem Fall sollte über die Farbeinstellung dafür gesorgt werden, dass die Cursor-Position sichtbar bleibt.

Diese Einstellmöglichkeiten sind in den drei folgenden Kapiteln beschrieben. Alle diese Einstellungen werden Optionen genannt. Sie gelten jeweils nur so lange, bis die TUSTEP-Sitzung mit dem Kommando `#BEENDE` oder `#NORMIERE` beendet wird. Die aktuellen Einstellungen können jedoch mit der Optionen-Anweisung (s. u.) in codierter Form in eine Datei geschrieben werden. Von dort können sie über die Spezifikation `DEFINITIONEN` beim Aufruf des Editors wieder übernommen werden und gelten dann, bis sie geändert werden bzw. bis zum Ende der TUSTEP-Sitzung. Sollen die Optionen automatisch beim ersten Aufruf des Editors in einer TUSTEP-Sitzung übernommen werden, können sie in das Segment `EDIT` der INI-Datei (siehe Seite 89) eingetragen werden.

Zu den Optionen zählen außer den genannten Einstellungen auch noch, ob Einfüge- oder Überschreibmodus eingestellt ist (vgl. die Steuerbefehle `TGL_INS`, `SET_INS` und `SET_REP` Seite 359).

## Automatisches Einstellen der Optionen

Optionen können beim ersten Aufruf des Editors in einer TUSTEP-Sitzung automatisch übernommen werden. Dazu müssen sie in das Segment `EDIT` der INI-Datei (siehe Seite 89) eingetragen werden:

```
O=000000-000000 373B17-7484F4A437-7484F4A4B4-171B37
```

Die Optionen sind hier unvollständig angegeben; das Beispiel soll nur zeigen, wie die Definition der Optionen etwa aussieht. Dieser Satz wird nicht von Hand eingegeben, sondern mit der folgenden Anweisung in die Datei eingetragen.

- , `pos` Eintragen der aktuell eingestellten Optionen auf die Satzposition `pos`. Falls schon ein Satz mit der Satznummer `pos` vorhanden ist, wird die Anweisung zurückgewiesen.

Werden Einstellungen geändert und sollen die Änderungen auch für nachfolgende TUSTEP-Sitzungen gelten, müssen die Optionen mit der folgenden Anweisung aktualisiert werden. Diese Anweisung muss auch statt der oben angegebenen verwendet werden, wenn der Satz schon existiert, in dem die Optionen eingetragen werden sollen.

- !, `pos` Eintragen der aktuell eingestellten Optionen auf die Satzposition `pos`. Falls schon ein Satz mit der Satznummer `pos` vorhanden ist, wird er überschrieben.

## Einstellen der Farben

Mit der Tastenkombination `Ctrl+F` bzw. `Strg+F` (Taste »F« drücken, während die `Ctrl-` bzw. `Strg-`Taste niedergehalten wird) wird das Einstellen der Farben eingeleitet. Das Editorfenster hat danach etwa folgendes Aussehen:

Menu line:	37	normal									3B	emphasized	17	border
Number field:	74	normal	84	cursor	F4	typed in	A4	emphasized	37	border				
Text field:	74	normal	84	cursor	F4	typed in	A4	emphasized	B4	marked				
Message line:	17	normal					1B	emphasized	37	border				
Command line:	7C	normal	84	cursor	F4	typed in	74	emphasized	B4	marked				
Status line:	37	normal					3B	emphasized	17	border				
Popup window:	74	normal	84	cursor			B4	emphasized	37	border				

01	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
02	12	22	32	42	52	62	72	82	92	A2	B2	C2	D2	E2	F2
03	13	23	33	43	53	63	73	83	93	A3	B3	C3	D3	E3	F3
04	14	24	34	44	54	64	74	84	94	A4	B4	C4	D4	E4	F4
05	15	25	35	45	55	65	75	85	95	A5	B5	C5	D5	E5	F5
06	16	26	36	46	56	66	76	86	96	A6	B6	C6	D6	E6	F6
07	17	27	37	47	57	67	77	87	97	A7	B7	C7	D7	E7	F7
08	18	28	38	48	58	68	78	88	98	A8	B8	C8	D8	E8	F8
09	19	29	39	49	59	69	79	89	99	A9	B9	C9	D9	E9	F9
0A	1A	2A	3A	4A	5A	6A	7A	8A	9A	AA	BA	CA	DA	EA	FA
0B	1B	2B	3B	4B	5B	6B	7B	8B	9B	AB	BB	CB	DB	EB	FB
0C	1C	2C	3C	4C	5C	6C	7C	8C	9C	AC	BC	CC	DC	EC	FC
0D	1D	2D	3D	4D	5D	6D	7D	8D	9D	AD	BD	CD	DD	ED	FD
0E	1E	2E	3E	4E	5E	6E	7E	8E	9E	AE	BE	CE	DE	EE	FE
0F	1F	2F	3F	4F	5F	6F	7F	8F	9F	AF	BF	CF	DF	EF	

Select field with up/down/left/right, type a hexadecimal code, exit with ENTER

Zu Anfang steht der Cursor in der ersten Zeile im ersten Eingabefeld. Im unteren Teil sind 256 Zeichenpaare angegeben, die im günstigsten Fall alle in einer anderen Darstellung (Farbe und Helligkeit der Schrift und des Hintergrundes) erscheinen. Aus ihnen können die gewünschten Darstellungen ausgewählt und im oberen Teil die entsprechenden Zeichenpaare in die Eingabefelder (das sind die Stellen, an denen bereits Zeichenpaare stehen) eingetragen werden.

Mit den Cursor-Tasten kann von einem Eingabefeld zum nächsten oder vorangehenden gesprungen werden. Wird ein Eingabefeld geändert, so wird auch der dahinter stehende Text entsprechend der neuen Einstellung angezeigt. Nachdem alle Änderungen vorgenommen wurden, müssen sie mit der Enter-Taste bestätigt werden. Danach erscheint wieder der vorherige Inhalt des Editorfensters. Die Änderungen sind aber erst nach der nächsten Ausgabe (z. B. nach einer Zeige-Anweisung) wirksam.

Folgende Attribute, die sich auf verschiedene Teile des Editorfensters beziehen, können in den Eingabefeldern gesetzt werden:

Menu line = Darstellung der Menü-Zeile

- normal für normal angezeigte Information
- emphasized für hervorgehoben angezeigte Information
- border für Trennzeichen zwischen Schaltflächen

Number field = Darstellung der Satznummer

- normal für normal angezeigte Satznummer
- cursor für das Zeichen, auf dem der Cursor steht
- typed in für eingetippte/eingefügte Zeichen
- emphasized für hervorgehoben angezeigte Satznummer
- border für Trennlinie zwischen Satznummer und Text

Text field = Darstellung der Daten

- normal für normal angezeigten Text
- cursor für das Zeichen, auf dem der Cursor steht
- typed in für eingetippte/eingefügte Zeichen
- emphasized für hervorgehoben angezeigten Text
- marked für markierten Text

Message line = Darstellung der Meldungen

- normal für Fragen und Rückmeldungen (normal)
- emphasized für Fehlermeldungen (hervorgehoben)
- border für Trennzeichen zwischen Schaltflächen

Command line = Darstellung der Anweisungen

- normal für die Eingabeaufforderung (normal)
- cursor für das Zeichen, auf dem der Cursor steht
- typed in für eingetippte/eingefügte Zeichen
- emphasized für vom Editor angezeigte Zeichen (hervorgehoben)
- marked für markierten Text

Status line = Darstellung der Zustandsmeldungen

- normal für ständige Anzeigen (normal)
- emphasized für warnende Anzeigen (hervorgehoben)
- border für Trennzeichen zwischen Feldern (Schaltflächen)

Popup window = Darstellung der Popup-Fenster

- normal für normal angezeigte Information
- cursor für das Zeichen, auf dem der Cursor steht
- emphasized für hervorgehoben angezeigte Information
- border für den Rahmen des Popup-Fensters

Achtung:

Die Einstellungen müssen unbedingt so gewählt werden, dass in den ersten sieben Zeilen die Wörter »normal«, »cursor«, »emphasized«, »marked« und »border« an allen Stellen im Editorfenster zu sehen sind, an denen sie in der obigen Abbildung auch angegeben sind. Andernfalls kann nicht vernünftig mit dem Editor gearbeitet werden, da dann z. B. nicht alle Daten oder keine Fehlermeldungen im Editorfenster zu sehen sind.



Es kann vorkommen, dass nach `Ctrl+F` bzw. `Strg+F` weder die Kommentare noch die Eingabefelder lesbar sind. In diesem Fall empfiehlt sich folgendes Vorgehen: Es wird im unteren Bereich ein gut lesbares Zeichenpaar ausgewählt und diese beiden Zeichen eingegeben, ohne den Cursor vorher mit den Cursor-Tasten zu bewegen. Danach wird die Eingabe mit der Enter-Taste bestätigt und `Ctrl+F` bzw. `Strg+F` wiederholt.

Hinweis:

Im Textfeld können darüber hinaus bestimmte Zeichenfolgen (z. B. Steuerzeichen) in anderen Farben angezeigt werden. Solche Zeichenfolgen und Farben können mit der Color-Anweisung (siehe Seite 298) festgelegt werden.

## Einstellen der Zeilenlänge und Zeilenzahl

Mit der Tastenkombination `Ctrl+L` bzw. `Strg+L` (Taste »L« drücken, während die `Ctrl-` bzw. `Strg-`Taste niedergehalten wird) wird das Einstellen von Zeilenlänge und Zeilenzahl eingeleitet. Das Editorfenster hat danach etwa folgendes Aussehen:

```

Number of lines (10-120) of the entire window (35):
Number of lines (3-110) of the upper text field:
Number of lines (3-110) of the lower text field:

Number of characters (80-240) of the entire window (80):

Number of characters (40-200) of the left text field:
Number of characters (40-200) of the right text field:

Line width (40-240) for data input in mode P:
Line width (40-240) for data input in mode T:

Width of region for wrapping (0-240) in mode P:
Width of region for wrapping (0-240) in mode T:

Select value to change with cursor up/down
Define value entering the number of lines/characters
Exit with ENTER

```

Die Zeile, in der eine Einstellung vorgenommen werden soll, kann mit den Tasten »Cursor nach oben« bzw. »Cursor nach unten« erreicht werden. Die Änderung einer Einstellung erfolgt durch Eingeben des gewünschten Zahlenwertes. Nachdem alle Änderungen vorgenommen wurden, müssen sie mit der Enter-Taste bestätigt werden. Danach erscheint wieder der vorherige Inhalt des Editorfensters (soweit die neuen Einstellungen dies zulassen).

Folgende Einstellungen können vorgenommen werden:

Number of lines / characters:

Damit kann die Größe des Editor-Fensters eingestellt werden.

Wird für das gesamte Editor-Fenster keine Zeilenzahl bzw. keine Zeichenzahl angegeben, so wird dafür die mit dem Kommando #DEFINIERE (siehe Seite 132) eingestellte Zeilen- bzw. Spaltenzahl verwendet.

Wird das Textfeld horizontal geteilt und sind für das obere und untere Textfeld keine Zeilenzahlen angegeben, so wird das Textfeld in zwei gleich hohe Fenster geteilt. Werden Zeilenzahlen angegeben, so dürfen oberes und unteres Textfeld zusammen maximal 114 Zeilen hoch sein.

Wird das Textfeld vertikal geteilt und sind für das linke und rechte Textfeld keine Zeichenzahlen angegeben, so wird das Textfeld in zwei gleich breite Fenster geteilt. Werden Zeichenzahlen angegeben, so dürfen linkes und rechtes Textfeld zusammen maximal 238 Spalten breit sein.

Achtung: Es dürfen maximal nur so viele Zeichen bzw. Zeilen angegeben werden, wie im TUSTEP-Fenster auch tatsächlich angezeigt werden können. Der Editor erkennt nicht, wenn hier zu große Werte angegeben werden. Die Auswirkungen zu großer Werte sind nicht vorhersehbar; beim Korrigieren muss in diesem Fall mit dem Verlust von Daten gerechnet werden.

Line width for data input:

Damit kann für den Programmmodus und den Textmodus die Anzahl der Zeichen angegeben werden, auf die die einzelnen Zeilen bei der Eingabe begrenzt werden sollen.

Width of region for wrapping:

Damit kann für den Programmmodus und den Textmodus die Anzahl der Zeichen angegeben werden, die für den Fall, dass ein Satz bei der Ausgabe ins Textfeld nicht in eine Zeile passt, vom Ende der Zeile her nach einem Leerzeichen (Wortzwischenraum) durchsucht werden sollen, um dann an diesem Leerzeichen eine Fortsetzungszeile zu beginnen. Wird innerhalb der angegebenen Anzahl von Zeichen kein Leerzeichen gefunden, wird die Zeile möglichst voll geschrieben und dann eine Fortsetzungszeile begonnen. Um anzuzeigen, dass am Zeilenumbruch kein Leerzeichen in den Daten vorhanden ist, wird die Zeile mit dem Zeichen Gravis » ` « abgeschlossen.

## Einstellen von Größe und Geschwindigkeit des Cursors

Mit der Tastenkombination `Ctrl+G` bzw. `Strg+G` (Taste »G« drücken, während die `Ctrl-` bzw. `Strg-`Taste niedergehalten wird) wird das Einstellen des Cursors eingeleitet. Das Editorfenster hat danach etwa folgendes Aussehen:

```

Cursor type for REPLACE mode:  0
Cursor type for INSERT mode:   0
Cursor SPEED:                  9
BEEP frequency (6):
BEEP duration (2):

Select value to change with cursor up/down
Select new value with cursor left/right
Exit with ENTER

```

Die Zeile, in der eine Einstellung vorgenommen werden soll, kann mit den Tasten »Cursor nach oben« bzw. »Cursor nach unten« erreicht werden. Die Änderung einer Einstellung erfolgt mit »Cursor nach rechts« bzw. »Cursor nach links«. Dabei werden die möglichen Einstellungen nacheinander angezeigt. Nachdem alle Änderungen vorgenommen wurden, müssen sie mit der `Enter`-Taste bestätigt werden. Danach erscheint wieder der vorherige Inhalt des Editorfensters.

Folgende Einstellungen sind möglich:

Cursor type:

Damit kann das Aussehen des Cursors für den Überschreibmodus (replace mode) und für den Einfügemodus (insert mode) eingestellt werden. Der Cursor wird in der jeweils eingestellten Form dargestellt.

Cursor SPEED:

Damit kann die Geschwindigkeit des Cursors für Dauerfunktion (z. B. bei anhaltendem Drücken einer Cursor-Taste) eingestellt werden. Eine Null bedeutet, dass keine besondere Geschwindigkeit eingestellt wird; bei anderen Ziffern bedeutet eine höhere Ziffer eine größere Geschwindigkeit.

BEEP frequency / duration:

Damit kann die Höhe und Länge des Warntons eingestellt werden, der ausgegeben wird, wenn ein Steuerbefehl (siehe Seite 350) nicht ausgeführt werden kann.

Einschränkung:

Die Einstellmöglichkeiten sind immer gleich, jedoch haben die Einstellungen unter Linux und macOS nicht immer eine entsprechende Wirkung, wenn das verwendete Terminal-Programm eine Einstellung nicht vorsieht.

## Tastenkombinationen für Funktionsaufrufe

In der folgenden Tabelle ist mit `Shift` die Taste für Großschreibung gemeint; auf deutschen Tastaturen ist die `Ctrl`-Taste mit `Strg` beschriftet.

In der ersten Spalte ist die einzugebende Tastenkombination angegeben, in der zweiten Spalte jeweils der Name der Funktion, die durch die Tastenkombination aufgerufen wird. In der dritten bzw. vierten Spalte ist der Name der Funktion angegeben, die aufgerufen wird, wenn vor der in der ersten Spalte angegebenen Tastenkombination zusätzlich ein- bzw. zweimal die Plus-Taste im Ziffernblock gedrückt wird.

Da auf manchen Notebooks die Plus-Taste nur umständlich zu erreichen oder nicht vorhanden ist, kann statt der Plus-Taste grundsätzlich auch die Tastenkombination `Ctrl+B` bzw. `Strg+B` verwendet werden.

	...	Plus-...	Plus-Plus-...	
F1	F1	F11	F21	
Fn	Fn	F1n	F2n	n = 2 bis 9
F10	F10	F20	F30	
F11	F11	F21	F31	
F12	F12	F22	F32	
Shift+F1	F11	F21	F31	
Shift+Fn	F1n	F2n	F3n	n = 2 bis 9
Shift+F10	F20	F30	F40	
Shift+F11	F21	F31	F41	
Shift+F12	F22	F32	F42	
Ctrl+F1	F21	F31	F41	
Ctrl+Fn	F2n	F3n	F4n	n = 2 bis 9
Ctrl+F10	F30	F40	F50	
Ctrl+F11	F31	F41	F51	
Ctrl+F12	F32	F42	F52	

### Anmerkungen:

Falls für die jeweilige Funktionstaste eine Editoranweisung definiert ist, wird vor der Ausführung der Anweisung automatisch der Steuerbefehl `CONFIRM` (siehe Seite 398) ausgeführt.

Falls für die jeweilige Funktionstaste ein Makroaufruf definiert ist, wird nur dieses Makro aufgerufen und ausgeführt.

Die Definition von Funktionstasten ist auf Seite 287 beschrieben.

## Tastenkombinationen für Makroaufrufe

In der folgenden Tabelle ist mit  $Z_n$  die Ziffer  $n$  im Ziffernblock, mit  $n$  die Ziffer  $n$  oberhalb der Buchstaben-Tasten und mit `Shift` die Taste für Großschreibung gemeint; auf deutschen Tastaturen ist die `Ctrl`-Taste mit `Strg` beschriftet.

In der ersten Spalte ist die einzugebende Tastenkombination angegeben, in der zweiten Spalte jeweils der Name des Makros, das durch die Tastenkombination aufgerufen und ausgeführt wird. In der dritten bzw. vierten Spalte ist der Name des Makros angegeben, das aufgerufen und ausgeführt wird, wenn vor der in der ersten Spalte angegebenen Tastenkombination zusätzlich ein- bzw. zweimal die Plus-Taste im Ziffernblock gedrückt wird.

Da auf manchen Notebooks die Plus-Taste nur umständlich zu erreichen oder nicht vorhanden ist, kann statt der Plus-Taste grundsätzlich auch die Tastenkombination `Ctrl+B` bzw. `Strg+B` verwendet werden.

Abhängig vom Tastatur-Treiber und dessen Einstellungen können die angegebenen Tastenkombinationen auch eine andere Wirkung haben. Dies betrifft unter Linux und macOS insbesondere die Tastenkombinationen mit `Shift+ALT`, `Ctrl+ALT` und `Shift+Ctrl`.

	...	Plus-...	Plus-Plus-...	
a		A	(s. u.)	
x		x	(s. u.)	x = b bis y
z		Z	(s. u.)	
ALT+a	A			
ALT+x	x			x = b bis y
ALT+z	Z			
Shift+ALT+a	SA_A			
Shift+ALT+x	SA_x			x = b bis y
Shift+ALT+z	SA_Z			
Ctrl+ALT+a	CA_A			
Ctrl+ALT+x	CA_x			x = b bis y
Ctrl+ALT+z	CA_Z			
Shift+Ctrl+a	SC_A			
Shift+Ctrl+x	SC_x			x = b bis y
Shift+Ctrl+z	SC_Z			
ALT+0	M_0	M_10	M_20	
ALT+n	M_n	M_1n	M_2n	n = 1 bis 8
ALT+9	M_9	M_19	M_29	
Shift+ALT+0	SA_0			

	...	Plus-...	Plus-Plus-...	
Shift+ALT+n	SA_n			n = 1 bis 8
Shift+ALT+9	SA_9			
Ctrl+ALT+0	CA_0			
Ctrl+ALT+n	CA_n			n = 1 bis 8
Ctrl+ALT+9	CA_9			
Shift+Ctrl+0	SC_0			
Shift+Ctrl+n	SC_n			x = 1 bis 8
Shift+Ctrl+9	SC_9			
Shift+Z0	S_0	S_10	S_20	
Shift+Zn	S_n	S_1n	S_2n	n = 1 bis 8
Shift+Z9	S_9	S_19	S_29	
Ctrl+Z0	C_0	C_10	C_20	
Ctrl+Zn	C_n	C_1n	C_2n	n = 1 bis 8
Ctrl+Z9	C_9	C_19	C_29	
ALT+Z0	A_0	A_10	A_20	
ALT+Zn	A_n	A_1n	A_2n	n = 1 bis 8
ALT+Z9	A_9	A_19	A_29	
NUM+Z0	N_0	N_10	N_20	
NUM+Zn	N_n	N_1n	N_2n	n = 1 bis 8
NUM+Z9	N_9	N_19	N_29	
ALT+F1	F_1	F_11	F_21	
ALT+Fn	F_n	F_1n	F_2n	n = 2 bis 9
ALT+F10	F_10	F_20	F_30	
ALT+F11	F_11	F_21	F_31	
ALT+F12	F_12	F_22	F_32	

Nach zweimaligem Drücken der Plus-Taste im Ziffernblock kann der Name eines beliebigen Makros eingegeben werden. Der Name wird in der Statuszeile angezeigt. Das entsprechende Makro wird ausgeführt, sobald die Eingabe mit der Enter-Taste bestätigt wird.

## Mausaktionen für Makroaufrufe

Es gibt fünf verschiedenen Mausaktionen:

**Click:** Maustaste drücken und wieder loslassen, ohne dazwischen die Maus zu bewegen.

**Press:** Maustaste drücken und anschließend die Maus (bevor die Maustaste wieder losgelassen wird) bewegen.

**Release:** Maustaste loslassen nach **Press**

**Wheel\_up:** Mousrad nach oben drehen

**Wheel\_down:** Mousrad nach unten drehen

Das Mousrad kann nicht nur gedreht, sondern auch wie eine mittlere Maustaste verwendet werden.

Steht bei einer Mausaktion der Mauszeiger im Textfeld oder in der Anweisungszeile, so wird der Cursor an diese Stelle positioniert.

Werden im Editorfenster zwei Textfelder angezeigt, so kann mit der Mausaktion **Click** von einem Textfeld in das andere gewechselt werden. Dabei wird zuvor automatisch der Steuerbefehl **CONFIRM** (siehe Seite 398) ausgeführt, um ggf. im Textfeld vorgenommene Änderungen in die Datei zu übertragen.

In der folgenden Tabelle ist mit **Shift** die Taste für Großschreibung gemeint; auf deutschen Tastaturen ist die **Ctrl**-Taste mit **Strg** beschriftet.

In der ersten Spalte ist eine Mausaktion oder eine Taste und eine Mausaktion, in den restlichen Spalten jeweils der Name des Makros angegeben, das damit aufgerufen und ausgeführt wird.

	Linke Maustaste	Mittlere Maustaste	Rechte Maustaste
Click	M_LC	M_MC	M_RC
Press	M_LP	M_MP	M_RP
Release	M_LR	M_MR	M_RR
Shift+Click	S_LC	S_MC	S_RC
Shift+Press	S_LP	S_MP	S_RP
Shift+Release	S_LR	S_MR	S_RR
Ctrl+Click	C_LC	C_MC	C_RC
Ctrl+Press	C_LP	C_MP	C_RP
Ctrl+Release	C_LR	C_MR	C_RR
ALT+Click	A_LC	A_MC	A_RC
ALT+Press	A_LP	A_MP	A_RP
ALT+Release	A_LR	A_MR	A_RR



Shift+ALT+Click	SA_IC	SA_MC	SA_RC
Shift+ALT+Press	SA_LP	SA_MP	SA_RP
Shift+ALT+Release	SA_LR	SA_MR	SA_RR
Ctrl+ALT+Click	CA_IC	CA_MC	CA_RC
Ctrl+ALT+Press	CA_LP	CA_MP	CA_RP
Ctrl+ALT+Release	CA_LR	CA_MR	CA_RR
Shift+Ctrl+Click	SC_IC	SC_MC	SC_RC
Shift+Ctrl+Press	SC_LP	SC_MP	SC_RP
Shift+Ctrl+Release	SC_LR	SC_MR	SC_RR
Wheel_up		M_UP	
Wheel_down		M_DN	
Shift+Wheel_up		S_UP	
Shift+Wheel_down		S_DN	
Ctrl+Wheel_up		C_UP	
Ctrl+Wheel_down		C_DN	
ALT+Wheel_up		A_UP	
ALT+Wheel_down		A_DN	
Shift+ALT+Wheel_up		SA_UP	
Shift+ALT+Wheel_down		SA_DN	
Ctrl+ALT+Wheel_up		CA_UP	
Ctrl+ALT+Wheel_down		CA_DN	
Shift+Ctrl+Wheel_up		SC_UP	
Shift+Ctrl+Wheel_down		SC_DN	

Einige dieser Makros werden bei Beginn einer TUSTEP-Sitzung automatisch definiert (siehe Seite 289).

Unter Linux und macOS kann es sein, dass das Terminal-Programm so konfiguriert ist, dass Mausaktionen mit der rechten und/oder mittleren Maustaste nicht an TUSTEP weitergegeben werden. Deshalb kann im TUSTEP-Editor ein Klick mit der rechten oder mittleren Maustaste auch simuliert werden. Hierzu muss rechts unten in der Statuszeile des Editors die Schaltfläche »MOUSE« angeklickt werden. Die Anzeige wechselt dadurch auf »MIDDLE«. Dies bedeutet, dass der nächste Klick bzw. das nächste Markieren mit der linken Maustaste wie eine Klick bzw. Markieren mit der mittleren Maustaste gewertet wird. Wird die Schaltfläche »MOUSE« zweimal angeklickt, wechselt die Anzeige erst zu »MIDDLE« und dann zu »RIGHT«. In diesem Fall wirkt der nächste Klick bzw. das nächste Markieren mit der linken Maustaste wie eine Klick bzw. Markieren mit der rechten Maustaste. Wird die Schaltfläche »MOUSE« ein drittes Mal angeklickt, wechselt die Anzeige zu »LEFT«.

## Tastenkombinationen für Steuerbefehle

In den folgenden Tabellen sind mit *oben*, *unten*, *links* und *rechts* die entsprechenden Pfeiltasten (Cursor-Tasten) und mit den Namen *Stern*, *Minus*, *Plus* und *Enter* die entsprechenden Tasten im Ziffernblock gemeint; bei den Tastenkombinationen *Ctrl+Sondertaste* bzw. *Strg+Sondertaste* ist jeweils die entsprechende Sondertaste im mittleren Tastenblock gemeint.

In der ersten Spalte ist die Tastenkombination angegeben, in der zweiten Spalte jeweils der Name des Steuerbefehls, der durch die Tastenkombination ausgeführt wird. In der dritten bzw. vierten Spalte ist der Name des Steuerbefehls angegeben, der ausgeführt wird, wenn vor der in der ersten Spalte angegebenen Tastenkombination zusätzlich ein- bzw. zweimal die Plus-Taste im Ziffernblock gedrückt wird. In der letzten Spalte ist die deutsche Bezeichnung der in der ersten Spalte angegebenen Tastenkombination angegeben.

Da auf manchen Notebooks die Plus-Taste nur umständlich zu erreichen oder nicht vorhanden ist, kann statt der Plus-Taste grundsätzlich auch die Tastenkombination *Ctrl+B* bzw. *Strg+B* verwendet werden.

	...	Plus-...	Plus-Plus-...	
Esc	CANCEL			Esc
Tab	TAB			Tabulator
Backspace	BSP	HOME	CLEAR	Rücktaste
Return	CR	SPLIT	JOIN	Return
oben	CUR_UP			oben
unten	CUR_DN			unten
links	CUR_LE			links
rechts	CUR_RI			rechts
Ins	TGL_INS			Einfüg
Del	DEL	DEL_REC	UND_REC	Entf
Home	SKP_BEG	DEL_BEG	UND_BEG	Pos1
End	SKP_END	DEL_END	UND_END	Ende
PgUp	SHW_UP	DEL_LINE	UND_LINE	Bild rauf
PgDn	SHW_DN	INS_LINE	DUP_LINE	Bild runter
Minus	SKP_WORD	DEL_WORD	UND_WORD	Minus
Stern	HOME	CLEAR	RESHOW	Stern
Enter	ENTER	TGL_INS	MRK_INS	Enter
Shift+Backspace	DEL	DEL_REC	UND_REC	Shift+Rückt.
Shift+Return	ENTER	TGL_INS	MRK_INS	Shift+Return
ALT+links	EXCH_CHAR_LE			
ALT+rechts	EXCH_CHAR_RI			
ALT+oben	EXCH_REC_UP			
ALT+unten	EXCH_REC_DN			
Ctrl+Return	LF			Strg+Return
Ctrl+oben	JMP_UP			Strg+oben
Ctrl+unten	JMP_DN			Strg+unten
Ctrl+links	SKP_LE	DEL_LE	UND_LE	Strg+links
Ctrl+rechts	SKP_RI	DEL_RI	UND_RI	Strg+rechts
Ctrl+Ins	MRK_INS			Strg+Einfüg
Ctrl+Del	MRK_DEL_REP			Strg+Entf
Ctrl+Home	MRK_INI			Strg+Pos1
Ctrl+End	MRK_REP			Strg+Ende
Ctrl+Enter	MRK_INS			Strg+Enter
Ctrl+A	END_CMD_LINE			Strg+A
Ctrl+B	(B)			Strg+B
Ctrl+C	MRK_CB			Strg+C
Ctrl+D	CANCEL			Strg+D
Ctrl+E	ENTER	TGL_INS	MRK_INS	Strg+E
Ctrl+F	(F)			Strg+F
Ctrl+G	(G)			Strg+G
Ctrl+H	BSP	HOME	CLEAR	Strg+H

	...	Plus-...	Plus-Plus-...	
Ctrl+I	TAB			Strg+I
Ctrl+J	LF			Strg+J
Ctrl+K	(K)			Strg+K
Ctrl+L	(L)			Strg+L
Ctrl+M	CR	SPLIT	JOIN	Strg+M
Ctrl+N	JMP_REC_NR			Strg+N
Ctrl+O	HELP			Strg+O
Ctrl+P	PRINT			Strg+P
Ctrl+Q	(Q)			Strg+Q
Ctrl+R	RESHOW			Strg+R
Ctrl+S	(S)			Strg+S
Ctrl+T	EXCH_CHAR			Strg+T
Ctrl+U	KEY_TST			Strg+U
Ctrl+V	INSERT_CB			Strg+V
Ctrl+W	REFRESH			Strg+W
Ctrl+X	MRK_DEL_CB			Strg+X
Ctrl+Y	CHG_SETTINGS			Strg+Y
Ctrl+Z	UND_CHAR			Strg+Z

(B) Gleiche Wirkung wie die Plus-Taste im Ziffernblock

(F) Farben einstellen, siehe Seite 335

(G) Größen einstellen, siehe Seite 339

(L) Längen einstellen, siehe Seite 337

(Q) Linux: Bildschirmausgabe fortsetzen

(S) Linux: Bildschirmausgabe anhalten

(K) Eingabehilfe und Markieren von Text:

Die Tastenkombination Ctrl+K-x bzw. Strg+K-x, wobei x ein Sonderzeichen oder eine Ziffer ist, dient als Eingabehilfe für Such- und Austausch-Tabellen (siehe Seite 269).

Außerdem gibt es noch folgende Tastenkombinationen, mit denen Texte ohne Maus markiert werden können:

Ctrl+K-Return	MRK_INI	Strg+K-Return
Ctrl+K-Backspace	MRK_REP	Strg+K-Rückt.
Ctrl+K-Home	MRK_BEG	Strg+K-Pos1
Ctrl+K-Minus	MRK_REP	Strg+K-Minus
Ctrl+K-End	MRK_END	Strg+K-Ende
Ctrl+K-Enter	MRK_INS	Strg+K-Enter
Ctrl+K-Del-Return	MRK_IGN	Strg+K-Entf-Return
Ctrl+K-Del-Backspace	MRK_DEL_REP	Strg+K-Entf-Rückt.
Ctrl+K-Del-Home	MRK_DEL_BEG	Strg+K-Entf-Pos1
Ctrl+K-Del-Minus	MRK_DEL_REP	Strg+K-Entf-Minus
Ctrl+K-Del-End	MRK_DEL_END	Strg+K-Entf-Ende
Ctrl+K-Del-Enter	MRK_DEL_INS	Strg+K-Entf-Enter
Ctrl+K-Del-Del	MRK_DEL_DEL	Strg+K-Entf-Entf

## Steuerbefehle im Editor

Um die Arbeit mit dem Editor zu erleichtern, gibt es Steuerbefehle. Jeder dieser Steuerbefehle hat einen Kurznamen, mit dem er in Editormakros angesprochen und ausgeführt werden kann. Einige Steuerbefehle können auch direkt mit Tastenkombinationen ausgeführt werden.

Sämtliche Steuerbefehle beziehen sich lediglich auf den Inhalt des Textfeldes bzw. der Anweisungszeile. Die damit veranlassten Änderungen im Textfeld werden nicht sofort in die Datei übertragen (siehe »Änderungen im Textfeld« Seite 271).

### Erläuterungen

Leerzeichen am Zeilenende werden nicht in die Datei übertragen.

Mit »Text« (»text«) wird in dieser Beschreibung der Text in der aktuellen Zeile (d. h. die Zeile, in der der Cursor steht) bezeichnet. Der Teil, der am Zeilenanfang für die Satznummer vorgesehen ist, gehört nicht zum »Text«.

Als »Wort« (»word«) gelten Zeichenfolgen, die durch Leerzeichen, »Text«anfang oder »Text«ende begrenzt sind; Leerzeichen (auch mehrere unmittelbar aufeinander folgende) gehören jeweils zum davor stehenden »Wort«.

## Tastenkombinationen für Steuerbefehle

Jeder Steuerbefehl hat einen Kurznamen. Am Ende der Beschreibung der einzelnen Steuerbefehle ist nach »T:« angegeben, mit welcher Tastenkombination der jeweilige Steuerbefehl ausgeführt werden kann. Alternative Tastenkombinationen sind durch Komma getrennt angegeben. Mit den Namen `Stern`, `Minus`, `Plus` und `Enter` sind die entsprechenden Tasten im Ziffernblock gemeint; bei den Tastenkombinationen `Ctrl+Sondertaste` bzw. `Strg+Sondertaste` ist jeweils die entsprechende Sondertaste im mittleren Tastenblock gemeint.

Da auf manchen Notebooks die Plus-Taste nur umständlich zu erreichen oder nicht vorhanden ist, kann statt der Plus-Taste grundsätzlich auch die Tastenkombination `Ctrl+B` bzw. `Strg+B` verwendet werden.

Bei Tastenkombinationen, die mit `Ctrl` (z. B. `Ctrl+A`) beginnen, muss

- die Taste `Ctrl` bzw. `Strg` gedrückt und niedergehalten werden,
- die nach dem Pluszeichen angegebene Taste (aus dem mittleren Block mit den Cursor-Tasten bzw. der angegebene Buchstabe) gedrückt und wieder losgelassen werden,
- die Taste `Ctrl` bzw. `Strg` wieder losgelassen werden,
- ggf. (nach `Ctrl+K` bzw. `Strg+K`) die Tasten für den Rest der Tastenkombination einzeln gedrückt und wieder losgelassen werden.

Bei allen anderen Tastenkombinationen müssen die Tasten in der angegebenen Reihenfolge einzeln gedrückt und wieder losgelassen werden.

## Cursor

`CUR_UP`

Cursor up

Cursor springt um eine Zeile nach oben

Steht der Cursor in der obersten Zeile, springt er in die Anweisungszeile; falls jedoch der `SCROLL`-Modus aktiv ist, wird der angezeigten Text wie mit `SCR_UP` (siehe Seite 358) um eine Zeile nach unten gescrollt; wurden im Textfeld Änderungen vorgenommen, so werden diese zuvor in die Datei übertragen.

T: Pfeil nach oben

Falls die Shift-Taste niedergehalten und die Pfeil-Taste gedrückt wird und außerdem das Markieren noch nicht (mit dem Steuerbefehl `MRK_INI`, siehe Seite 365) initialisiert ist, wird zuerst das Markieren initialisiert und dann erst der Cursor an die neue Position bewegt.

`CUR_DN`

Cursor down

Cursor springt um eine Zeile nach unten

Steht der Cursor in der untersten Zeile, springt er in die Anweisungszeile; falls jedoch der `SCROLL`-Modus aktiv ist, wird der angezeigten Text wie mit `SCR_DN` (siehe Seite 358) um eine Zeile nach

oben gescrollt; wurden im Textfeld Änderungen vorgenommen, so werden diese zuvor in die Datei übertragen.

T: Pfeil nach unten

Falls die Shift-Taste niedergehalten und die Pfeil-Taste gedrückt wird und außerdem das Markieren noch nicht (mit dem Steuerbefehl `MRK_INI`, siehe Seite 365) initialisiert ist, wird zuerst das Markieren initialisiert und dann erst der Cursor an die neue Position bewegt.

`CUR_RI`

Cursor right

Cursor springt um ein Zeichen nach rechts

Steht der Cursor nach einem oder mehreren Leerzeichen hinter dem »Text«, springt er an den Anfang des »Textes« in der nächsten Zeile.

T: Pfeil nach rechts

Falls die Shift-Taste niedergehalten und die Pfeil-Taste gedrückt wird und außerdem das Markieren noch nicht (mit dem Steuerbefehl `MRK_INI`, siehe Seite 365) initialisiert ist, wird zuerst das Markieren initialisiert und dann erst der Cursor an die neue Position bewegt.

`CUR_LE`

Cursor left

Cursor springt um ein Zeichen nach links

Steht der Cursor am Anfang des »Textes«, springt er ans Ende des »Textes« in der vorangehenden Zeile.

T: Pfeil nach links

Falls die Shift-Taste niedergehalten und die Pfeil-Taste gedrückt wird und außerdem das Markieren noch nicht (mit dem Steuerbefehl `MRK_INI`, siehe Seite 365) initialisiert ist, wird zuerst das Markieren initialisiert und dann erst der Cursor an die neue Position bewegt.

`CMD_LINE`

Jump to command line

Wie `BEG_CMD_LINE`; siehe auch `CLR_CMD_LINE` Seite 364.

`BEG_CMD_LINE`

Jump to start of command line

Cursor springt an den Anfang des Eingabebereichs der Anweisungszeile.

`END_CMD_LINE`

Jump to end of text in command line

Cursor springt ans Ende des Textes in der Anweisungszeile.

T: `Ctrl+A`



HOME	Home / Jump to first line of text Wie FIRST_LINE
FIRST_LINE	Jump to first line of text Cursor springt an den Anfang des »Textes« in der obersten Zeile im Textfeld. T: Stern, Plus-Rücktaste, Plus-Backspace
LF	Line feed: Skip to next start of text Cursor springt in die nächste Zeile an den Anfang des »Textes«. T: Ctrl+Return, Ctrl+J
LAST_LINE	Skip to last line of text – bei Dateneingabe (z. B. nach der Anweisung ee): Cursor springt in der letzten Zeile mit Text ans Ende des Textes. Falls kein Text vorhanden ist, springt der Cursor an den Anfang der oberste Zeile. – sonst: (noch) nicht definiert
NEXT_LINE	Skip to next line after text – bei Dateneingabe (z. B. nach der Anweisung ee): Cursor springt an den Anfang der (leeren) Zeile, die auf die letzte Zeile mit Text folgt. Falls kein Text vorhanden ist, springt der Cursor in die oberste Zeile. Falls nach dem Text keine leere Zeile mehr folgt, wird die oberste Zeile in die Datei übertragen, die restlichen Zeilen werden um eine Zeile nach oben verschoben und der Cursor springt an den Anfang der untersten Zeile. – sonst: (noch) nicht definiert
BEG_REC	Skip to start of record – bei Dateneingabe (z. B. nach der Anweisung ee): Cursor springt an den Anfang der Zeile. – sonst: Cursor springt an den Anfang des Satzes (= Zeile mit Satznummer), in dem er steht.
END_REC	Skip to end of record – bei Dateneingabe (z. B. nach der Anweisung ee): Cursor springt ans Ende der Zeile. – sonst: Cursor springt ans Ende des Satzes, in dem er steht.
PREV_REC	Skip to start of previous record – bei Dateneingabe (z. B. nach der Anweisung ee): Cursor springt an den Anfang der vorangehenden Zeile bzw. in die Anweisungszeile, falls der Cursor in der obersten Zeile steht. – sonst: Cursor springt an den Anfang des vorangehenden Satzes

- (= Zeile mit Satznummer) bzw. in die Anweisungszeile, falls der Cursor im ersten angezeigten Satz steht.
- NEXT\_REC** Skip to start of next record
- bei Dateneingabe (z. B. nach der Anweisung `ee`): Cursor springt an den Anfang der nachfolgenden Zeile bzw. in die Anweisungszeile, falls der Cursor in der untersten Zeile steht.
  - sonst: Cursor springt an den Anfang des nächsten Satzes (= Zeile mit Satznummer) bzw. in die Anweisungszeile, falls der Cursor im letzten angezeigten Satz steht.
- TAB** Skip to next tabulator
- im `SPLIT`-Modus und `ENTER`-Modus (vgl. `CHG_SETTINGS` Seite 395):
- Cursor springt auf die nächstfolgende Tabulatorposition.
- Hinweis: Tabulatorpositionen können mit der Anweisung `tab` (siehe Seite 286) definiert werden.
- im `MACRO`-Modus (vgl. `CHG_SETTINGS` Seite 395):
- `TAB` führt das Makro mit dem Namen `TAB` aus. Falls das Makro `TAB` nicht definiert ist, hat `TAB` die gleiche Wirkung wie in den Modi `SPLIT` und `ENTER`.
- Falls eine Makroleiste (siehe Seite 291) angezeigt wird, wird das Makro `nameTAB` aufgerufen, wobei `name` der Name der Makroleiste ist. Falls das Makro `nameTAB` nicht definiert ist, hat `TAB` die gleiche Wirkung, wie wenn keine Makroleiste angezeigt wird.
- T: Tab, Ctrl+I
- INDENT** Indent record
- Cursor springt an den Anfang des Satzes und rückt den Text um ebensoviele Leerstellen ein, wie der Text des im Textfeld davor stehenden Satzes eingerückt ist.
- SKP\_BEG** Skip to start of text
- Cursor springt an den Anfang des »Textes« in der aktuellen Zeile; wird dieser Steuerbefehl mehrfach unmittelbar hintereinander ausgeführt, springt er bei jeder folgenden Ausführung jeweils an den Anfang des »Textes« in der vorhergehenden Zeile.
- T: Pos1, Home
- Falls die Shift-Taste niedergehalten und die genannte Taste gedrückt wird und außerdem das Markieren noch nicht (mit dem Steuerbefehl `MRK_INI`, siehe Seite 365) initialisiert ist, wird zuerst das Markieren initialisiert und dann erst der Cursor an die neue Position bewegt.

- SKP\_END** Skip to end of text
- Cursor springt an das Ende des »Textes« in der aktuellen Zeile; wird dieser Steuerbefehl mehrfach unmittelbar hintereinander ausgeführt, springt er bei jeder folgenden Ausführung jeweils an das Ende des »Textes« in der nächsten Zeile.
- T: Ende, End
- Falls die Shift-Taste niedergehalten und die genannte Taste gedrückt wird und außerdem das Markieren noch nicht (mit dem Steuerbefehl `MRK_INI`, siehe Seite 365) initialisiert ist, wird zuerst das Markieren initialisiert und dann erst der Cursor an die neue Position bewegt.
- SKP\_WORD** Skip to next word
- Wie `SKP_RI`
- SKP\_RI** Skip to next word right
- Cursor springt an den Anfang des nächsten »Wortes«. Steht der Cursor im letzten »Wort« einer Zeile, so springt er hinter dieses »Wort« auf die Position, an der das nächste »Wort« beginnen kann; steht der Cursor schon hinter dem letzten »Wort«, springt er an den Anfang des ersten »Wortes« in der nächsten Zeile.
- T: Minus, `Ctrl`+Pfeil nach rechts
- SKP\_LE** Skip to preceding word left
- Cursor springt an den Anfang des »Wortes«, in dem er steht. Steht der Cursor am Anfang eines »Wortes«, springt er an den Anfang des vorhergehenden »Wortes«; steht der Cursor am Anfang des ersten »Wortes« einer Zeile oder davor, springt er hinter das letzte »Wort« in der vorhergehenden Zeile auf die Position, an der das nächste »Wort« beginnen kann.
- T: `Ctrl`+Pfeil nach links
- FND\_BLANK** Find next blank
- Sucht im Textfeld nach einem Leerzeichen. Am Zeilenende wird jeweils ein Leerzeichen angenommen; Zeilen ohne Text werden bei der Suche übergangen. Die Suche beginnt mit dem Zeichen nach der Cursor-Position; steht der Cursor in der Anweisungszeile, beginnt die Suche in der ersten Zeile des Textfeldes. Der Cursor wird auf das Leerzeichen positioniert; wird kein Leerzeichen gefunden, wird der Cursor in die Anweisungszeile positioniert.
- Soll nicht vorwärts, sondern rückwärts gesucht werden, so muss vor `FND_BLANK` der Steuerbefehl `REVERSE` angegeben werden. Die Suche beginnt in diesem Fall mit dem Zeichen vor der Cursor-Position.

- FND\_BEG** Find start of word
- Der Cursor springt auf das erste Zeichen eines Wortes. Steht der Cursor in einem Wort, so springt er auf das erste Zeichen dieses Wortes, andernfalls auf das erste Zeichen des nächsten Wortes.
- Als Wort gilt bei diesem Steuerbefehl eine Zeichenfolge, die aus Buchstaben (einschließlich Akzent- und Sonderbuchstaben), Ziffern und Backslash besteht. Andere Zeichen gelten als nicht zum Wort gehörend.
- FND\_END** Find end of word
- Der Cursor springt hinter das letzte Zeichen eines Wortes. Steht der Cursor in einem Wort, so springt er hinter das letzte Zeichen dieses Wortes, andernfalls hinter das letzte Zeichen des vorangehenden Wortes.
- Als Wort gilt bei diesem Steuerbefehl eine Zeichenfolge, die aus Buchstaben (einschließlich Akzent- und Sonderbuchstaben), Ziffern und Backslash besteht. Andere Zeichen gelten als nicht zum Wort gehörend.
- JMP\_REC\_NR** Jump to record number
- Steht der Cursor im Textfeld, springt er in der gleichen Zeile in das Feld mit den Satznummern; steht er in der Anweisungszeile, springt an den Anfang des Satzes mit der aktuellen Satzposition, falls dieser im Textfeld mit Satznummer angezeigt wird.
- T: `Ctrl+N`
- JMP\_DN** Jump to next emphasized field
- Cursor springt an den Anfang der nächsten hervorgehobenen Zeichenfolge (z. B. nach der Zeige-Anweisung `zn`). Falls von der aktuellen Cursor-Position bis zum Ende des Textfeldes keine solche vorhanden ist, springt der Cursor in die Anweisungszeile.
- T: `Ctrl+Pfeil nach unten`
- JMP\_DN:xx** Jump to next `xx` colored field
- Cursor springt an den Anfang der nächsten Zeichenfolge, die in der Farbe `xx` angezeigt wird.
- Für `xx` sind die gleichen Angaben vorgesehen wie für den Steuerbefehl `MRK_CHG:xx` (siehe Seite 368).
- JMP\_UP** Jump to preceding emphasized field
- Cursor springt ans Ende der vorhergehenden hervorgehobenen Zeichenfolge (z. B. nach der Zeige-Anweisung `zn`). Falls von der aktuellen Cursor-Position bis zum Anfang des Textfeldes keine solche vorhanden ist, springt der Cursor in die Anweisungszeile.
- T: `Ctrl+Pfeil nach oben`

JMP_UP : xx	<p>Jump to preceding xx colored field</p> <p>Cursor springt ans Ende der vorhergehenden Zeichenfolge, die in der Farbe xx angezeigt wird.</p> <p>Für xx sind die gleichen Angaben vorgesehen wie für den Steuerbefehl MRK_CHG : xx (siehe Seite 368).</p>
SAVE_CUR	<p>Save cursor position</p> <p>Merkt die aktuelle Cursor-Position im Editorfenster.</p>
REST_CUR	<p>Restore cursor position</p> <p>Setzt den Cursor auf die zuletzt mit SAVE_CUR oder EXCH_CUR gemerkte Position.</p>
EXCH_CUR	<p>Exchange cursor position</p> <p>Merkt die aktuelle Cursor-Position im Editorfenster und setzt gleichzeitig den Cursor auf die zuletzt mit SAVE_CUR oder EXCH_CUR gemerkte Position.</p>
CUR_POS : n ; m	<p>Set cursor to position n ; m</p> <p>Setzt den Cursor in der n-ten Zeile des Textfeldes auf die m-te Spalte des Textes. Ist für n Null angegeben, so wird die Zeilenposition des Cursors beibehalten und nur die Spaltenposition geändert; ist für m Null angegeben, so wird nur die Zeilenposition des Cursors geändert und die Spaltenposition beibehalten. Ist für n und m Null angegeben, so wird geprüft, ob im Textfeld die Satznummern angezeigt werden und sich der Cursor im Textfeld innerhalb des angezeigten Textes befindet; falls dies nicht der Fall ist wird eine entsprechende Fehlermeldung angezeigt.</p>

## Blättern, Scrollen

SHW_DN	<p>Show next screen of text</p> <p>Blättert vorwärts weiter, beginnend mit dem Satz, in dem der Cursor steht; steht der Cursor in der Anweisungszeile: beginnend mit dem Satz, der dem letzten im Textfeld stehenden folgt; bei leerem Textfeld: beginnend mit dem Satz, der auf die aktuelle Satzposition folgt.</p> <p>Wurden im Textfeld Änderungen vorgenommen, so werden diese zuvor in die Datei übertragen.</p> <p>T: PgDn</p>
SHW_UP	<p>Show preceding screen of text</p> <p>Blättert rückwärts weiter, beginnend mit dem Satz, in dem der Cursor steht; steht der Cursor in der Anweisungszeile: beginnend mit dem Satz, der dem ersten im Textfeld stehenden vorangeht; bei lee-</p>

rem Textfeld: beginnend mit dem Satz, der der aktuellen Satzposition vorangeht.

Wurden im Textfeld Änderungen vorgenommen, so werden diese zuvor in die Datei übertragen.

T: PgUp

SCR\_UP Scroll up one line

Scrollt den angezeigten Text um eine Zeile nach unten; falls danach oben im Textfeld genügend Raum frei ist, wird zusätzlich der dem ersten angezeigten Satz vorangehende Satz angezeigt.

Wurden im Textfeld Änderungen vorgenommen, so werden diese zuvor in die Datei übertragen.

Maus: Wheel\_up = Makro M\_UP (Voreinstellung)

SCR\_DN Scroll down one line

Scrollt den angezeigten Text um eine Zeile nach oben; falls danach unten im Textfeld genügend Raum frei ist, wird zusätzlich der dem letzten angezeigten Satz nachfolgende Satz angezeigt.

Wurden im Textfeld Änderungen vorgenommen, so werden diese zuvor in die Datei übertragen.

Maus: Wheel\_down = Makro M\_DN (Voreinstellung)

SCR\_UP\_BOTH Scroll up one line in both text fields

Scrollt bei geteiltem Textfeld den angezeigten Text in beiden Teilen um eine Zeile nach unten; falls danach oben im Textfeld genügend Raum frei ist, wird zusätzlich der dem ersten angezeigten Satz vorangehende Satz angezeigt.

Wurden im Textfeld Änderungen vorgenommen, so werden diese zuvor in die Datei übertragen.

Maus: Shift+Wheel\_up = Makro S\_UP (Voreinstellung)

SCR\_DN\_BOTH Scroll down one line in both text fields

Scrollt bei geteiltem Textfeld den angezeigten Text in beiden Teilen um eine Zeile nach oben; falls danach unten im Textfeld genügend Raum frei ist, wird zusätzlich der dem letzten angezeigten Satz nachfolgende Satz angezeigt.

Wurden im Textfeld Änderungen vorgenommen, so werden diese zuvor in die Datei übertragen.

Maus: Shift+Wheel\_down = Makro S\_DN (Voreinstellung)

## Einfügen, Überschreiben

SET_INS	<p>Set insert mode</p> <p>Schaltet den Einfügemodus ein. Im Einfügemodus werden die neu eingegebenen Zeichen an der aktuellen Cursor-Position eingeschoben, d. h. alle Zeichen von der Cursor-Position bis zum Zeilenende werden nach rechts verschoben. Ist am Zeilenende kein Platz mehr vorhanden, werden die nicht mehr in die Zeile passenden Zeichen wortweise in eine Folgezeile verschoben.</p>
SET_REP	<p>Set replace mode</p> <p>Schaltet den Überschreibmodus ein. Im Überschreibmodus werden bereits vorhandene Zeichen mit den neu eingegebenen überschrieben.</p>
TGL_INS	<p>Toggle insert mode / replace mode</p> <p>Schaltet vom Einfügemodus (vgl. SET_INS) in den Überschreibmodus (vgl. SET_REP) und umgekehrt.</p> <p>Der aktuell eingestellte Modus wird in der Statuszeile unten rechts mit »INSERT« bzw. »REPLACE« angezeigt. Wird mit der linken Maustaste auf diese Anzeige geklickt, so wechselt der Editor vom Einfügemodus in den Überschreibmodus bzw. umgekehrt.</p> <p>T: Einfg, Ins, Plus-Enter</p>
INS:text	<p>Insert text</p> <p>Fügt die für <code>text</code> angegebene Zeichenfolge so vor der aktuellen Cursor-Position ein, als wäre die Zeichenfolge dort mit der Tastatur im Einfügemodus eingegeben worden.</p> <p>Die für <code>text</code> angegebene Zeichenfolge muss mit einem frei wählbaren Begrenzungszeichen (Sonderzeichen außer Komma und Leerzeichen) eingeleitet und abgeschlossen werden.</p>
REP:text	<p>Replace text</p> <p>Schreibt die für <code>text</code> angegebene Zeichenfolge so ab der aktuellen Cursor-Position, als wäre die Zeichenfolge dort mit der Tastatur im Überschreibmodus eingegeben worden.</p> <p>Die für <code>text</code> angegebene Zeichenfolge muss mit einem frei wählbaren Begrenzungszeichen (Sonderzeichen außer Komma und Leerzeichen) eingeleitet und abgeschlossen werden.</p>
SELECT_TEXT	<p>Select text</p> <p>Falls Text markiert ist, wird der markierte Text in die Liste der gemerkten Textteile aufgenommen; ist kein Text markiert, so wird der Anfang aller gemerkten Textteile angezeigt.</p> <p>Soll einer dieser Textteile an der aktuellen Cursor-Position eingefügt werden, so muss dieser mit Pfeil nach unten/oben ausgewählt</p>

und die Auswahl mit der Return-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der linken Maustaste angeklickt werden.

Die Anzeige kann durch Drücken der Leertaste oder Escape-Taste wieder gelöscht werden.

Um die Liste zu reduzieren, kann mit der Delete-Taste der jeweils markierte Textteil aus der Liste entfernt werden. Wird die Anzeige anschließend mit der Escape-Taste gelöscht, so werden alle mit der Delete-Taste entfernten Textteile wieder in die Liste aufgenommen.

T: ALT+X (Voreinstellung)

DUP\_LINE Duplicate line

Die aktuelle Zeile wird verdoppelt. Dabei werden alle auf die aktuelle Zeile folgenden Zeilen um eine Zeile nach unten verschoben. Der »Text« wird in die freigewordene Zeile kopiert. Falls möglich, wird die neue Zeile automatisch mit einer noch freien Satznummer versehen.

T: Plus-Plus-PgDn

INS\_LINE Insert line

Die aktuelle Zeile und alle folgenden Zeilen werden um eine Zeile nach unten verschoben. Dadurch entsteht eine Leerzeile, in die anschließend Text eingetragen werden kann. Falls möglich, wird die neue Zeile automatisch mit einer noch freien Satznummer versehen.

T: Plus-PgDn

INS\_LINE\_IND Insert line and indent

Wie INS\_LINE; zusätzlich wird der Cursor um ebensoviele Leerstellen eingerückt, wie der Text des im Textfeld davor stehenden Satzes eingerückt ist.

T: ALT+N (Voreinstellung)

SPLIT Split line

Die aktuelle Zeile wird an der Cursor-Position aufgeteilt. Dabei werden alle auf die aktuelle Zeile folgenden Zeilen um eine Zeile nach unten verschoben. Der Teil des »Textes« links vom Cursor bleibt in der aktuellen Zeile; der Rest des »Textes« wird in die freigewordene Zeile verschoben. Steht der Cursor auf einem Leerzeichen, so wird dieses nicht in die folgende Zeile verschoben. Falls möglich, wird die neue Zeile automatisch mit einer noch freien Satznummer versehen.

T: Plus-Return



TRY\_SPLIT:name Try split line

Wie SPLIT; falls die Zeile nicht aufgeteilt werden kann oder falls keine Satznummer mehr frei ist, wird das Editormakro mit dem angegebenen Namen aufgerufen und ausgeführt.

## Löschen und Einfügen

DEL Delete: Delete character in line

Löscht das Zeichen auf der aktuellen Cursor-Position und verschiebt alle in der Zeile folgenden Zeichen um eine Position nach links.

Ist im Editorfenster Text markiert, so wird der gesamte markierte Text gelöscht, ohne ihn im Editor-Zwischenspeicher zu merken (entspricht in diesem Fall MRK\_DEL\_DEL, siehe Seite 366).

T: Entf, Del, Shift+Rücktaste, Shift+Backspace

DEL\_BLANK Delete if blank

Wie DEL, falls das Zeichen auf der aktuellen Cursor-Position ein Leerzeichen ist.

BSP Backspace: Delete character in line

Löscht das Zeichen links von der aktuellen Cursor-Position und verschiebt alle in der Zeile folgenden Zeichen um eine Position nach links. Der Cursor rückt ebenfalls um eine Position nach links.

Steht der Cursor am Anfang einer Fortsetzungszeile, springt er an das Ende des »Textes« in der vorangehenden Zeile; steht er am Anfang eines Satzes, so wird dieser Satz mit dem im Textfeld unmittelbar vorangehenden Satz zusammengehängt.

T: Rücktaste, Backspace, Ctrl+H

BSP\_BLANK Backspace if blank

Wie BSP, falls das Zeichen links von der aktuellen Cursor-Position ein Leerzeichen ist.

UND\_CHAR Undo (= Insert characters deleted with) DEL or BSP

Fügt die zuletzt mit DEL und/oder BSP gelöschte Zeichenfolgen vor der aktuellen Cursor-Position ein. Alle in der Zeile folgenden Zeichen werden nach rechts verschoben. Der Cursor wird mitverschoben und steht danach hinter den eingefügten Zeichen.

T: Ctrl+Z

RECALL\_CHAR Recall deleted characters

Zeigt die mit DEL und/oder BSP gelöschten Zeichenfolgen an.

Soll eine dieser Zeichenfolge wieder zurückgeholt werden, so dass sie mit UND\_CHAR wieder eingefügt werden kann, so muss diese mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Return-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der linken Maustaste angeklickt werden.

Soll eine dieser Zeichenfolge nur vollständig angezeigt werden, so muss diese mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Enter-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der rechten Maustaste angeklickt werden.

Die Anzeige kann durch Drücken der Leertaste oder Escape-Taste wieder gelöscht werden.

Um die Liste zu reduzieren, kann mit der Delete-Taste die jeweils markierte Zeichenfolge aus der Liste entfernt werden. Wird die Anzeige anschließend mit der Escape-Taste gelöscht, so werden alle mit der Delete-Taste entfernten Zeichenfolgen wieder in die Liste aufgenommen.

T: ALT+Z (Voreinstellung)

DEL\_BEG Delete to start of text

Löscht alle Zeichen links von der aktuellen Cursor-Position bis zum »Text«anfang. Alle in der Zeile folgenden Zeichen werden um die entsprechende Anzahl von Zeichen nach links verschoben. Der Cursor steht danach auf dem ersten Zeichen des »Textes«.

T: Plus-Pos1, Plus-Home

UND\_BEG Undo (= Insert text deleted with) DEL\_BEG

Fügt die zuletzt mit DEL\_BEG gelöschten Zeichen vor der aktuellen Cursor-Position ein. Alle in der Zeile folgenden Zeichen werden nach rechts verschoben. Der Cursor wird mitverschoben und steht danach hinter den eingefügten Zeichen.

T: Plus-Plus-Pos1, Plus-Plus-Home

DEL\_END Delete to end of text

Löscht alle Zeichen von der aktuellen Cursor-Position bis zum »Text«ende.

T: Plus-Ende, Plus-End

UND\_END Undo (= Insert text deleted with) DEL\_END

Fügt die zuletzt mit DEL\_END gelöschten Zeichen und ein Leerzeichen vor der aktuellen Cursor-Position ein. Alle in der Zeile folgenden Zeichen werden nach rechts verschoben. Der Cursor wird nicht verschoben und steht danach am Anfang der eingefügten Zeichen.

T: Plus-Plus-Ende, Plus-Plus-End

DEL_WORD	<p>Delete to next word / end of text</p> <p>Wie DEL_RI (s. u.)</p>
UND_WORD	<p>Undo (= Insert text deleted with) DEL_WORD</p> <p>Wie UND_RI (s. u.)</p>
DEL_RI	<p>Delete (right) to next word / end of text</p> <p>Löscht den Rest des »Wortes« (einschließlich der unmittelbar dahinter stehenden Leerzeichen) rechts der aktuellen Cursor-Position (das ganze »Wort«, wenn der Cursor am »Wort«anfang steht). Alle in der Zeile folgenden Zeichen werden um die entsprechende Anzahl von Zeichen nach links verschoben.</p> <p>T: Plus-Minus, Plus-Ctrl+Pfeil nach rechts</p>
UND_RI	<p>Undo (= Insert text deleted with) DEL_RI</p> <p>Fügt die zuletzt mit DEL_WORD oder DEL_RI gelöschten Zeichen vor der aktuellen Cursor-Position ein. Alle in der Zeile folgenden Zeichen werden nach rechts verschoben. Der Cursor wird nicht verschoben und steht danach auf dem ersten der eingefügten Zeichen.</p> <p>T: Plus-Plus-Minus, Plus-Plus-Ctrl+Pfeil nach rechts</p>
DEL_LE	<p>Delete (left) to start of word / start of text</p> <p>Löscht den Rest des »Wortes« links von der aktuellen Cursor-Position (das ganze vorangehende »Wort«, wenn der Cursor am »Wort«anfang steht). Alle in der Zeile folgenden Zeichen werden um die entsprechende Anzahl von Zeichen nach links verschoben.</p> <p>T: Plus-Ctrl+Pfeil nach links</p>
UND_LE	<p>Undo (= Insert text deleted with) DEL_LE</p> <p>Fügt die zuletzt mit DEL_LE gelöschten Zeichen vor der aktuellen Cursor-Position ein. Alle in der Zeile folgenden Zeichen werden nach rechts verschoben. Der Cursor wird mit verschoben und steht danach auf dem Zeichen, vor dem die Zeichen eingefügt wurden.</p> <p>T: Plus-Plus-Ctrl+Pfeil nach links</p>
TXT_DEL:xx	<p>Set color of text to delete to xx</p> <p>Setzt die Farbe des ggf. zu löschenden Textes. Ein in der angegebenen Farbe angezeigter Text wird gelöscht, wenn der Cursor in diesem Text steht und ein beliebiges Zeichen eingegeben wird oder wenn ein Zeichen von diesem Text mit DEL oder BSP gelöscht wird.</p> <p>Für xx sind die gleichen Angaben vorgesehen wie für den Steuerbefehl MRK_CHG:xx (siehe Seite 368).</p>

- DEL\_LINE** Delete line
- Löscht die aktuelle Zeile (nur im Editorfenster, falls damit ein vollständiger Satz einschließlich der Satznummer gelöscht wird) und verschiebt alle folgenden Zeilen um eine Zeile nach oben.
- Hinweis: Wird ein vollständiger Satz samt der dazugehörenden Satznummer im Editorfenster gelöscht, so bewirkt dies kein Löschen (und keine Änderung) dieses Satzes in der Datei (vgl. Beispiel auf Seite 276). Soll der ganze Satz in der Datei gelöscht werden, kann dafür DEL\_REC (siehe Seite 364) verwendet werden.
- T: Plus-PgUp
- UND\_LINE** Undo (= Insert line deleted with) DEL\_LINE
- Die aktuelle Zeile und alle folgenden Zeilen werden um eine Zeile nach unten verschoben. In die entstehende Leerzeile wird der »Text« der zuletzt mit DEL\_LINE gelöschten Zeile eingefügt. Falls möglich, wird die neue Zeile automatisch mit einer noch freien Satznummer versehen.
- T: Plus-Plus-PgUp
- DEL\_REC** Delete record
- Löscht den Satz, in dem der Cursor steht.
- T: Plus-Entf, Plus-Del
- DEL\_REC\_NEXT** Delete record, skip to next record
- Wie DEL\_REC, jedoch wird zusätzlich der Cursor an den Anfang des nächsten Satzes positioniert.
- T: Shift+Ctrl+Backspace
- UND\_REC** Undo (= Insert record deleted with) DEL\_REC
- Die aktuelle Zeile und alle folgenden Zeilen werden um eine dem Bedarf entsprechend große Anzahl von Zeilen nach unten verschoben. In die entstehenden Leerzeilen wird der »Text« des zuletzt mit DEL\_REC gelöschten Satzes eingefügt. Falls möglich, wird der neue Satz automatisch mit einer noch freien Satznummer versehen.
- T: Plus-Plus-Entf, Plus-Plus-Del
- CLR\_LINE** Clear line
- Löscht den Text der aktuellen Zeile (Satznummer bleibt ggf. erhalten).
- CLR\_CMD\_LINE** Clear command line
- Cursor springt an den Anfang des Eingabebereichs in der Anweisungszeile; der Text im Eingabebereich wird gelöscht.

**CLEAR** Clear screen

Löscht das Textfeld und die Anweisungszeile. Dies ist sinnvoll, wenn Änderungen im Editorfenster unberücksichtigt bleiben sollen oder wenn eine Anweisung gegeben werden soll, die nicht in die Anweisungszeile passt.

T: Ctrl+Backspace, Plus-Stern, Plus-Plus-Backspace

## Markieren, Kopieren, Löschen, Einfügen, Suchen

**MRK\_INI** Initialize marking of text

Definiert die Anfangsposition der Markierung.

Hinweis: Nach MRK\_INI muss der Cursor auf die Endposition für die Markierung positioniert werden und die Markierung mit einem der folgenden Steuerbefehle abgeschlossen werden. Dazwischen kann der Fensterinhalt nicht verändert werden. Wird irgendetwas unternommen, das den Inhalt ändert (z. B. Weiterblättern), wird automatisch ein MRK\_IGN ausgeführt.

T: Ctrl+Home, Strg+Pos1

Maus: Left press = Makro M\_LP (Voreinstellung)  
 Maus: Right press = Makro M\_RP (Voreinstellung)  
 Maus: Shift+Left press = Makro S\_LP (Voreinstellung)  
 Maus: Shift+Right press = Makro S\_RP (Voreinstellung)

**MRK\_CON** Continue marking of text

Wie MRK\_INI, falls noch keine Anfangsposition der Markierung definiert ist; andernfalls ohne Wirkung.

**MRK\_IGN** Ignore marking of text

Hebt eine mit MRK\_INI begonnene Markierung auf.

T: Ctrl+K-Del-Return, Strg+K-Entf-Return

**MRK\_REP** Copy marked text to buffer replacing the contents

Kopiert den markierten Text in den Editor-Zwischenspeicher. Falls der Editor-Zwischenspeicher schon Text enthält, wird dieser vorher gelöscht.

T: Ctrl+End, Strg+Ende, Ctrl+K-Minus

**MRK\_BEG** Insert marked text at beginning of buffer

Kopiert den markierten Text in den Editor-Zwischenspeicher. Falls der Editor-Zwischenspeicher schon Text enthält, wird der markierte Text vor diesen in den Editor-Zwischenspeicher eingeschoben.

T: Ctrl+K-Home, Strg+K-Pos1

- 
- MRK\_END** Append marked text at end of buffer  
Kopiert den markierten Text in den Editor-Zwischenspeicher. Falls der Editor-Zwischenspeicher schon Text enthält, wird der markierte Text an diesen im Editor-Zwischenspeicher angehängt.  
T: Ctrl+K-End, Strg+K-Ende
- MRK\_INS** Insert buffer contents into the text  
Fügt den im Editor-Zwischenspeicher gemerkten Text vor der aktuellen Cursor-Position ein. Alle in der Zeile folgenden Zeichen werden nach rechts verschoben. Der Cursor wird mitverschoben und steht danach hinter den eingefügten Zeichen.  
T: Ctrl+Ins, Strg+Einf, Ctrl+Enter  
Maus: Right click = Makro M\_RC (Voreinstellung)
- MRK\_DEL\_REP** Delete marked text, copy it to the buffer  
Wie MRK\_REP, jedoch wird zusätzlich der markierte Text im Editorfenster gelöscht.  
T: Ctrl+Del, Strg+Entf,
- MRK\_DEL\_BEG** Delete marked text, insert it at beginning of buffer  
Wie MRK\_BEG, jedoch wird zusätzlich der markierte Text im Editorfenster gelöscht.  
T: Ctrl+K-Del-Home, Strg+K-Entf-Pos1
- MRK\_DEL\_END** Delete marked text, append it at end of buffer  
Wie MRK\_END, jedoch wird zusätzlich der markierte Text im Editorfenster gelöscht.  
T: Ctrl+K-Del-End, Strg+K-Entf-Ende,
- MRK\_DEL\_INS** Replace marked text with buffer contents  
Löscht den markierten Text, ohne ihn vorher im Editor-Zwischenspeicher zu merken, und fügt den im Editor-Zwischenspeicher gemerkten Text an dieser Stelle ein.  
T: Ctrl+K-Del-Enter, Strg+K-Entf-Enter  
Maus: Right release = Makro M\_RR (Voreinstellung)
- MRK\_DEL\_DEL** Delete marked text without saving it  
Löscht den markierten Text, ohne ihn im Editor-Zwischenspeicher zu merken.  
T: Ctrl+K-Del-Del, Strg+K-Entf-Entf

- MRK\_FND** Find next occurrence of marked text
- Sucht im Textfeld nach einem Text, der mit dem markierten Text übereinstimmt. Die Suche beginnt mit dem Zeichen nach der Cursor-Position; steht der Cursor in der Anweisungszeile, beginnt die Suche in der ersten Zeile des Textfeldes. Der Cursor wird an den Anfang des gefundenen Textes positioniert; wird kein entsprechender Text gefunden, wird der Cursor in die Anweisungszeile positioniert. Groß- und Kleinbuchstaben werden bei der Suche nicht unterschieden.
- Falls kein Text markiert ist, so wird nach dem gleichen Text wie beim letzten MRK\_FND oder MRK\_DEL\_FND gesucht, bzw. nach dem Text gesucht, der zuletzt mit SET\_FND definiert wurde.
- Soll nicht vorwärts, sondern rückwärts gesucht werden, so muss vor MRK\_FND der Steuerbefehl REVERSE angegeben werden. Die Suche beginnt in diesem Fall mit dem Zeichen vor der Cursor-Position.
- MRK\_DEL\_FND** Delete marked text and find next occurrence
- Wie MRK\_FND, jedoch wird zuvor der markierte Text im Editorfenster gelöscht.
- SET\_FND:text** Define text for MRK\_FND
- Definiert die für `text` angegebene Zeichenfolge zum Suchen mit MRK\_FND ohne vorangehende Markierung eines Textes.
- Die für `text` angegebene Zeichenfolge muss mit einem frei wählbaren Begrenzungszeichen (Sonderzeichen außer Komma und Leerzeichen) eingeleitet und abgeschlossen werden.
- MRK\_TAG\_DEL** Mark tag and delete it
- Markiert das Tag, in dem der Cursor steht, kopiert das Tag in den Editor-Zwischenspeicher und löscht dann das Tag im Editorfenster. Falls der Editor-Zwischenspeicher schon Text enthält, wird dieser vorher gelöscht.
- Maus: `Middle click` = Makro `M_MC` (Voreinstellung)
- MRK\_ASK** Ask what to do with marked text
- Fragt nach, was mit dem markierten Text geschehen soll. Dazu wird in der Statuszeile eine Leiste mit neun Feldern angezeigt. Als Antwort kann eines dieser Felder mit der linken Maustaste angeklickt werden; bei jeder anderen Eingabe wird die Markierung aufgehoben und die Eingabe interpretiert.
- In der folgenden Tabelle ist für die neun Felder jeweils links die Feldbezeichnung und rechts der Steuerbefehl angegeben, der ausgeführt wird, wenn das entsprechende Feld angeklickt wird.

```

" FIND " : MRK_FND
"< COPY " : MRK_BEG
" COPY " : MRK_REP
" COPY ">" : MRK_END
" DELETE " : MRK_DEL_DEL
"< MOVE " : MRK_DEL_BEG
" MOVE " : MRK_DEL_REP
" MOVE ">" : MRK_DEL_END
" IGNORE " : MRK_IGN

```

Maus: Left release = Makro M\_LR (Voreinstellung)

MRK\_CHG:xx Change the color for marking of text to xx

Wechselt während des Markierens (nach MRK\_INI) die Farbe, in der der Text markiert wird. Für xx sind folgende Angaben vorgesehen:

```

xx Farbe mit dem Hexadezimal-Code xx
NFN Farbe für »Number field: normal«
NFC Farbe für »Number field: cursor«
NFT Farbe für »Number field: typed in«
NFE Farbe für »Number field: emphasized«
NFB Farbe für »Number field: border«
TFN Farbe für »Text field: normal«
TFC Farbe für »Text field: cursor«
TFT Farbe für »Text field: typed in«
TFE Farbe für »Text field: emphasized«
TFM Farbe für »Text field: marked«
MLN Farbe für »Message line: normal«
MLE Farbe für »Message line: emphasized«
MLB Farbe für »Message line: border«
CLN Farbe für »Command line: normal«
CLC Farbe für »Command line: cursor«
CLT Farbe für »Command line: typed in«
CLE Farbe für »Command line: emphasized«
CLM Farbe für »Command line: marked«
SLN Farbe für »Status line: normal«
SLE Farbe für »Status line: emphasized«
SLB Farbe für »Status line: border«
PWN Farbe für »Popup Window: normal«

```



PWC Farbe für »Popup Window: cursor«

PWE Farbe für »Popup Window: emphasized«

PWB Farbe für »Popup Window: border«

Die möglichen Hexadezimal-Codes und die jeweils für die anderen Angaben aktuell eingestellten Farben können der Tabelle entnommen werden, die durch die Tastenkombinationen `Ctrl+F` bzw. `Strg+F` angezeigt wird.

MRK\_MRK Leave the marked text marked

Wie `MRK_IGN` (siehe Seite 365), jedoch bleibt die Markierung im Editorfenster erhalten.

MRK\_TST:stab Test marked text

Prüft, ob der markierte Text eine in der Such-Tabelle `stab` (siehe Seite 331) angegebene Zeichenfolge enthält. Die Tabelle darf maximal 250 Zeichen lang sein und muss mit einem frei wählbaren Sonderzeichen, das in der Tabelle selbst nicht vorkommt, eingeleitet und abgeschlossen werden.

Hinweis: Ob der markierte Text eine entsprechende Zeichenfolge enthält, kann danach mit den unter »Verzweigen in andere Makros« ab Seite 375 beschriebenen Steuerbefehlen `IF_MATCH`, `NO_MATCH` und `DO_MATCH` abgefragt werden.

MRK\_STR:stab Mark string

Markiert im Textfeld eine in der Such-Tabelle `stab` (siehe Seite 331) angegebene Zeichenfolge. Es wird die längste Zeichenfolge markiert, die innerhalb der gleichen Zeile wie der Cursor steht und das Zeichen unter dem Cursor enthält. Gibt es mehrere gleich lange Zeichenfolgen, die diese Bedingung erfüllen, so wird die im Textfeld weiter rechts stehende markiert. Die Tabelle darf maximal 250 Zeichen lang sein und muss mit einem frei wählbaren Sonderzeichen, das in der Tabelle selbst nicht vorkommt, eingeleitet und abgeschlossen werden.

Hinweis: Die Markierung muss danach mit einem der oben beschriebenen Steuerbefehle abgeschlossen werden.

Hinweis: Ob der Text eine entsprechende Zeichenfolge enthält, kann danach mit den unter »Verzweigen in andere Makros« weiter unten beschriebenen Steuerbefehlen `IF_MATCH`, `NO_MATCH` und `DO_MATCH` abgefragt werden.

TEST\_MRK:stab Test buffer contents

Prüft, ob der Editor-Zwischenspeicher eine in der Such-Tabelle `stab` (siehe Seite 331) angegebene Zeichenfolge enthält. Die Tabelle darf maximal 250 Zeichen lang sein und muss mit einem frei wählbaren Sonderzeichen, das in der Tabelle selbst nicht vorkommt, eingeleitet und abgeschlossen werden.

Hinweis: Ob der Editor-Zwischenspeicher eine entsprechende Zeichenfolge enthält, kann danach mit den unter »Verzweigen in andere Makros« ab Seite 375 beschriebenen Steuerbefehlen `IF_MATCH`, `NO_MATCH` und `DO_MATCH` abgefragt werden.

`CLR_MRK` Clear buffer

Löscht den Inhalt des Editor-Zwischenspeichers.

`SET_MRK:text` Copy text to buffer replacing contents

Kopiert die für `text` angegebene Zeichenfolge in den Editor-Zwischenspeicher. Falls der Editor-Zwischenspeicher schon Text enthält, wird dieser vorher gelöscht.

Die für `text` angegebene Zeichenfolge muss mit einem frei wählbaren Begrenzungszeichen (Sonderzeichen außer Komma und Leerzeichen) eingeleitet und abgeschlossen werden.

`ADD_MRK:text` Add text to buffer

Fügt die für `text` angegebene Zeichenfolge am Anfang des im Editor-Zwischenspeicher enthaltenen Textes ein.

Die für `text` angegebene Zeichenfolge muss mit einem frei wählbaren Begrenzungszeichen (Sonderzeichen außer Komma und Leerzeichen) eingeleitet und abgeschlossen werden.

`APP_MRK:text` Append text to buffer

Kopiert die für `text` angegebene Zeichenfolge ans Ende des im Editor-Zwischenspeicher enthaltenen Textes.

Die für `text` angegebene Zeichenfolge muss mit einem frei wählbaren Begrenzungszeichen (Sonderzeichen außer Komma und Leerzeichen) eingeleitet und abgeschlossen werden.

`ENTER_MRK:text` Enter text for buffer

Zeigt die für `text` angegebene Zeichenfolge in der Anweisungszeile an und wartet, bis die Eingabe einer Zeichenfolge mit der Return- oder Enter-Taste abgeschlossen wird. Die eingegebene Zeichenfolge wird in den Editor-Zwischenspeicher kopiert. Falls der Editor-Zwischenspeicher schon Text enthält, wird dieser vorher gelöscht.

Bevor dieser Steuerbefehl aufgerufen wird, muss der Editor-Zwischenspeicher (z. B. mit `SET_MRK`, siehe Seite 370) mit einer Zeichenfolge belegt werden. Diese Zeichenfolge dient als Standardwert für die Eingabe, falls ohne Text einzugeben nur die Return- oder Enter-Taste gedrückt wird. Wird die Eingabe mit der Escape-Taste abgeschlossen, wird eine leere Zeichenfolge in den Editor-Zwischenspeicher kopiert.

Die für `text` angegebene Zeichenfolge muss mit einem frei wählbaren Begrenzungszeichen (Sonderzeichen außer Komma und Leerzeichen) eingeleitet und abgeschlossen werden.

- FILE\_MRK** Copy file name to buffer  
Kopiert den Dateinamen in den Editor-Zwischenspeicher, der bei Hole- und Rette-Anweisungen eingesetzt wird, falls er dort weggelassen wird. (vgl. »Namen abfragen/einstellen/löschen« Seite 280)
- SEGMENT\_MRK** Copy segment name to buffer  
Kopiert den Segmentnamen in den Editor-Zwischenspeicher, der bei Hole- und Rette-Anweisungen eingesetzt wird, falls er dort weggelassen wird. (vgl. »Namen abfragen/einstellen/löschen« Seite 280)
- COUNTER\_MRK** Copy counter to buffer  
Kopiert die Anzahl der Zeichenfolgen in den Editor-Zwischenspeicher, die bei der zuletzt ausgeführten erweiterten Zeige-Anweisung gefunden wurden.
- SAVE\_MRK** Save buffer  
Merkt den Inhalt des Editor-Zwischenspeichers.
- REST\_MRK** Restore buffer  
Ersetzt den Inhalt des Editor-Zwischenspeichers mit dem zuletzt mit **SAVE\_MRK** oder **EXCH\_MRK** gemerkten Inhalt.
- EXCH\_MRK** Exchange buffer  
Merkt den Inhalt des Editor-Zwischenspeichers und ersetzt ihn gleichzeitig mit dem zuletzt mit **SAVE\_MRK** oder **EXCH\_MRK** gemerkten Inhalt.
- X\_MRK:atab** Exchange strings within the buffer  
Tauscht im Editor-Zwischenspeicher die in der Austausch-Tabelle **atab** (siehe Seite 332) angegebenen Zeichenfolgen aus. Die Tabelle darf maximal 250 Zeichen lang sein und muss mit einem frei wählbaren Sonderzeichen, das in der Tabelle selbst nicht vorkommt, eingeleitet und abgeschlossen werden.
- DEC\_MRK\_HEX** Decode hexadecimal to decimal within the buffer  
Konvertiert die Hexadezimalzahl im Editor-Zwischenspeicher (andere Daten dürfen nicht enthalten sein) in eine Dezimalzahl.
- ENC\_MRK\_HEX** Encode decimal to hexadecimal within the buffer  
Konvertiert die Dezimalzahl (ohne Vorzeichen) im Editor-Zwischenspeicher (andere Daten dürfen nicht enthalten sein) in eine Hexadezimalzahl.

RECALL\_MRK Recall buffer

Zeigt die früheren Inhalte des Editor-Zwischenspeichers an.

Soll ein früherer Inhalt wieder in den Editor-Zwischenspeicher geholt werden, so muss dieser mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Return-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der linken Maustaste angeklickt werden.

Soll ein früherer Inhalt nur vollständig angezeigt werden, so muss dieser mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Enter-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der rechten Maustaste angeklickt werden.

Die Anzeige kann durch Drücken der Leertaste oder Escape-Taste wieder gelöscht werden.

Um die Liste zu reduzieren, kann mit der Delete-Taste die jeweils markierte Zeichenfolge aus der Liste entfernt werden. Wird die Anzeige anschließend mit der Escape-Taste gelöscht, so werden alle mit der Delete-Taste entfernten Zeichenfolgen wieder in die Liste aufgenommen.

T: ALT+M (Voreinstellung)

## Wiederherstellen gelöschter/geänderter Sätze

RECALL\_DEL Recall deleted records

Zeigt die Sätze, die durch Löschen im Textfeld (z. B. mit DEL\_REC oder DEL\_REC\_NEXT, siehe Seite 364) in der Editor-Datei gelöscht wurden, mit dem Inhalt an, den sie zuvor hatten.

Soll ein Satz wieder hergestellt werden, so muss dieser mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Return-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der linken Maustaste angeklickt werden.

Achtung: Die Sätze werden mit der Satznummer angezeigt, die sie beim Löschen hatten. Falls sich die Satznummern in der Datei inzwischen geändert wurden, sollte der Satz ggf. nur angezeigt (s. u.) werden, um dann die Satznummer entsprechend ändern zu können.

Soll ein Satz nur vollständig angezeigt werden, so muss dieser mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Enter-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der rechten Maustaste angeklickt werden.

Die Anzeige kann durch Drücken der Leertaste oder Escape-Taste wieder gelöscht werden.

Um die Liste zu reduzieren, kann mit der Delete-Taste der jeweils

markierte Satz aus der Liste entfernt werden. Wird die Anzeige anschließend mit der Escape-Taste gelöscht, so werden alle mit der Delete-Taste entfernten Sätze wieder in die Liste aufgenommen.

T: ALT+L (Voreinstellung)

RECALL\_MOD Recall modified records

Zeigt die Sätze, die durch Ändern im Textfeld in der Editor-Datei geändert wurden, mit dem Inhalt an, den sie vor der Änderung hatten.

Soll ein Satz wieder hergestellt werden, so muss dieser mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Return-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der linken Maustaste angeklickt werden.

Achtung: Die Sätze werden mit der Satznummer angezeigt, die sie beim Ändern hatten. Falls sich die Satznummer in der Datei inzwischen geändert hat, sollte der Satz ggf. nur angezeigt (s. u.) werden, um dann die Satznummer entsprechend ändern zu können.

Soll ein Satz nur vollständig angezeigt werden, so muss dieser mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Enter-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der rechten Maustaste angeklickt werden.

Die Anzeige kann durch Drücken der Leertaste oder Escape-Taste wieder gelöscht werden.

Um die Liste zu reduzieren, kann mit der Delete-Taste der jeweils markierte Satz aus der Liste entfernt werden. Wird die Anzeige anschließend mit der Escape-Taste gelöscht, so werden alle mit der Delete-Taste entfernten Sätze wieder in die Liste aufgenommen.

T: ALT+K (Voreinstellung)

## Lesezeichen

Im TUSTEP-Editor wird ein Lesezeichen nicht auf eine bestimmte Stelle im Text, sondern auf eine Satznummer gesetzt. Werden die Sätze mit der Anweisung `u` umnummeriert oder umgestellt, bleiben die Lesezeichen erhalten und werden den entsprechenden neuen Satznummern zugeordnet. Bei Korrigieren des Textes gilt jedoch folgendes:

- Wird eine Satznummer gelöscht, geht das Lesezeichen verloren, auch wenn der Text des dazugehörigen Satzes nicht gelöscht wird.
- Wird ein Text eines Satzes auf eine andere Satznummer verschoben, wird das Lesezeichen nicht mitverschoben, sondern bleibt bei der jeweiligen Satznummer.

---

DEFINE_BM	<p>Define bookmark</p> <p>Steht der Cursor in einem Satz, wird auf diesen Satz ein Lesezeichen gesetzt; steht er in der Anweisungszeile, wird auf die aktuelle Satzposition ein Lesezeichen gesetzt.</p> <p>T: ALT+B (Voreinstellung)</p> <p>Soll ein Lesezeichen mit einem Namen definiert werden, so kann in der Anweisungszeile die Anweisung »*=name« eingegeben und mit der Return-Taste abgeschickt werden; damit wird auf die aktuellen Satzposition ein Lesezeichen mit dem angegebenen Namen gesetzt.</p>
PREV_BM	<p>Go to previous bookmark</p> <p>Zeigt den Satz mit Umgebung an, der als nächster vor (oberhalb) der aktuellen Satzposition ein Lesezeichen hat.</p> <p>T: ALT+O (Voreinstellung)</p>
NEXT_BM	<p>Go to next bookmark</p> <p>Zeigt den Satz mit Umgebung an, der als nächster nach (unterhalb) der aktuellen Satzposition ein Lesezeichen hat.</p> <p>T: ALT+U (Voreinstellung)</p>
SELECT_BM	<p>Select bookmark</p> <p>Zeigt den Anfang aller Sätze mit Lesezeichen an; die Reihenfolge entspricht der Reihenfolge in der Datei.</p> <p>Soll einer der Sätze mit Umgebung angezeigt werden, so muss dieser mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Return-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der linken Maustaste angeklickt werden.</p> <p>T: ALT+I (Voreinstellung)</p>
PREV_VBM	<p>Go to previous visited bookmark</p> <p>Zeigt den Satz mit Umgebung an, dessen Lesezeichen als nächstes zuvor (früher) verwendet wurde.</p> <p>T: ALT+F (Voreinstellung)</p>
NEXT_VBM	<p>Go to next visited bookmark</p> <p>Zeigt den Satz mit Umgebung an, dessen Lesezeichen als nächstes danach (später) verwendet wurde (nur unmittelbar nach dem Steuerbefehl PREV_VBM möglich).</p> <p>T: ALT+S (Voreinstellung)</p>

`SELECT_VBM` Select visited bookmark

Zeigt den Anfang aller Sätze mit Lesezeichen an; die Reihenfolge entspricht der Reihenfolge in der die Lesezeichen verwendet wurden.

Soll einer der Sätze mit Umgebung angezeigt werden, so muss dieser mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Return-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der linken Maustaste angeklickt werden.

T: ALT+D (Voreinstellung)

## Verzweigen in andere Makros

`DO:name` Do macro

Ruft das Editormakro mit dem angegebenen Namen auf und führt es aus. Falls dabei kein Fehler auftritt, werden danach die nach diesen Steuerbefehl folgenden Steuerbefehle ausgeführt.

`SWITCH:name` Switch to macro

Ruft das Editormakro mit dem angegebenen Namen auf und führt es aus.

Hinweise:

Dieser Steuerbefehl ist nur als letzter in einem Makro sinnvoll; nachfolgende Steuerbefehle würden nie ausgeführt.

Sollen mehrere Makros (mit verschiedenen Namen) die gleiche Leistung erbringen, genügt es, ein einziges Makro ausführlich mit den entsprechenden Makroanweisungen zu definieren und in den anderen Makros nur diesen Steuerbefehl mit dem Namen des ausführlich definierten Makros anzugeben.

`SWITCH:erg?name` Display macro bar and switch to a macro

Zeigt die temporäre bzw. imaginäre Makroleiste `name` an und wartet auf genau eine Eingabe. Aus der im `SWITCH`-Befehl angegebenen Namensergänzung `erg` und der unmittelbar nächsten Eingabe wird ein Name zusammengesetzt und das Makro mit diesem Namen aufgerufen. Ist kein Makro mit diesem Namen definiert, wird ein Signalton (`BEEP`) ausgegeben; dann werden die nach diesen Steuerbefehl folgenden Makroanweisungen ausgeführt.

Weitere Einzelheiten und die Definition von Makroleisten sind ab Seite 291 beschrieben.

`SWITCH:erg?` Wait for input and switch to a macro

Wie `SWITCH:erg?name`, jedoch ohne Anzeige einer Makroleiste. Es wird wie bei der Anzeige einer imaginären Makroleiste auf genau eine Eingabe mit der Tastatur gewartet und dann ein Makro aufgerufen.

Weitere Einzelheiten sind ab Seite 291 beschrieben.

`RETRY` Retry `SWITCH`

Springt innerhalb eines Makros zum vorangehenden Steuerbefehl `SWITCH:erg?name` bzw. `SWITCH:erg?` zurück und führt das Makro ab dieser Stelle erneut aus.

Dieser Steuerbefehl besitzt einen internen Zähler. Er kann durch eine vorangehende Häufigkeitsangabe (z. B. `10*RETRY`) auf einen bestimmten Anfangswert gesetzt werden; fehlt die Häufigkeitsangabe, hat er den Anfangswert 1. Bei jeder Ausführung des Steuerbefehls wird zuerst geprüft, ob der Zähler 0 ist. Wenn er nicht 0 ist, wird er um 1 erniedrigt, und dann wird zum vorangehenden `SWITCH`-Befehl zurückgesprungen; wenn er 0 ist, wird nicht mehr zurückgesprungen, sondern es werden die nachfolgenden Makroanweisungen ausgeführt.

Hinweis: Die Makroanweisungen hinter `SWITCH:erg?name` bzw. `SWITCH:erg?` werden nur ausgeführt, wenn durch die Eingabe kein Name eines definierten Makros entstanden ist, wenn also eine nicht vorgesehene Eingabe erfolgte. Zwischen `SWITCH` und `RETRY` können dann z. B. mit `DISPLAY` (siehe Seite 384) Hinweise ausgegeben werden, um dann vielleicht eine vorgesehene Eingabe, für die ein Makro definiert ist, zu erhalten. Durch die Angabe `n*RETRY` kann mit `n` die maximale Anzahl der Fehlversuche festgelegt werden; erst nach diesen `n` Fehlversuchen werden die hinter `RETRY` folgenden Makroanweisungen ausgeführt.

`CHAR` Write character

Schreibt das Zeichen, das zuletzt eingegeben wurde, als einer der Steuerbefehle `SWITCH:erg?name` oder `SWITCH:erg?` auf eine Eingabe gewartet hat, an die aktuelle Cursor-Position. Eingaben mit der Maus, Tastenkombinationen für Steuerzeichen und Funktionstasten bleiben dafür unberücksichtigt.

`KEY` Execute key

Wertet die Eingabe aus, die zuletzt für einen der Steuerbefehle `SWITCH:erg?name` und `SWITCH:erg?` erfolgte.

Wurde zuletzt ein Zeichen eingegeben, so wirkt dieser Steuerbefehl wie der Steuerbefehl `CHAR`; wurde eine Tastenkombination für einen Steuerbefehl eingegeben, wird dieser ausgeführt; wurde eine Funktionstaste gedrückt, wird die entsprechende Funktion aufgerufen; wurde eine Tastenkombination für einen Makroaufruf eingegeben, wird das entsprechende Makro aufgerufen.



- REPL\_ABBR**    Replace abbreviation
- Löscht den Namen, in dem bzw. hinter dem der Cursor steht, ergänzt den Namen um einen Abkürzungspunkt und ruft das Makro mit diesem Namen auf. Falls der Cursor nicht in oder hinter einem Namen steht, wird das Makro mit dem Namen ».« aufgerufen.
- T: ALT+P (Voreinstellung)
- SELECT\_ABBR**    Select abbreviation
- Zeigt den Anfang aller Makros an, deren Namen mit einem Abkürzungspunkt endet.
- Soll eines dieser Makros ausgeführt werden, so muss dieses mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Return-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der linken Maustaste angeklickt werden.
- Die Anzeige kann durch Drücken der Leertaste oder Escape-Taste wieder gelöscht werden.
- Um die Liste zu reduzieren, kann mit der Delete-Taste das jeweils markierte Makro aus der Liste entfernt werden. Wird die Anzeige anschließend mit der Escape-Taste gelöscht, so werden alle mit der Delete-Taste entfernten Makros wieder in die Liste aufgenommen.
- T: ALT+Y (Voreinstellung)
- NEXT\_CR:name**    Switch to macro when executing CR next time
- Wenn der Steuerbefehl CR zum nächsten Mal ausgeführt werden soll (z. B. beim Drücken der Return-Taste), wird stattdessen das Makro mit dem angegebenen Namen aufgerufen und ausgeführt.
- NEXT\_CR:\***    Continue macro when executing CR next time
- Die Abarbeitung des Makros wird unterbrochen; wenn der Steuerbefehl CR zum nächsten Mal ausgeführt werden soll (z. B. beim Drücken der Return-Taste), wird stattdessen die Abarbeitung des Makros fortgesetzt.
- NEXT\_TAB:name**    Switch to macro when executing TAB next time
- Wenn der Steuerbefehl TAB zum nächsten Mal ausgeführt werden soll (z. B. beim Drücken der Tabulatortaste), wird stattdessen das Makro mit dem angegebenen Namen aufgerufen und ausgeführt.
- NEXT\_TAB:\***    Continue macro when executing TAB next time
- Die Abarbeitung des Makros wird unterbrochen; wenn der Steuerbefehl TAB zum nächsten Mal ausgeführt werden soll (z. B. beim Drücken der Tabulatortaste), wird stattdessen die Abarbeitung des Makros fortgesetzt.

`NEXT_BTAB:name` Switch to macro when executing `BACKTAB` next time

Wenn der Steuerbefehl `BACKTAB` zum nächsten Mal ausgeführt werden soll (z. B. beim Drücken der Shift- und Tabulatortaste), wird stattdessen das Makro mit dem angegebenen Namen aufgerufen und ausgeführt.

`NEXT_BTAB:*` Continue macro when executing `BACKTAB` next time

Die Abarbeitung des Makros wird unterbrochen; wenn der Steuerbefehl `BACKTAB` zum nächsten Mal ausgeführt werden soll (z. B. beim Drücken der Shift- und Tabulatortaste), wird stattdessen die Abarbeitung des Makros fortgesetzt.

`NEXT_M_AR:name` Switch to macro when releasing any mouse key

Bei der nächsten Ausführung einer Mausaktion »Release« (siehe Seite 344) wird unabhängig davon, ob eine Zusatztaste (Shift, Ctrl oder Alt) gedrückt ist oder nicht, das Makro mit dem angegebenen Namen aufgerufen und ausgeführt.

`NEXT_M_AR:*` Continue macro when releasing any mouse key

Die Abarbeitung des Makros wird unterbrochen und dann fortgesetzt, wenn zum nächsten Mal eine Mausaktion »Release« ausgeführt wird.

`IF_EMPTY:name` Switch to macro if file is empty

Wenn die Editor-Datei leer ist, wird das Makro mit dem angegebenen Namen aufgerufen und ausgeführt; andernfalls werden die nach dem Steuerbefehl folgenden Makroanweisungen ausgeführt.

`NOT_EMPTY:name` Switch to macro if file is not empty

Wenn die Editor-Datei nicht leer ist, wird das Makro mit dem angegebenen Namen aufgerufen und ausgeführt; andernfalls werden die nach dem Steuerbefehl folgenden Makroanweisungen ausgeführt.

`DO_EMPTY:name` Do macro if file is empty

Wenn die Editor-Datei leer ist, wird das Makro mit dem angegebenen Namen aufgerufen und ausgeführt; andernfalls werden gleich die nach dem Steuerbefehl folgenden Makroanweisungen ausgeführt. Falls bei der Ausführung des aufgerufenen Makros kein Fehler auftritt, werden danach ebenfalls die nach diesem Steuerbefehl folgenden Steuerbefehle ausgeführt.

`IF_ZERO:name` Switch to macro if record number is zero

Wenn die aktuelle Satzposition Null (0.0/0 bzw. 0/0) ist, wird das Makro mit dem angegebenen Namen aufgerufen und ausgeführt; andernfalls werden die nach dem Steuerbefehl folgenden Makroanweisungen ausgeführt.

**NOT\_ZERO:name** Switch to macro if record number is not zero

Wenn die aktuelle Satzposition nicht Null (0.0/0 bzw. 0/0) ist, wird das Makro mit dem angegebenen Namen aufgerufen und ausgeführt; andernfalls werden die nach dem Steuerbefehl folgenden Makroanweisungen ausgeführt.

**DO\_ZERO:name** Do macro if record number is zero

Wenn die aktuelle Satzposition Null (0.0/0 bzw. 0/0) ist, wird das Makro mit dem angegebenen Namen aufgerufen und ausgeführt; andernfalls werden gleich die nach dem Steuerbefehl folgenden Makroanweisungen ausgeführt. Falls bei der Ausführung des aufgerufenen Makros kein Fehler auftritt, werden danach ebenfalls die nach diesem Steuerbefehl folgenden Steuerbefehle ausgeführt.

**IF\_FIRST:name** Switch to macro if file is edited the first time

Wenn die Datei zum ersten Mal ediert wird (genauer: wenn noch keine Einstellung der Farbe oder Tags in der Datei vermerkt ist), wird das Makro mit dem angegebenen Namen aufgerufen und ausgeführt; andernfalls werden die nach dem Steuerbefehl folgenden Makroanweisungen ausgeführt.

**NOT\_FIRST:name** Switch to macro if file is not edited the first time

Wenn die Datei nicht zum ersten Mal ediert wird (genauer: wenn schon die Einstellung der Farbe oder Tags in der Datei vermerkt ist), wird das Makro mit dem angegebenen Namen aufgerufen und ausgeführt; andernfalls werden die nach dem Steuerbefehl folgenden Makroanweisungen ausgeführt.

**DO\_FIRST:name** Do macro if file is edited the first time

Wenn die Datei zum ersten Mal ediert wird (genauer: wenn noch keine Einstellung der Farbe oder Tags in der Datei vermerkt ist), wird das Makro mit dem angegebenen Namen aufgerufen und ausgeführt; andernfalls werden gleich die nach dem Steuerbefehl folgenden Makroanweisungen ausgeführt. Falls bei der Ausführung des aufgerufenen Makros kein Fehler auftritt, werden danach ebenfalls die nach diesem Steuerbefehl folgenden Steuerbefehle ausgeführt.

**IF\_MATCH:name** Switch to macro in case of match

Wenn die unten (nach dem Steuerbefehl **DO\_MATCH**) beschriebene **MATCH**-Bedingung erfüllt ist, wird das Makro mit dem angegebenen Namen aufgerufen und ausgeführt; andernfalls werden die nach dem Steuerbefehl folgenden Makroanweisungen ausgeführt.

**NO\_MATCH:name** Switch to macro in case of no match

Wenn die unten (nach dem Steuerbefehl **DO\_MATCH**) beschriebene **MATCH**-Bedingung nicht erfüllt ist, wird das Makro mit dem angegebenen Namen aufgerufen und ausgeführt; andernfalls werden die nach dem Steuerbefehl folgenden Makroanweisungen ausgeführt.

DO\_MATCH:name      Do macro in case of match

Wenn die unten beschriebene MATCH-Bedingung erfüllt ist, wird das Makro mit dem angegebenen Namen aufgerufen und ausgeführt; andernfalls werden gleich die nach dem Steuerbefehl folgenden Makroanweisungen ausgeführt. Falls bei der Ausführung des aufgerufenen Makros kein Fehler auftritt, werden danach ebenfalls die nach diesem Steuerbefehl folgenden Steuerbefehle ausgeführt.

MATCH-Bedingung:

- Die letzte ausgeführte Anweisung war eine erweiterte Zeige-Anweisung; bei ihrer Ausführung wurde eine Zeichenfolge gefunden und seither wurde keiner der unten genannten Steuerbefehle ausgeführt.
- Die letzte ausgeführte Anweisung war eine erweiterte Lösche-Anweisung; bei ihrer Ausführung wurde eine Zeichenfolge gefunden und seither wurde keiner der unten genannten Steuerbefehle ausgeführt.
- Die letzte ausgeführte Anweisung war eine Austausch-Anweisung; bei ihrer Ausführung wurde eine Zeichenfolge ausgetauscht und seither wurde keiner der unten genannten Steuerbefehle ausgeführt.
- Die letzte ausgeführte Anweisung war eine Such-Anweisung; bei ihrer Ausführung wurde eine Zeichenfolge gefunden und seither wurde keiner der unten genannten Steuerbefehle ausgeführt.
- Nach der letzten Anweisung wurde noch der Steuerbefehl JMP\_DN ausgeführt und dabei zwischen der aktuellen Cursor-Position und dem Ende des Textfeldes eine hervorgehobene Zeichenfolge gefunden.
- Nach der letzten Anweisung wurde noch der Steuerbefehl JMP\_UP ausgeführt und dabei zwischen der aktuellen Cursor-Position und dem Anfang des Textfeldes eine hervorgehobene Zeichenfolge gefunden.
- Nach der letzten Anweisung wurde noch einer der Steuerbefehle MRK\_FND, MRK\_DEL\_FND, FND\_BEG, FND\_END oder FND\_BLANK ausgeführt und dabei zwischen der aktuellen Cursor-Position und dem Ende des Textfeldes eine entsprechende Zeichenfolge gefunden.
- Nach der letzten Anweisung wurde noch der Steuerbefehl NEXT\_REC ausgeführt und dabei zwischen der aktuellen Cursor-Position und dem Ende des Textfeldes ein weiterer Satz gefunden.
- Nach der letzten Anweisung wurde noch der Steuerbefehl PREV\_REC ausgeführt und dabei zwischen der aktuellen Cursor-Position und dem Anfang des Textfeldes ein weiterer Satz gefunden.
- Nach der letzten Anweisung wurde noch der Steuerbefehl FND\_TAG\_POS ausgeführt und zu diesem Zeitpunkt der Satz der aktuellen Tag-Position mit der Satznummer angezeigt.

- Nach der letzten Anweisung wurde noch einer der Steuerbefehle MRK\_STR, MRK\_TST oder TEST\_MRK ausgeführt und dabei eine entsprechende Zeichenfolge gefunden.
- Nach der letzten Anweisung wurde noch der Steuerbefehl SET\_MATCH ausgeführt.
- Nach der letzten Anweisung wurde der Steuerbefehl CLR\_MATCH nicht ausgeführt.

Entscheidend ist jeweils nur die zuletzt ausgeführte Anweisung bzw. nur der zuletzt ausgeführte Steuerbefehl.

CHECK\_FN

Check file name

Wenn mit dem Makro FN\_# eine Such-Tabelle definiert ist (vgl. Seite 290), wird eine entsprechende Zeichenfolge im Dateinamen (genauer: in der Dateibezeichnung der Form »Projektname\*Dateiname«) gesucht. Falls die n-te Zeichenfolge der Such-Tabelle als erste gefunden wird, erfolgt der Aufruf des Makros FN\_#, wobei an Stelle des Nummernzeichens die Zahl n eingesetzt wird; falls keine entsprechende Zeichenfolge gefunden wird, erfolgt der Aufruf des Makros FN\_0 (# = Ziffer 0).

CHECK\_FT

Check file title

Wenn mit dem Makro FT\_# eine Such-Tabelle definiert ist (vgl. Seite 290), wird eine entsprechende Zeichenfolge im Dateititel gesucht. Falls die n-te Zeichenfolge der Such-Tabelle als erste gefunden wird, erfolgt der Aufruf des Makros FT\_#, wobei an Stelle des Nummernzeichens die Zahl n eingesetzt wird; falls keine entsprechende Zeichenfolge gefunden wird, erfolgt der Aufruf des Makros FT\_0 (# = Ziffer 0).

CHECK\_FC

Check file contents

Wenn mit dem Makro FC\_# eine Such-Tabelle definiert ist (vgl. Seite 290), wird eine entsprechende Zeichenfolge im ersten Satz der Editor-Datei gesucht. Falls die n-te Zeichenfolge der Such-Tabelle als erste gefunden wird, erfolgt der Aufruf des Makros FC\_#, wobei an Stelle des Nummernzeichens die Zahl n eingesetzt wird; falls keine entsprechende Zeichenfolge gefunden wird, erfolgt der Aufruf des Makros FC\_0 (# = Ziffer 0).

CHECK\_FX

Check file contents extended

Wenn mit dem Makro FX\_# eine Such-Tabelle definiert ist (vgl. Seite 290), wird eine entsprechende Zeichenfolge im ersten und im zweiten Satz der Editor-Datei gesucht. Falls die n-te Zeichenfolge der Such-Tabelle als erste gefunden wird, erfolgt der Aufruf des Makros FX\_#, wobei an Stelle des Nummernzeichens die Zahl n eingesetzt wird; falls keine entsprechende Zeichenfolge gefunden wird, erfolgt der Aufruf des Makros FX\_0 (# = Ziffer 0).

## Anzeigen von Daten

- SHW\_BEG** Show data from the begin of file  
Zeigt die Sätze vom Dateianfang an.  
Wurden im Textfeld Änderungen vorgenommen, so werden diese zuvor in die Datei übertragen.
- SHW\_CUR** Show data around the current record  
Zeigt die Sätze um die aktuelle Satzposition.  
Falls der Cursor in einem Satz steht, wird zuerst die aktuelle Satzposition auf die Satznummer dieses Satzes gesetzt.  
Falls die Satzposition Null ist, werden die Sätze wie bei SHW\_BEG vom Dateianfang an gezeigt.  
Wurden im Textfeld Änderungen vorgenommen, so werden diese zuvor in die Datei übertragen.  
Maus: Shift+Left click = Makro S\_LC (Voreinstellung)
- SHW\_END** Show data from the end of file  
Zeigt die Sätze vom Dateiende her.  
Wurden im Textfeld Änderungen vorgenommen, so werden diese zuvor in die Datei übertragen.

## Anzeigen von Meldungen

- MESSAGE:"text"** Show text in message line  
Zeigt den angegebenen Text in der Meldungszeile an. Der Text muss mit einem frei wählbaren Sonderzeichen, das im Text selbst nicht vorkommt, eingeleitet und abgeschlossen werden.  
Der Text wird in der für die Meldungszeile eingestellten Farbe angezeigt. Soll er in einer anderen Farbe angezeigt werden, so kann nach "text" durch einen Doppelpunkt getrennt der entsprechende Hexadezimal-Code angegeben werden. Die möglichen Hexadezimal-Codes können der Tabelle entnommen werden, die durch die Tastenkombination Ctrl+F bzw. Strg+F angezeigt wird.  
Wird als Text eine leere Zeichenfolge angegeben, so wird der in der Anweisungszeile stehende Text angezeigt. Damit ist es möglich, einen dynamisch erstellten Text anzuzeigen.
- STATUS:"text"** Show text in status line  
Zeigt den angegebenen Text links im freien Bereich der Statuszeile an. Der Text muss mit einem frei wählbaren Sonderzeichen, das im Text selbst nicht vorkommt, eingeleitet und abgeschlossen werden.

Der Text wird in der für die Statuszeile eingestellten Farbe angezeigt. Soll er in einer anderen Farbe angezeigt werden, so kann nach "text" durch einen Doppelpunkt getrennt der entsprechende Hexadezimal-Code angegeben werden. Die möglichen Hexadezimal-Codes können der Tabelle entnommen werden, die durch die Tastenkombination `Ctrl+F` bzw. `Strg+F` angezeigt wird.

Wird als Text eine leere Zeichenfolge angegeben, so wird der in der Anweisungszeile stehende Text angezeigt. Damit ist es möglich, einen dynamisch erstellten Text anzuzeigen.

## Anzeigen von Makroleisten

Die Definition von Makroleisten ist ab Seite 291 beschrieben.

Die folgenden Steuerbefehle beziehen sich auf permanente und optionale Makroleisten. Temporäre oder imaginäre Makroleisten können mit dem Steuerbefehl `SWITCH:erg?name` (siehe Seite 375) angezeigt werden.

`CHG_MACRO:name~`      Change macro bar

Zeigt die permanente Makroleiste `name` in der Menü-Zeile an. Fehlt der Name (nicht aber die Tilde), so wird die namenlose permanente Makroleiste angezeigt.

`CHG_MACRO:name*`      Change macro bar

Zeigt die optionale Makroleiste `name` in der Meldungszeile an. Fehlt der Name (nicht aber der Stern), so wird die namenlose optionale Makroleiste angezeigt.

`HIDE_MACRO`      Hide macro bar

Zeigt die optionale Makroleiste (in der Meldungszeile) nicht mehr an, bis einer der Steuerbefehle `SHOW_MACRO`, `ENTER` oder `CONFIRM` ausgeführt wird.

`SHOW_MACRO`      Show macro bar

Zeigt die optionale Makroleiste (in der Meldungszeile) wieder an, falls sie mit einer Meldung des Editors verdeckt ist.

`WHICH_MACRO`      Which macro bar

Falls eine permanente bzw. optionale Makroleiste eingestellt ist, werden ihre Namen (zu Kontrollzwecken) in der Statuszeile abwechselnd angezeigt.

## Anzeigen von Menüs

`SELECT:name`    Select macro from menu

Zeigt den Inhalt des Segments `name` aus der für Menüs definierten Datei zur Auswahl eines Editormakros im Editorfenster an.

Jede Zeile des Segments muss der Form »`name: Kommentar`« (ohne Anführungszeichen) genügen. Angezeigt werden nur die Kommentare. Nach Auswahl eines Kommentars wird die Anzeige wieder gelöscht und das Makro mit dem dazugehörenden Namen aufgerufen. Die Auswahl kann durch Anklicken mit der linken Maustaste erfolgen oder durch Drücken der Return-Taste nachdem der Cursor ggf. mit den Pfeiltasten in die entsprechende Zeile positioniert wurde.

Ist kein Makro mit dem entsprechenden Namen definiert, wird ein Signalton (`BEEP`) ausgegeben; die auf den Steuerbefehl `SELECT` folgenden Steuerbefehle werden in diesem Fall nicht ausgeführt. Wird kein Kommentar ausgewählt, sondern die Anzeige mit der Escape-Taste wieder gelöscht, so werden die auf den Steuerbefehl `SELECT` folgenden Steuerbefehle ausgeführt.

Stehen im Kommentartext Zeichenfolgen der Form `#:(xx)`, wobei `xx` ein Hexadezimal-Code ist, so werden die in der Zeile nachfolgenden Zeichen in der entsprechenden Farbe dargestellt. Für `xx` sind die gleichen Angaben vorgesehen wie für den Steuerbefehl `MRK_CHG:xx` (siehe Seite 368). Mit `#:(PWN)` kann auf die voreingestellte Farbe zurückgestellt werden.

Hinweis: Aus welcher Datei die Menüs geholt werden, kann beim Aufruf des Editors über die Spezifikation `DEFINITIONEN` bestimmt werden (siehe Seite 152).

## Anzeigen von Hilfetexten

`HELP`            Call online help

Ruft das Online-Hilfe-Programm auf (siehe Kommando `#HILFE` Seite 162).

T: `Ctrl+O`

`DISPLAY:name`    Display help text

Zeigt den Inhalt des Segments `name` aus der für Hilfetexte definierten Datei im Editorfenster an. Die Anzeige kann durch Drücken der Leertaste oder Escape-Taste wieder gelöscht werden. Mit der Tastenkombination `Ctrl+P` bzw. `Strg+P` wird der Inhalt des angezeigten Segments ins Zweitprotokoll kopiert und die Anzeige wieder gelöscht.

Stehen im Hilfetext Zeichenfolgen der Form `#:(xx)`, wobei `xx` eine Farbangabe ist, so werden die in der Zeile nachfolgenden Zeichen in



der entsprechenden Farbe dargestellt. Für `xx` sind die gleichen Angaben vorgesehen wie für den Steuerbefehl `MRK_CHG:xx` (siehe Seite 368). Mit `#:` (`PWN`) kann auf die voreingestellte Farbe zurückgestellt werden.

Hinweis: Aus welcher Datei die Hilfetexte geholt werden, kann beim Aufruf des Editors über die Spezifikation `DEFINITIONEN` bestimmt werden (siehe Seite 152).

`DISPLAY_MRK`    Display buffer

Zeigt den aktuellen Inhalt des Editor-Zwischenspeichers an.

Die Anzeige kann durch Drücken der Leertaste oder Escape-Taste wieder gelöscht werden.

An Stellen, an denen im Editor-Zwischenspeicher die Zeichenfolge `#:` (`NL`) steht, wird in der Anzeige eine neue Zeile begonnen.

Stehen im Editor-Zwischenspeicher Zeichenfolgen der Form `#:(xx)`, wobei `xx` eine Farbangabe ist, so werden die in der Zeile nachfolgenden Zeichen in der entsprechenden Farbe dargestellt. Für `xx` sind die gleichen Angaben vorgesehen wie für den Steuerbefehl `MRK_CHG:xx` (siehe Seite 368). Mit `#:` (`PWN`) kann auf die voreingestellte Farbe zurückgestellt werden.

Hinweis: Der anzuzeigende Text kann z. B. mit den Steuerbefehlen `ADD_MRK` und `APP_MRK` (siehe Seite 370) im Editor-Zwischenspeicher zusammengestellt werden.

## Anzeigen vorgesehener Tags

`SELECT_TAG`    Select tag from menu

Wenn der Cursor außerhalb eines Tags steht, wird eine Liste der Tags angezeigt, die an der Cursor-Position in der Tag-Hierarchie erlaubt sind; wenn der Cursor innerhalb eines Tags steht, wird eine Liste der Attribute angezeigt, die in dem betreffenden Tag erlaubt sind. Dazu müssen alle erlaubten Tags mit ihren Attributen in einer Tag-Gruppe definiert und diese als aktuelle Tag-Gruppe eingestellt sein. Die Anweisungen dafür sind im Kapitel »Tag-Prüfung definieren/...« (siehe Seite 300) beschrieben.

Soll eines dieser Tags bzw. Attribute an der Cursor-Position eingefügt werden, so muss dieses mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Return-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der linken Maustaste angeklickt werden.

Die Anzeige kann durch Drücken der Leertaste oder Escape-Taste wieder gelöscht werden.

T: `ALT+G` (Voreinstellung)

## Anzeigen des Inhalts einer Segment-Datei

FETCH\_SEGM Fetch segment

Zeigt das Inhaltsverzeichnis der eingestellten Segment-Datei (siehe Seite 280) an.

Soll ein Segment in die Editor-Datei geholt werden, so muss dieses mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Return-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der linken Maustaste angeklickt werden.

Soll das Segment nicht automatisch geholt werden, sondern nur die entsprechende Anweisung in die Anweisungszeile geschrieben werden, so muss dieses mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Enter-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der rechten Maustaste angeklickt werden.

Die Anzeige kann durch Drücken der Leertaste oder Escape-Taste wieder gelöscht werden.

T: ALT+H (Voreinstellung)

## Übernehmen von zusätzlichen Definitionen

DEFINE:name Take over definitions

Übernimmt zusätzlich die Definitionen, die im Segment `name` in der für zusätzliche Definitionen definierten Datei stehen.

Hinweis: Aus welcher Datei die zusätzlichen Definitionen geholt werden, kann beim Aufruf des Editors über die Spezifikation `DEFINITIONEN` bestimmt werden (siehe Seite 152).

## Tags

Ein Tag ist eine in spitzen Klammern eingeschlossene Zeichenfolge. Es gibt Anfangs-Tags, Ende-Tags und leere Tags:

Anfangs-Tag:

```
<name>
<name attribut1=wert1, attribut2=wert2, ... >
```

Ende-Tag:

```
</name>
```

Leeres Tag:

```
<name/>
<name attribut1=wert1, attribut2=wert2, ... />
```

Damit die nachfolgend beschriebenen Steuerbefehle die angegebene Wirkung haben, müssen folgende Bedingungen erfüllt sein:

- Die spitzen Klammern dürfen nur als Anfangs- bzw. Endekennung von Tags, in Akzent-Codierungen (z. B. »%<e«) oder in den Codes für doppelte Anführungszeichen (»#. <« und »#. >«) vorkommen; andere spitze Klammern müssen z. B. mit ^< bzw. ^> codiert sein.
- Anfangs- und Endekennung eines Tags (beide spitze Klammern) müssen im selben Satz stehen.
- Tag- und Attribut-Namen dürfen aus Buchstaben, Ziffern, Minuszeichen, Punkt, Doppelpunkt und »\_« bestehen; das erste Zeichen darf kein Minuszeichen und kein Punkt sein.

RD\_TAG\_NAME    Read tag name

Liest den Namen des Tags, in dem der Cursor steht, und speichert ihn als Tag-Namen.

WR\_TAG\_NAME    Write tag name

Schreibt den mit RD\_TAG\_NAME gemerkten Tag-Namen an die aktuelle Cursor-Position.

ADD\_STRT\_TAG    Add start tag

Sucht im Textfeld ab der Cursor-Position vorwärts nach einem Ende-Tag, zu dem im durchsuchten Text kein Anfangs-Tag vorhanden ist, und fügt an der Cursor-Position das entsprechende Anfangs-Tag ein.

T: ALT+A (Voreinstellung)

ADD\_END\_TAG    Add end tag

Sucht im Textfeld ab der Cursor-Position rückwärts nach einem Anfangs-Tag, zu dem im durchsuchten Text kein Ende-Tag vorhanden ist, und fügt an der Cursor-Position das entsprechende Ende-Tag ein.

T: ALT+E (Voreinstellung)

SET\_TAG\_POS    Set tag position

Merkt die Tag-Position des Tags, in dem der Cursor steht.

Voraussetzung: Satznummern müssen angezeigt sein.

Eine Tag-Position besteht aus der Satznummer des entsprechenden Satzes und einer Zahl, die angibt, um das wievielte Tag es sich in diesem Satz handelt. Steht der Cursor nicht innerhalb eines Tags, gibt die Zahl an, wieviele Tags in diesem Satz vor dem Cursor stehen.

Hinweis: Die so gemerkte Tag-Position gilt zugleich als aktuelle Tag-Position bis diese durch eine der Anweisungen TSV oder TSR geändert wird.

- FND\_TAG\_POS** Find tag position
- Setzt den Cursor in das Tag, das der aktuellen Tag-Position entspricht.
- Voraussetzung: Satznummern müssen angezeigt sein.
- Enthält der Satz weniger Tags, springt der Cursor ans Ende des Satzes und die MATCH-Bedingung (siehe Seite 380) ist nicht erfüllt. Falls der entsprechende Satz nicht angezeigt wird, bleibt die Cursor-Position unverändert und die MATCH-Bedingung ist nicht erfüllt.
- Hinweis: Die aktuelle Tag-Position wird durch den Steuerbefehl `SET_TAG_POS` und durch die Anweisungen `TSV` und `TSR` bestimmt.
- SHOW\_TAGS** Show open tags
- Prüft die Tags vom Dateianfang an bis zur Cursor-Position und zeigt die an der Cursor-Position noch offenen Tags an. Steht der Cursor in der Anweisungszeile, so werden die Tags bis zur aktuellen Satzposition geprüft (vgl. »Anzeigen offener Tags« Seite 318). Hinter jedem Tag wird in eckigen Klammern angegeben, das wievielte gleichnamige Tag es ist, das jeweils dieselben (!) übergeordneten Tags hat; beim jeweils ersten Tag wird diese Angabe (»[1]«) unterdrückt.
- Wurden im Textfeld Änderungen vorgenommen, so werden diese zuvor in die Datei übertragen.
- T: `ALT+T` (Voreinstellung)
- XPATH\_MRK** Copy xpath to buffer
- Kopiert die Namen der Tags in den Editor-Zwischenspeicher, die bei der zuletzt ausgeführten Anweisung `tz` (vgl. »Anzeigen offener Tags« Seite 318) bzw. Steuerbefehl `SHOW_TAGS` als offen angezeigt wurden.
- SHW\_STRT\_TAG** Show start tag
- Sucht von der Cursor-Position an zum Dateianfang hin (rückwärts) nach dem nächsten Anfangs-Tag, für das in den durchsuchten Daten kein entsprechendes Ende-Tag vorhanden ist, und zeigt den betreffenden Satz an. Steht der Cursor in der Anweisungszeile, so wird von der aktuellen Tag-Position an gesucht (vgl. »Suchen von Tags« Seite 321).
- Wurden im Textfeld Änderungen vorgenommen, so werden diese zuvor in die Datei übertragen.
- T: `ALT+R` (Voreinstellung)
- SHW\_END\_TAG** Show end tag
- Sucht von der Cursor-Position an zum Dateiende hin (vorwärts) nach dem nächsten Ende-Tag, für das in den durchsuchten Daten kein entsprechendes Anfangs-Tag vorhanden ist, und zeigt den betreffenden Satz an. Steht der Cursor in der Anweisungszeile, so wird von der aktuellen Tag-Position an gesucht (vgl. »Suchen von Tags« Seite 321).
- Wurden im Textfeld Änderungen vorgenommen, so werden diese zuvor in die Datei übertragen.
- T: `ALT+V` (Voreinstellung)

## Laufende Nummer

Der Editor verwaltet intern eine laufende Nummer. Sie hat zu Anfang einer TUSTEP-Sitzung den Wert Null und kann mit den folgenden Steuerbefehlen verändert und verwendet werden. Für die Nummer sind Zahlenwerte von 0 bis 999999999 zugelassen.

RD_NUM	Read and set current number Liest eine an der aktuellen Cursor-Position stehende Ziffernfolge als Zahl ein und speichert sie als laufende Nummer. Steht an der aktuellen Cursor-Position keine Ziffernfolge, bleibt die laufende Nummer unverändert.
DEC_NUM	Decrement current number by 1 Erniedrigt die laufende Nummer um 1.
INC_NUM	Increment current number by 1 Erhöht die laufende Nummer um 1.
WR_NUM	Write current number Schreibt die laufende Nummer an die aktuelle Cursor-Position.
RST_NUM	Reset current number to 0 Setzt die laufende Nummer auf Null zurück.

## Satznummer

REC_NR	Write current record number Schreibt die aktuelle Satznummer (d. i. die in der Anweisungszeile nach »*=« stehende Satznummer) an die aktuelle Cursor-Position.
RD_REC_NR	Read record number Liest eine an der aktuellen Cursor-Position stehende Satznummer und merkt sie. Steht der Cursor im Feld mit der Satznummer, darf er auch vor oder hinter der Satznummer stehen.
WR_REC_NR	Write record number Schreibt die mit RD_REC_NR gelesene Satznummer an die aktuelle Cursor-Position.

## Seitennummer

PAGE_NR	Write current page number Schreibt die aktuelle Seitennummer (d. i. die in der Anweisungszeile nach »*=« stehende Seitennummer) an die aktuelle Cursor-Position.
---------	---

PAGE\_NR\_INC Write current page number incremented by 1  
Schreibt die um 1 erhöhte aktuelle Seitennummer (d. i. die in der Anweisungszeile nach »\*=« stehende Seitennummer) an die aktuelle Cursor-Position.

PAGE\_NR\_DEC Write current page number decremented by 1  
Schreibt die um 1 erniedrigte aktuelle Seitennummer (d. i. die in der Anweisungszeile nach »\*=« stehende Seitennummer) an die aktuelle Cursor-Position.

## Dateiname und Segmentname

EDIT\_FILE Write name of edit file  
Schreibt den Namen der Editor-Datei an die aktuelle Cursor-Position.

SEGM\_FILE Write name of segment file  
Schreibt den Namen der eingestellten Segment-Datei an die aktuelle Cursor-Position.

SEGM\_NAME Write name of segment  
Schreibt den Namen des eingestellten Segments an die aktuelle Cursor-Position.

WHICH\_SEGM Which segment  
Falls eine Segment-Datei ediert wird und der Cursor in einem zu einem Segment gehörenden Satz steht bzw. der Cursor in der Anweisungszeile steht und die aktuelle Satzposition auf eine zu einem Segment gehörenden Satz zeigt, wird der Name dieses Segments in der Statuszeile angezeigt.  
T: ALT+W (Voreinstellung)

## Aktuelles Datum

DATE\_0 Write weekday  
Schreibt den Wochentag (z. B. »Sonntag«) an die aktuelle Cursor-Position.

DATE\_1 Write date as xx.xx.xx  
Schreibt das Datum in der Form xx.xx.xx (z. B. »25.01.16«) an die aktuelle Cursor-Position.

DATE_2	Write date as <code>xx. xxx. xxxx</code> Schreibt das Datum in der Form <code>xx. xxx. xxxx</code> (z. B. »25. Jan. 2019«) an die aktuelle Cursor-Position.
DATE_3	Write date as <code>xx. xxxxxxxx xxxx</code> Schreibt das Datum in der Form <code>xx. xxxxxx xxxx</code> (z. B. »25. Januar 2019«) an die aktuelle Cursor-Position.
DATE_4	Write date as <code>xxxx-xx-xx</code> Schreibt das Datum in der Form <code>xxxx-xx-xx</code> (z. B. »2019-01-25«) an die aktuelle Cursor-Position.

### Aktuelle Uhrzeit

TIME_1	Write time as <code>hh.mm</code> Schreibt die Uhrzeit in der Form <code>hh.mm</code> (z. B. »12.34«) an die aktuelle Cursor-Position.
TIME_2	Write time as <code>hh:mm</code> Schreibt die Uhrzeit in der Form <code>hh:mm</code> (z. B. »12:34«) an die aktuelle Cursor-Position.

### Aktuelle Satzposition

Die aktuelle Satzposition wird in der Anweisungszeile angezeigt. Sie ist die Satznummer des Satzes, der

- als letzter in die Datei eingefügt wurde bzw.
- als letzter in der Datei geändert wurde bzw.
- als letzter in der Datei gelöscht wurde bzw.
- als letzter im Textfeld neu angezeigt wurde bzw.

ist die Satznummer, die als letzte mit einem der nachfolgend beschriebenen Steuerbefehle als aktuelle Satznummer definiert wurde.

SAVE_REC_NR	Save record number Merkt die aktuelle Satzposition.
REST_REC_NR	Restore record number Setzt die aktuelle Satzposition auf die zuletzt mit <code>SAVE_REC_NR</code> oder <code>EXCH_REC_NR</code> gemerkte Satznummer. Die Anzeige der aktuellen Satzposition in der Anweisungszeile wird jedoch nicht aktualisiert.

- EXCH\_REC\_NR** Exchange record number  
Merkt die aktuelle Satzposition und setzt gleichzeitig die aktuelle Satzposition auf die zuletzt mit `SAVE_REC_NR` oder `EXCH_REC_NR` gemerkte Satznummer. Die Anzeige der aktuellen Satzposition in der Anweisungszeile wird jedoch nicht aktualisiert.
- XFER\_REC\_NR** Transfer record number  
Setzt die aktuelle Satzposition auf die Satznummer des Satzes im Textfeld, in dem sich der Cursor befindet. Die Anzeige der aktuellen Satzposition in der Anweisungszeile wird jedoch nicht aktualisiert.
- RST\_REC\_NR** Reset record number  
Setzt die aktuelle Satzposition auf die Satznummer zurück, die beim Öffnen der Editor-Datei (z. B. nach dem Wechseln der Datei mit der Datei-Anweisung) die aktuelle Satznummer war. Die Anzeige der aktuellen Satzposition in der Anweisungszeile wird jedoch nicht aktualisiert.

## Aktuelle Wortposition

- GET\_WRD\_NUM** Get word number  
Merkt die Satz- und Wortnummer des Wortes im Editorfenster, in dem sich der Cursor befindet.  
Voraussetzung: Satz muss mit Seiten- und Zeilennummer angezeigt sein.
- PUT\_WRD\_NUM** Put word number  
Schreibt die mit `GET_WRD_NUM` gemerkte Satz- und Wortnummer an die aktuelle Cursor-Position.

## Zwischenablage

- MRK\_CB** Copy marked text to Clipboard  
Kopiert den markierten Text in die Zwischenablage.  
T: `Ctrl+C`  
Maus: `Shift+Left release` = Makro `S_LR` (Voreinstellung)
- MRK\_DEL\_CB** Move marked text to Clipboard  
Wie `MRK_CB`, jedoch wird zusätzlich der markierte Text im Editorfenster gelöscht.  
T: `Ctrl+X`



INSERT_CB	<p>Insert clipboard into the text</p> <p>Fügt den Inhalt der Zwischenablage vor der aktuellen Cursor-Position ein. Alle in der Zeile folgenden Zeichen werden nach rechts verschoben. Der Cursor wird mitverschoben und steht danach hinter den eingefügten Zeichen.</p> <p>T: Ctrl+V</p> <p>Maus: Shift+Right click = Makro S_RC (Voreinstellung)</p>
MRK_INS_CB	<p>Replace marked text with clipboard contents</p> <p>Löscht den markierten Text, ohne ihn vorher im Editor-Zwischenspeicher zu merken, und fügt den Inhalt der Zwischenablage an dieser Stelle ein.</p> <p>Maus: Shift+Right release = Makro S_RR (Voreinstellung)</p>
DEFINE_CB	<p>Define Clipboard</p> <p>Kopiert den Inhalt des Editor-Zwischenspeichers (z. B. mit MRK_REP belegt) in die Zwischenablage.</p>
FETCH_CB	<p>Fetch Clipboard</p> <p>Kopiert den Inhalt der Zwischenablage in den Editor-Zwischenspeicher des Editors.</p>
EXCH_CB	<p>Exchange Clipboard</p> <p>Tauscht den Inhalt des Editor-Zwischenspeichers mit dem Inhalt der Zwischenablage aus.</p>
RECALL_CB	<p>Recall Clipboard</p> <p>Zeigt die früheren Inhalte der Zwischenablage an; es werden jedoch nur diejenigen angezeigt, die mit TUSTEP in die Zwischenablage geschrieben bzw. aus der Zwischenablage gelesen wurden.</p> <p>Soll ein früherer Inhalt wieder in die Zwischenablage geholt werden, so muss dieser mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Return-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der linken Maustaste angeklickt werden.</p> <p>Soll ein früherer Inhalt nur vollständig angezeigt werden, so muss dieser mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Enter-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der rechten Maustaste angeklickt werden.</p> <p>Die Anzeige kann durch Drücken der Leertaste oder Escape-Taste wieder gelöscht werden.</p> <p>Um die Liste zu reduzieren, kann mit der Delete-Taste die jeweils markierte Zeichenfolge aus der Liste entfernt werden. Wird die Anzeige anschließend mit der Escape-Taste gelöscht, so werden alle mit der Delete-Taste entfernten Zeichenfolgen wieder in die Liste aufgenommen.</p>

T: ALT+C (Voreinstellung)

## Anzeigen von Dateien, Starten von Anwendungen

BROWSE:"file"      Browse file

Startet die Anwendung, die für die Endung der angegebenen Datei im Betriebssystem definiert ist. Der Dateiname muss mit dem vollständigen Pfad versehen sein und mit einem frei wählbaren Sonderzeichen, das weder im Pfad noch im Dateinamen vorkommt, eingeleitet und abgeschlossen werden.

BRS\_MRK      Browse file given in buffer

Startet die Anwendung, die für die Endung der Datei, deren Name im Editor-Zwischenspeicher steht, im Betriebssystem definiert ist. Der Inhalt des Editor-Zwischenspeicher muss ein Dateiname ohne Projektname nach der für TUSTEP üblichen Konvention oder nach der für das jeweilige Betriebssystem üblichen Konvention sein (unter Windows mit Laufwerksbuchstabe oder Tilde beginnend, unter Linux und macOS mit Schrägstrich oder Tilde beginnend). Enthält der Editor-Zwischenspeicher nur einen Dateinamen, so wird der Pfad der Editor-Datei (ohne Dateiname) intern ergänzt.

## Carriage Return

CR      Carriage Return

– im SPLIT-Modus (vgl. CHG\_SETTINGS Seite 395):

CR führt den Steuerbefehl SPLIT aus, wenn der Cursor im Textfeld steht, und den Steuerbefehl ENTER, wenn der Cursor in der Anweisungszeile steht.

– im ENTER-Modus (vgl. CHG\_SETTINGS Seite 395):

Während der Dateneingabe (z. B. nach der Anweisung ee) wird automatisch in den LINEFEED-Modus geschaltet. Dies bedeutet, dass CR den Steuerbefehl LF ausführt, wenn der Cursor im Textfeld steht, und den Steuerbefehl ENTER, wenn der Cursor in der Anweisungszeile steht.

Sonst bleibt der ENTER-Modus eingestellt. Dies bedeutet, dass CR den Steuerbefehl ENTER ausführt, unabhängig davon, wo der Cursor steht.

– im MACRO-Modus (vgl. CHG\_SETTINGS Seite 395):

CR führt das Makro mit dem Namen CR aus, wenn der Cursor im Textfeld steht, und das Makro mit dem Namen CMD, wenn der Cursor in der Anweisungszeile steht. Falls das Makro CR bzw. CMD nicht definiert ist, hat CR die gleiche Wirkung wie im SPLIT-Modus.

Falls eine Makroleiste (siehe Seite 291) angezeigt wird, werden die Makros `nameCR` bzw. `nameCMD` aufgerufen, wobei `name` der Name der Makroleiste ist. Falls das Makro `nameCR` bzw. `nameCMD` nicht definiert ist, hat `CR` die gleiche Wirkung, wie wenn keine Makroleiste angezeigt wird.

T: Return, Ctrl+M

#### CHG\_SETTINGS      Change settings

Zeigt ein Menü an, mit dem Einstellung geändert werden können.

- `SPLIT`, `MACRO` oder `ENTER` für Return (`CR`)
- `SCROLL` ein/aus für `CUR_UP` und `CUR_DN`
- `INSERT` oder `REPLACE` für Dateneingabe

Hierzu muss die entsprechende Zeile mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Return-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der linken Maustaste angeklickt werden.

Die Anzeige kann durch Drücken der Leertaste oder Escape-Taste wieder gelöscht werden.

T: Ctrl+Y

#### RST\_CR              Reset Carriage Return

Schaltet den Steuerbefehl `CR` in den `SPLIT`-Modus.

#### NEXT\_CR:name      Switch to macro when executing CR next time

Wenn der Steuerbefehl `CR` zum nächsten Mal ausgeführt werden soll, wird stattdessen das Makro mit dem angegebenen Namen aufgerufen und ausgeführt.

## Anzeige-Modus

#### REC\_NR\_ON          Record number on

Daten mit Satznummern anzeigen.

#### REC\_NR\_OFF        Record number off

Daten ohne Satznummern anzeigen.

#### ENCODE\_ON         Encode on

Daten im eingestellten Code anzeigen.

#### ENCODE\_OFF        Encode off

Daten in Eingabe-Codierung anzeigen.

---

ECHO_OFF	Screen output off Schaltet die Bildschirmausgabe aus. Danach werden die durch Steuerbefehle vorgenommenen Änderungen nicht mehr angezeigt. Falls nach Beenden eines Editormakros die Bildschirmausgabe noch nicht explizit mit ECHO_ON wieder eingeschaltet wurde, werden automatisch noch die Steuerbefehle ECHO_ON und REFRESH ausgeführt.
ECHO_ON	Screen output on Schaltet die Bildschirmausgabe wieder ein. Achtung: ECHO_ON aktualisiert die Bildschirmanzeige nicht; ggf. muss dies mit REFRESH explizit veranlasst werden.

## Fehlersuche

TRACE_ON	Trace on Stellt den Testmodus für die Ausführung der Editormakros ein. Dieser kann mit TRACE_OFF wieder beendet werden. In diesem Testmodus werden während der Ausführung von Editormakros die Steuerbefehle einzeln angezeigt und erst ausgeführt, wenn ein Leerzeichen eingegeben wird. Bei jeder anderen Tastatureingabe wird die Ausführung des Makros abgebrochen und die Eingabe ausgewertet; Eingaben mit der Maus werden ignoriert.
TRACE_OFF	Trace off Hebt den Testmodus für die Ausführung der Editormakros wieder auf.
KEY_TST	Key test Stellt die Tastatur in den Testmodus um. Dieser kann durch Drücken der Escape-Taste oder Eingabe eines beliebigen Zeichens wieder beendet werden. In diesem Testmodus werden mit der Tastatur oder Maus eingegebene Steuerbefehle, Funktions- und Makroaufrufe nicht ausgeführt, sondern es wird jeweils nur deren Name in der Statuszeile angezeigt.

T: Ctrl+U

## Weitere Steuerbefehle

- EXCH\_CHAR** Exchange character  
Tauscht das Zeichen auf der aktuellen Cursor-Position mit dem unmittelbar davor stehenden Zeichen aus.  
T: `Ctrl+T`
- EXCH\_CHAR\_LE** Exchange character left  
Tauscht das Zeichen auf der aktuellen Cursor-Position mit dem unmittelbar davor stehenden Zeichen aus; der Cursor rückt um eine Position nach links.  
Bei n-maliger Ausführung von **EXCH\_CHAR\_LE** wird das Zeichen um n Zeichen nach links verschoben.  
T: `ALT+Pfeil nach links`
- EXCH\_CHAR\_RI** Exchange character right  
Tauscht das Zeichen auf der aktuellen Cursor-Position mit dem unmittelbar danach stehenden Zeichen aus; der Cursor rückt um eine Position nach rechts.  
Bei n-maliger Ausführung von **EXCH\_CHAR\_RI** wird das Zeichen um n Zeichen nach rechts verschoben.  
T: `ALT+Pfeil nach rechts`
- EXCH\_REC\_DN** Exchange record down  
Tauscht den Satz, in dem der Cursor steht, mit dem im Textfeld unmittelbar nachfolgenden Satz aus; die Satznummern werden dabei nicht mit ausgetauscht.  
Bei n-maliger Ausführung von **EXCH\_REC\_DN** wird der Satz um n Sätze nach unten verschoben.  
T: `ALT+Pfeil nach unten`
- EXCH\_REC\_UP** Exchange record up  
Tauscht den Satz, in dem der Cursor steht, mit dem im Textfeld unmittelbar vorangehenden Satz aus; die Satznummern werden dabei nicht mit ausgetauscht.  
Bei n-maliger Ausführung von **EXCH\_REC\_UP** wird der Satz um n Sätze nach oben verschoben.  
T: `ALT+Pfeil nach oben`
- SAVE\_CMD** Save command  
Merkt die in der Anweisungszeile stehende Zeichenfolge und löscht diese. Der Cursor steht anschließend in der Anweisungszeile.

---

REST_CMD	Restore command Ersetzt die in der Anweisungszeile stehende Zeichenfolge mit der zuletzt mit SAVE_CMD gemerkten Zeichenfolge. Der Cursor steht anschließend in der Anweisungszeile hinter dieser Zeichenfolge.
JOIN	Join records – bei Dateneingabe (z. B. nach der Anweisung ee): Hängt die aktuelle Zeile mit allen Zeilen von der vorangehenden Leerzeile bis zur nachfolgenden Leerzeile zusammen. – sonst: Hängt den Satz, in dem der Cursor steht, mit dem im Textfeld unmittelbar vorangehenden Satz zusammen. T: ALT+J (Voreinstellung), Plus-Plus-Return
JOIN_LINE	Join lines Hängt die aktuelle Zeile mit der folgenden Zeile und allen ggf. daran anschließenden Fortsetzungszeilen zusammen.
ENTER	End of input Bestätigen der Änderungen bzw. der Eingabe. Daten werden in die Datei übertragen; Anweisungen werden ausgeführt. Der Cursor springt an den Anfang des Eingabebereichs in der Anweisungszeile. T: Enter, Shift+Return, Ctrl+E
CONFIRM	Confirm modifications Wie ENTER, falls im Editorfenster seit dem letzten ENTER Änderungen vorgenommen wurden, sonst wie CMD_LINE.
IGNORE	Ignore modifications Die im Textfeld vorgenommenen Änderungen werden ignoriert.
CANCEL	Cancel input / terminate editor – bei Dateneingabe (z. B. nach der Anweisung ee): Dateneingabe wird beendet; evtl. noch im Textfeld stehende Daten werden ignoriert. – bei Anzeige eines Menüs oder Hilfetextes: Anzeige wird gelöscht und zur vorherigen Anzeige zurückgekehrt. – sonst: Editor wird beendet; eine evtl. vorgenommene Änderungen im Textfeld, die noch nicht bestätigt und in die Datei übertragen wurden (z. B. mit der Enter-Taste), oder eine evtl. eingegebene Anweisung, die noch nicht ausgeführt wurde (z. B. mit der Return-Taste), wird ignoriert. T: Esc, Ctrl+D

COLOR:n	<p>Set color group n</p> <p>Stellt für die Colorierung die Farbgruppe n (n = 1 bis 9) als aktuelle Farbgruppe ein bzw. schaltet die Colorierung aus (n = 0).</p>
TAGS:n	<p>Set tag group n</p> <p>Schaltet für die Tag-Prüfung die zusätzliche Verwendung der in der Tag-Gruppe n (n = 1 bis 9) definierten Regeln ein bzw. schaltet für die Tag-Prüfung die zusätzliche Verwendung der in den Tag-Gruppen definierten Regeln aus (n = 0).</p>
CHG_TF	<p>Change text field</p> <p>Wenn das Textfeld horizontal oder vertikal geteilt ist, wird das andere Textfeld aktiviert und der Cursor springt in die Anweisungszeile.</p>
SELECT_CHAR	<p>Select special character</p> <p>Zeigt Sonderzeichen und Umlaute an, die auf manchen Tastaturen fehlen oder schwer erreichbar sind.</p> <p>Soll eines dieser Zeichen an der Cursor-Position eingefügt werden, so muss es mit Pfeil nach unten/oben ausgewählt und die Auswahl mit der Return-Taste bestätigt werden, oder es muss die entsprechende Zeile mit der linken Maustaste angeklickt werden.</p> <p>Die Anzeige kann durch Drücken der Leertaste oder Escape-Taste wieder gelöscht werden.</p> <p>T: ALT+Q (Voreinstellung)</p>
RESHOW	<p>Reshow textfield</p> <p>Wiederholt die letzte Anzeige im Textfeld. Dies kann z. B. nach unerwünschten Änderungen im Textfeld hilfreich sein.</p> <p>T: Plus-Plus-Stern, Ctrl+R</p>
REFRESH	<p>Refresh screen</p> <p>Stellt den durch Systemmeldungen, Leitungsstörungen u. ä. zerstörten Fensterinhalt wieder her.</p> <p>T: Ctrl+W</p>
CLICK	<p>Click</p> <p>Keine besondere Wirkung.</p> <p>Dieser Steuerbefehl ist erforderlich, wenn ein mit der Maus aufgerufenes Makro (z. B. M_LC) nichts weiter machen soll, als den CURSOR an die Stelle zu positionieren, an der der Mauszeiger steht.</p> <p>Maus: Left click = Makro M_LC (Voreinstellung)</p>

---

BEEP	<p>Beep</p> <p>Gibt einen Signalton aus.</p> <p>Die Tonhöhe und Tonlänge kann mit der Tastenkombination <code>Ctrl+G</code> bzw. <code>Strg+G</code> (siehe Seite 339) oder mit dem Kommando <code>#DEFINIERE</code> (siehe Seite 132) eingestellt werden.</p>
WAIT	<p>Wait until striking any key</p> <p>Unterbricht das Abarbeiten des Editormakros und wartet auf eine Eingabe. Wird die Leertaste gedrückt, wird das Editormakro fortgesetzt; bei jeder anderen Taste werden die restlichen Steuerbefehle des Editormakros übergangen.</p>
PRINT	<p>Print screen</p> <p>Falls der Cursor in der Anweisungszeile steht, wird das Editorfenster ins Zweitprotokoll kopiert; falls der Cursor im Textfeld steht, wird nur das Textfeld ins Zweitprotokoll kopiert. Dies geschieht unabhängig davon, ob das Zweitprotokoll ein- oder ausgeschaltet ist (vgl. Kommando <code>#PROTOKOLL</code> Seite 209).</p> <p>T: <code>Ctrl+P</code></p>
MEM_OFF	<p>Memory off</p> <p>Schaltet das Merken der Anweisungen (vgl. »Ausgeben und Wiederholen von Anweisungen« Seite 304) aus. Danach werden keine Anweisungen mehr gemerkt.</p> <p>Falls nach Beenden eines Editormakros das Merken der Anweisungen noch nicht explizit mit <code>MEM_ON</code> wieder eingeschaltet wurde, wird automatisch noch der Steuerbefehl <code>MEM_ON</code> ausgeführt.</p>
MEM_ON	<p>Memory on</p> <p>Schaltet das Merken der Anweisungen wieder ein.</p>
TXT_CHG:xx	<p>Change the color of text to xx</p> <p>Wechselt die Farbe, in der der Text angezeigt wird, der danach auf Grund von Tastatureingaben oder Steuerbefehlen im Textfeld des Editors eingefügt wird.</p> <p>Für xx sind die gleichen Angaben vorgesehen wie für den Steuerbefehl <code>MRK_CHG:xx</code> (siehe Seite 368).</p>
SET_MATCH	<p>Set match condition</p> <p>Setzt die MATCH-Bedingung (siehe Seite 380).</p>
CLR_MATCH	<p>Clear match condition</p> <p>Löscht die MATCH-Bedingung (siehe Seite 380).</p>



DO_NOTHING	Do nothing Hat keine Wirkung.
STOP	Stop macro Beendet das Abarbeiten von Makros
ERROR	Stop macro with error signal Beendet das Abarbeiten von Makros mit einem Fehlersignal.



**Makros**

---

Allgemeines . . . . .	407
Aufruf . . . . .	407
Makrovariablen . . . . .	410
Makroanweisungen . . . . .	413
Spezifikationen . . . . .	413
Interpretationsmodi einstellen . . . . .	413
Steuerzeichen ändern . . . . .	417
Definition von Variablen – Wertzuweisungen . . . . .	418
Freigabe von Variablen . . . . .	419
Definieren von Trägerangaben . . . . .	420
Zugriff auf Makrovariablen . . . . .	420
Zugriff auf Tabellen . . . . .	421
Zugriff auf TUSTEP-Variablen . . . . .	422
Zugriff auf definierte Dateinamen . . . . .	423
Zugriff auf System-Variablen . . . . .	424
Zugriff auf die Zwischenablage . . . . .	425
Zugriff auf Editor-Information . . . . .	425
Anfragen . . . . .	426
Dateneingabe im Dialog-Modus . . . . .	427
Dateneingabe im Batch-Modus . . . . .	427
Dateneingabe von Dateien . . . . .	427
Datenausgabe in Dateien . . . . .	428
Ausgabe in eine Datei umleiten . . . . .	430
Ausgabe automatisch modifizieren . . . . .	431
Kommentare . . . . .	431
Meldungen . . . . .	431
Fehlermeldungen und Fehlerflag . . . . .	433
Signalton . . . . .	434
Warten . . . . .	434
Schleifen . . . . .	435
Auswahl über beliebige Bedingungen . . . . .	437
Auswahl über Zeichenfolgen oder Zahlenwerte . . . . .	438
Such-Tabellen, Austausch-Tabellen, Recherchier-Tabellen . . . . .	440
Wörterbücher . . . . .	445
Stapelspeicher . . . . .	448
Sektionen . . . . .	450
Submakros . . . . .	451
Einfügen von Segmenten/Dateien . . . . .	452
Compilieren von Makroanweisungen . . . . .	453
Ausführen von Kommandos . . . . .	456
Ausführen von System-Kommandos . . . . .	457
Ausführen von Programmen . . . . .	457
Starten einer TUSTEP-Sitzung . . . . .	458
Starten des TUSTEP-Editors . . . . .	458
Starten und Beenden von Anwendungen . . . . .	458
Fehlersuche . . . . .	459

Anstehende Kommandos löschen . . . . .	460
Abarbeiten des Makros beenden . . . . .	460
TUSTEP-Sitzung beenden . . . . .	460
Bedingungen . . . . .	461
Vergleichen von Zahlen . . . . .	462
Vergleichen von Zeichenfolgen . . . . .	463
Prüfen von Zeichenfolgen mit Hilfe von Schlüsselwörtern . . . . .	465
Prüfen von Zeichenfolgen mit Hilfe von Tabellen . . . . .	467
Prüfen von Projektnamen, Dateinamen und Dateien . . . . .	469
Abfragen von Einstellungen . . . . .	472
Status-Abfragen . . . . .	473
Rechenanweisungen . . . . .	477
Makrofunktionen . . . . .	479
Makrofunktionen zur Dateiverwaltung . . . . .	479
Makrofunktionen zur Dateisperre . . . . .	490
Makrofunktionen für Dateiinhalte . . . . .	491
Makrofunktionen für Datei- und Projektnamen . . . . .	497
Makrofunktionen für Datenträger . . . . .	501
Makrofunktionen für Datum und Uhrzeit . . . . .	502
Makrofunktionen für sonstige Informationen . . . . .	506
Makrofunktion zum Abschicken einer E-Mail . . . . .	508
Makrofunktion zum Informationsabruf von WWW-Servern . . . . .	508
Makrofunktionen für beliebige Variablen-Inhalte . . . . .	509
Makrofunktionen für Teilzeichenfolgen/Zeilen . . . . .	541
Makrofunktionen für Wertelisten . . . . .	553
Makrofunktionen für Tags . . . . .	559
Makrofunktion zum Prüfen von Klammern . . . . .	569
Makrofunktionen zum Sortieren . . . . .	570
Makrofunktion für nicht-numerische/mehrstufige Zähler . . . . .	575
Makrofunktion für Zufallszahlen . . . . .	575
Makrofunktion zum Decodieren . . . . .	576
Makrofunktion zum Codieren . . . . .	578
Makrofunktionen zur Anzeige einer Meldung/Anfrage . . . . .	581
Makrofunktionen zur Anzeige eines Eingabefeldes . . . . .	583
Makrofunktionen zur Anzeige eines Auswahlfeldes . . . . .	587
Makrofunktion zur Anzeige einer Farbauswahl . . . . .	590
Makrofunktionen für Makrofenster . . . . .	591
Prüfen von Klammern . . . . .	594
Dateizugriffe – Allgemeines . . . . .	597
Warten bis Dateizugriff möglich . . . . .	598
Suchbedingungen . . . . .	599

---

Dateizugriffe – Sätze, Zeilen, Seiten . . . . .	603
Definieren des Dateizugriffs . . . . .	603
Schreiben einer Texteinheit . . . . .	605
Lesen einer Texteinheit . . . . .	606
Prüfen einer Texteinheit . . . . .	607
Suchen einer bestimmten Texteinheit . . . . .	607
Zählen bestimmter Texteinheiten . . . . .	609
Löschen einer Texteinheit . . . . .	609
Dateizugriffe – Daten mit Anfangs- und Endekennungen . . . . .	610
Definieren des Dateizugriffs . . . . .	611
Schreiben einer Textportion . . . . .	613
Lesen einer Textportion . . . . .	615
Dateizugriffe – Daten mit Tags . . . . .	617
Definieren des Dateizugriffs . . . . .	619
Schreiben einer Textportion . . . . .	622
Lesen einer Textportion . . . . .	623
Dateizugriffe – Daten mit definierten Strukturen . . . . .	627
Nummerierung . . . . .	627
Datenstruktur . . . . .	627
Definieren des Dateizugriffs . . . . .	631
Schreiben einer Einheit . . . . .	632
Lesen einer Einheit . . . . .	632
Prüfen einer Einheit . . . . .	633
Suchen einer bestimmten Einheit . . . . .	633
Zählen bestimmter Einheiten . . . . .	635
Löschen einer Einheit . . . . .	635
Beispiel für Dateizugriffe . . . . .	636
Datei-Transfer . . . . .	639
Import einer Datei in die aktuelle TUSTEP-Sitzung . . . . .	639
Export einer Datei aus der aktuellen TUSTEP-Sitzung . . . . .	639
Makrofenster . . . . .	641
Anzeigen eines Makrofensters . . . . .	641
Definition eines Makrofensters . . . . .	641
Makroanweisungen für die einzelnen Felder . . . . .	644
Feldtypen, ihre Eigenschaften und Steuerbefehle . . . . .	646
Testhilfen . . . . .	658
Beispiele . . . . .	659

## Allgemeines

Makros bieten die Möglichkeit, eigene Kommandos zu definieren. Die Leistung eines solchen Kommandos wird durch das Makro vorgegeben und kann auf Grund von Spezifikationsangaben, Zustandsabfragen und Interaktion mit dem Benutzer variiert werden.

Ein Makro kann eine Leistung allein durch Ausführen von Makroanweisungen erbringen oder durch Zusammenstellen einer Kommandofolge (ggf. einschließlich Parameter), die anschließend automatisch ausgeführt wird.

In Makros werden die einzelnen Zeilen entweder als Makroanweisung oder als Daten angesehen. Zeilen, die mit zwei Dollarzeichen beginnen, werden jeweils als Makroanweisung interpretiert; alle anderen Zeilen werden als Daten angesehen. Mit den Makroanweisungen `MODE` (siehe ab Seite 413) können andere Konventionen zur Unterscheidung von Makroanweisungen und Daten festgelegt werden.

Zeilen mit Daten werden in die zusammenzustellende Kommandofolge bzw. (bei CGI-Makros) in die Standard-Ausgabe eingefügt, falls sie nicht auf Grund einer vorangehenden Makroanweisung übergangen werden oder eine besondere Bedeutung haben (z. B. Bestandteil der Definition einer Sternvariablen sind). Auf diese Weise zusammengestellte Kommandos werden anschließend automatisch ausgeführt; bei CGI-Makros wird die Standard-Ausgabe anschließend dem Betriebssystem übergeben.

Um Makros übersichtlicher zu gestalten, können in Makroanweisungen vor den beiden Dollarzeichen, die eine Zeile als Makroanweisung kennzeichnen, und an allen Stellen, an denen in der Beschreibung der Makroanweisung schon ein Leerzeichen steht, zusätzliche Leerzeichen eingefügt werden.

## Aufruf

Makros können mit dem Kommando `#MAKRO` (siehe Seite 132) ausgeführt werden; dabei können aber keine Spezifikationen an das Makro übergeben werden.

In der Regel werden Makros jedoch in der Form eines Kommandos (siehe »Allgemeines zu den Kommandos« Seite 110) aufgerufen und damit ausgeführt. Dabei stehen an Stelle des Kommandonamens ein bis drei Dollarzeichen und der Name des Makros:

```
#$Makroname, ...
```

Die Dollarzeichen unterscheiden Makroaufrufe von den anderen Kommandos. Im Gegensatz zu Kommandonamen dürfen Makronamen nicht abgekürzt werden. Ob und ggf. welche Spezifikationsangaben möglich sind, muss mit der Makroanweisung `$$!` (siehe Seite 413) angegeben sein.

Um ein Makro so aufrufen zu können, muss es als Segment in einer Segment-Datei (siehe Seite 35) abgespeichert sein. Der Name des Segments ist zugleich Name des Makros.

Der Name eines Makros kann aus 1 bis 12 Zeichen (Buchstaben, Ziffern und »\_«) bestehen, muss mit einem Buchstaben beginnen und darf nicht mit »\_« enden.

Bevor Makros aus einer Segment-Datei aufgerufen werden können, muss diese mit dem Kommando #DEFINIERE (siehe Seite 132) als »Makro-Datei« (vgl. Seite 36) definiert werden.

Beim Aufruf eines Makros wird das Makro in den mit dem Kommando #DEFINIERE definierten Makro-Dateien gesucht. In welcher Makro-Datei mit der Suche begonnen wird, hängt davon ab, wieviele Dollarzeichen im Makroaufruf angegeben sind: Ist eines angegeben, wird mit der ersten begonnen, sind zwei angegeben, wird mit der zweiten begonnen, und sind drei angegeben, wird mit der dritten begonnen. Wurde ein Makro noch nicht gefunden, nachdem die dritte (bzw. die letzte, wenn weniger Makro-Dateien definiert sind) Makro-Datei durchsucht wurde, wird die Suche beendet.

Wurde das Makro gefunden, wird überprüft, ob es auch (in der Regel auf Grund einer entsprechenden Hole-Anweisung) in der Datei steht, in der zuletzt mit dem Editor ein Segment einer Segment-Datei bearbeitet wurde. Ist dies der Fall, so wird das Makro aus dieser Datei gelesen, andernfalls aus der Makro-Datei. Dadurch wird erreicht, dass zum Testen eines Makros dieses nicht immer wieder (mit der Rette-Anweisung) in die Makro-Datei zurückkopiert werden muss.

In Einzelfällen kann es sinnvoll sein, ein Makro aus einer bestimmten Datei unter Umgehen der oben genannten Regeln auszuführen. Hierfür kann die Datei beim Aufruf durch ein Dollarzeichen getrennt nach dem Makronamen angegeben werden:

```
#$Makroname$Dateiname, ...
```

Die Datei muss in diesem Fall eine Segment-Datei sein, aber nicht als Makro-Datei definiert sein. Enthält eine Datei, die keine Segment-Datei ist, ein Makro, so kann beim Aufruf an Stelle des Segmentnamens ein Fragezeichen angegeben werden:

```
#$?$Dateiname, ...
```

Die Datei, die das Makro enthält, muss angemeldet sein.

Makros können auch mit Hilfe des Makros \*M aufgerufen werden. Wird dieses Makro ohne Spezifikationsangaben aufgerufen, so wird folgende Eingabemaske angezeigt:

TUSTEP-Datei-Manager		Aufrufen eines Makros		Beenden	
Name	<input type="text"/>	Auswählen	*	<<	>>
					X
Datei	anmelden	Beschreibung	einblenden	Makro	aufrufen merken

Um ein Makro auszuwählen, gibt es drei Möglichkeiten:



- Den Namen des aufzurufenden Makros (mit »\*« bzw. »\$«, falls ohne nicht eindeutig) in das Feld oben links eingeben und mit der Return-Taste bestätigen.
- Auf die Schaltfläche »Auswählen« klicken und damit ein Menü öffnen, aus dem eine Makro-Datei ausgewählt werden kann. Danach wird eine Liste mit den in der ausgewählten Datei enthaltenen Makros (genauer: Segmente) angezeigt, aus der das aufzurufende Makro ausgewählt werden kann. Das Auswählen kann jeweils dadurch erfolgen, dass die entsprechende Zeile mit der Maus angeklickt wird, oder, dass der Cursor mit den Pfeiltasten in die entsprechende Zeile positioniert und mit der Return-Taste die Auswahl bestätigt wird.
- Beim Aufruf des Makros »\*M« zur Spezifikation MAKRO den Namen (mit »\*« bzw. »\$«, falls ohne nicht eindeutig) des aufzurufenden Makros angeben. Falls der Name des Makros nicht mit »\*« beginnt bzw. das Makro in keiner der mit dem Kommando #DEFINIERE (siehe Seite 132) definierten Makro-Datei steht, muss zur Spezifikation DATEI auch der Name der Makro-Datei angegeben werden, in dem das Makro steht.

Nachdem ein Makro ausgewählt wurde, werden die Spezifikation und die Beschreibung dieses Makros in der Eingabemaske angezeigt:

TUSTEP-Datei-Manager
Aufrufen eines Makros
Beenden

Name

Auswählen
\*
<<
>>
X

QUELLE	=		▼	Name der Eingabedatei
ZIEL	=		▼	Name der Ausgabedatei
MODUS	=	-STD-	▼	Nur -STD- vorgesehen
LOESCHEN	=	-	▼	Daten in Ausgabedateien löschen?
LISTE	=	+	▼	Liste der Tags ausgeben?
IGNORIEREN	=		▼	Tags und Attribute ignorieren?
ERSETZEN	=	-STD-	▼	Namen von Formatvorlagen ändern?
SATZLAENGE	=	-STD-	▼	Lange Datensätze aufteilen?
KENNUNG	=	-STD-	▼	Beim Aufteilen Kennung einfügen?
OPTIONEN	=	-	▼	Zusätzliche Angaben
PROTOKOLL	=	+	▼	Protokolldatei / Ablaufprotokoll

Mit dem Makro \*IMPORT können Texte, die in RTF-Dateien oder in XML-Dateien im Format "Word 2003 XML-Dokument" (nicht "Word XML-Dokument") abgespeichert sind, nach TUSTEP importiert werden.

Das Makro \*IMPORT hat folgende Spezifikationen:

QUELLE	=	datei	Name der Quelldatei
ZIEL	=	datei	Name der Zieldatei
MODUS	=	-STD- *	(andere Angaben z.Z. noch nicht vorgesehen)
LOESCHEN	=	-	* Daten der ZIEL- und LISTE-Datei nicht löschen.
	=	+	* Daten der ZIEL- und LISTE-Datei zuerst löschen.
LISTE	=	+	* Liste der Tags ins Ablaufprotokoll ausgeben.

Datei anmelden
Beschreibung ausblenden
Makro aufrufen
merken

In der zweiten Spalte können nun die erforderlichen Spezifikationswerte eingetragene bzw. geändert werden. Falls das aufzurufende Makro entsprechende Informationen über die möglichen Spezifikationswerte enthält, kann alternativ in der dritten Spalte das Dreieck bzw. Fragezeichen angeklickt werden, um dann in einem Menü den gewünschten Spezifikationswert auszuwählen. Hinweise, in welcher Form diese

Informationen dazu im Makro vorhanden sein müssen, werden mit dem Kommando #INFORMIERE, M\_BEISPIEL ausgegeben.

Im unteren Teil der Eingabemaske wird die Beschreibung des ausgewählten Makros angezeigt. Wenn sie nicht vollständig in das dafür vorgesehene Feld passt, kann das Feld gescrollt werden.

In der rechten Hälfte der obersten Zeile und in der letzten Zeile befinden sich Schaltflächen. Eine Schaltfläche kann aktiviert werden, indem sie entweder mit der linken Maustaste angeklickt wird oder indem der Cursor mit der Tabulator-Taste auf die Schaltfläche positioniert und dann die Return-Taste (nicht Enter-Taste) gedrückt wird.

Muss vor dem Aufruf des Makros noch eine Datei angemeldet oder eingerichtet werden, kann dies mit Hilfe eines einfachen Dateimanagers erledigt werden, der durch Aktivieren der Schaltfläche »anmelden« aufgerufen wird.

Wenn das Makro mehr Spezifikationen hat, als angezeigt werden können, kann es sinnvoll sein, mit der Schaltfläche »ausblenden« das Feld mit der Beschreibung auszublenden.

Nachdem alle erforderlichen Spezifikationswerte in die Eingabemaske eingetragen sind, kann das Makro durch Aktivieren der Schaltfläche »aufrufen« gestartet werden; zuvor werden die Spezifikationswerte in der Datei \*TUSTEP.MCR gespeichert.

Wird die Schaltfläche »merken« aktiviert, so werden nur die Spezifikationswerte in der Datei \*TUSTEP.MCR gespeichert.

Wurden für ein Makro schon Spezifikationswerte gespeichert, können sie mit der Schaltfläche »<<<« in der oberen Zeile zurückgeholt werden. Wurden schon mehrmals Spezifikationswerte gespeichert kann mit der Schaltfläche »<<<« jeweils eine ältere Version und mit der Schaltfläche »>>>« jeweils eine neuere Version der Spezifikationswerte in die Eingabemaske geholt werden. Zwischen diesen beiden Schaltflächen wird jeweils das Datum und die Uhrzeit angezeigt, zu der die Spezifikationswerte gespeichert wurden.

Wird die Schaltfläche »X« rechts in der oberen Zeile aktiviert, so werden die zuletzt mit einer Schaltflächen »<<<« oder »>>>« geholten Spezifikationswerte aus der Liste der gemerkten Spezifikationswerte entfernt.

Wird die Schaltfläche »\*« in der Mitte der oberen Zeile aktiviert, so werden die für das Makro voreingestellten Spezifikationswerte in die Eingabemaske geholt.

## Makrovariablen

Makrovariablen stehen stellvertretend für jeweils eine Zeichenfolge. Diese Zeichenfolge wird »Wert der Variablen« oder auch »Inhalt der Variablen« genannt.

Jede Makrovariable hat einen Namen; er kann aus 1 bis 12 Zeichen (Buchstaben, Ziffern und »\_«) bestehen, muss mit einem Buchstaben beginnen und darf nicht mit »\_« enden.

An allen Stellen, an denen in einem Makro der Name einer Makrovariablen in spitzen

Klammern steht, wird dieser (einschließlich der Klammern) durch den aktuellen Wert der Makrovariablen ersetzt. Mit der Makroanweisung `$$=` (siehe Seite 417) können andere Klammern (z. B. die geschweiften Klammern) zur Kennzeichnung der zu ersetzenden Variablennamen festgelegt werden.

Da Makroanweisungen erst interpretiert werden, nachdem die in spitzen Klammern stehenden Namen von Makrovariablen ersetzt sind, kann es vorkommen, dass eine korrekt formulierte Makroanweisung nicht mehr entsprechend interpretiert werden kann. Ein Beispiel hierfür ist im Abschnitt »Interpretationsmodi einstellen« auf Seite 416 angegeben.

Bei manchen Makroanweisungen wird der Inhalt einer Variablen in einzelne Teilzeichenfolgen aufgeteilt. Als Trennzeichen dient dabei der Apostroph. Teilzeichenfolgen können auch leer sein (d. h. aus null Zeichen bestehen). Enthält eine Variable z. B. nur einen Apostroph, so sind dies zwei leere Teilzeichenfolgen; enthält sie keine Zeichen, so sind das null Teilzeichenfolgen. Mit der Makroanweisung `$$=` (siehe Seite 417) kann ein anderes Sonderzeichen (z. B. der Schrägstrich) als Trennzeichen festgelegt werden.

Eine Makrovariable wird dadurch definiert, dass ihr mit einer Makroanweisung ein Wert (eine Zeichenfolge) zugewiesen wird. Sie kann (nachdem sie definiert ist) innerhalb des Makros verwendet werden und gilt jeweils nur solange als definiert, bis sie mit der Makroanweisung `UNSET` (siehe Seite 419) freigegeben wird bzw. bis das Makro abgearbeitet ist. Beim nächsten Aufruf eines Makros kann nicht mehr auf ihren Inhalt zurückgegriffen werden.

Es gibt jedoch außerhalb von Makros ebenfalls Variablen. Sie werden TUSTEP-Variablen genannt und können mit dem Kommando `#DEFINIERE` (siehe Seite 132) oder mit der Makroanweisung `DEFINE` (siehe Seite 422) definiert werden. Sie gelten jeweils, bis sie umdefiniert werden bzw. bis zum Ende der TUSTEP-Sitzung. Auf den Inhalt einer solchen TUSTEP-Variablen kann in Makros mit der Makroanweisung `FETCH` (siehe Seite 422) zugegriffen werden.

Eine spezielle Art von Makrovariablen sind die »Sternvariablen«. Sie stehen stellvertretend für null, eine oder mehrere Zeichenfolgen, wobei jede dieser Zeichenfolgen in einer eigenen Zeile steht.

In den Zeilen, die eine Makroanweisung enthalten (also mit `$$` beginnen), dürfen jeweils mehrere in spitzen Klammern eingeschlossene Sternvariablen vorkommen; die Namen der Sternvariablen (einschließlich der Klammern) werden jedoch nicht durch den Inhalt der Variablen, sondern durch `»*«` ersetzt.

In den Zeilen, die Daten enthalten (also nicht mit `$$` beginnen), darf jeweils nur eine einzige in spitzen Klammern eingeschlossene Sternvariable vorkommen; eine solche Zeile wird so oft eingefügt, wie die Sternvariable Zeichenfolgen/Zeilen umfasst; dabei wird der Name der Sternvariablen (einschließlich der Klammern) jeweils durch eine Zeichenfolge/Zeile der Sternvariablen ersetzt.

Beispiel:

Es seien folgende Makrovariablen definiert:

1. Die Variable »Name« enthalte die Zeichenfolge »PAR«.
2. Die Variable »Nummer« enthalte den Wert 2.
3. Die Sternvariable »Wörter« enthalte folgende drei Zeilen:

```
aber
oder
und
```

Dann ergibt die Zeile

```
Beispiel: <Name> <Nummer> /<Wörter>/
```

nach dem Ersetzen der Variablen folgende drei Zeilen:

```
Beispiel: PAR 2 /aber/
Beispiel: PAR 2 /oder/
Beispiel: PAR 2 /und/
```

## Makroanweisungen

In jedem Makro, das in Form eines Kommandos aufgerufen werden soll, müssen am Anfang mit der nachfolgend beschriebenen Makroanweisung die Spezifikationen für das Kommando definiert werden. Diese Anweisung muss auch angegeben werden, wenn das Kommando keine Spezifikationen haben soll. Sie ist die einzige obligate Makroanweisung.

### Spezifikationen

Variablen können beim Aufruf des Makros definiert und mit aktuellen Werten belegt werden. Dazu müssen die entsprechenden Variablen mit der folgenden Makroanweisung als Spezifikationen des Kommandos definiert werden.

```
$$! spezifikationsname1, spezifikationsname2, ...
```

Die Namen der so definierten Spezifikationen sind zugleich die Namen der Variablen. Werden beim Aufruf des Makros Spezifikationswerte angegeben, so werden diese Werte den entsprechenden Variablen zugewiesen. Wird beim Aufruf des Makros zu einer Spezifikation kein Wert angegeben, so wird der entsprechenden Variablen eine leere Zeichenfolge zugewiesen. Soll in diesem Fall statt der leeren Zeichenfolge eine andere Zeichenfolge zugewiesen werden, so kann dies durch eine Ergänzung der obigen Makroanweisung geschehen:

```
$$! spezifikationsname1=spezifikationswert1, ...
```

Auf diese Weise können Voreinstellungen für die Spezifikationen definiert werden.

Die Definition der Spezifikationen kann auf mehrere Zeilen aufgeteilt werden. Diese müssen jedoch unmittelbar aufeinander folgen; die Angaben zu einer Spezifikation müssen jeweils zusammen in einer Zeile stehen, dürfen also nicht aufgeteilt werden.

```
$$! spezifikationsname1, spezifikationsname2,  
$$! spezifikationsname3
```

Die Definition der Spezifikationen muss am Anfang des Makros erfolgen. Davor dürfen nur mit »\$\$-« gekennzeichnete Kommentarzeilen (siehe Seite 431) stehen.

### Interpretationsmodi einstellen

Die im folgenden beschriebene `MODE`-Anweisung wird immer sofort ausgeführt (auch wenn sie z. B. zwischen `IF` und `ENDIF` steht und die Bedingung nicht erfüllt ist); der angegebene Modus gilt jeweils für alle nachfolgenden Anweisungen im jeweiligen Makro bzw. im jeweiligen Segment.

Ein erster Interpretationsmodus legt fest, wie Zeilen mit Makroanweisungen von Zeilen mit Daten unterschieden werden. Um die Zeilen unterscheiden zu können, müssen entweder die Anweisungszeilen mit zwei Dollarzeichen oder die Datenzeilen mit der Zeichenfolge `DATA` beginnen (d. h. gekennzeichnet sein). Voreingestellt ist das Kennzeichnen der Anweisungszeilen mit zwei Dollarzeichen; alle anderen Zeilen

werden als Datenzeilen angesehen. Bei umfangreichen Makros ist es oft günstiger, wenn auf die Kennzeichnung der Makroanweisungen verzichtet wird, und stattdessen die Daten gekennzeichnet werden.

Mit der Anweisung

```
$$ MODE STATEMENT
```

kann eingestellt werden, dass nicht gekennzeichnete Zeilen als Makroanweisungen angesehen werden. Datenzeilen müssen in diesem Fall mit `DATA` gekennzeichnet sein:

```
DATA beliebige Daten
```

Folgen auf das Schlüsselwort `DATA` mehrere Leerzeichen, so wird nur das erste als Trennzeichen zwischen Schlüsselwort und Daten gewertet; die restlichen Leerzeichen werden als Daten angesehen. Dies ist insbesondere bei Daten zu beachten, die als Tabellen nach einer `BUILD`- oder `CHECK`-Anweisung (siehe ab Seite 440) folgen.

Mit der Anweisung

```
$$ MODE DATA
```

kann eingestellt werden, dass nicht gekennzeichnete Zeilen als Daten angesehen werden. Anweisungszeilen müssen in diesem Fall mit zwei Dollarzeichen gekennzeichnet sein.

Ein zweiter Interpretationsmodus legt fest, mit welcher Konvention Zeichen-, und Stringgruppen sowie Such-, Austausch- und Recherchier-Tabellen interpretiert werden:

```
$$ MODE {}
```

bzw.

```
$$ MODE <>
```

Aus Kompatibilitätsgründen ist die Konvention `»<>«` voreingestellt. Zur Kennzeichnung von Zeichen- und Stringgruppen, Häufigkeitsangaben, Verweisen usw. werden dabei die spitzen Klammern verwendet.

Für neue Projekte empfiehlt es sich, die Konvention `»{}«` zu verwenden, da bei ihr die Codierungen leichter zu merken und auch leichter zu lesen sind. Zur Kennzeichnung von Zeichen- und Stringgruppen, Häufigkeitsangaben, Verweisen usw. werden dabei die geschweiften Klammern verwendet.

Bei den Beispielen in der nachfolgenden Beschreibung wird davon ausgegangen, dass die Konvention `»{}«` mit der `MODE`-Anweisung eingestellt wurde.

Ein dritter Interpretationsmodus gibt an, unter welchem Betriebssystem Farbwerte bestimmt wurden:

```
$$ MODE WINDOWS
```

bzw.

```
$$ MODE LINUX
```

bzw.

```
$$ MODE MAC
```

In TUSTEP können für den Hintergrund und für die Schrift jeweils 16 verschiedene Farben verwendet werden. Die Farben werden dazu hexadezimal von 0 bis F nummeriert. Betriebssystembedingt sind diese Farben unter Windows und Linux bzw. macOS nicht in der gleichen Reihenfolge angeordnet. Die Farben 1 und 4, 3 und 6, 9 und C sowie B und E sind jeweils vertauscht; z. B. ist unter Windows die Farbe 9 blau und die Farbe C rot, unter Linux und macOS ist die Farbe 9 rot und die Farbe C blau.

Werden Farbangaben in Makroanweisungen verwendet, so muss mit der `MODE`-Anweisung angegeben werden, unter welchem Betriebssystem die Farbwerte (mit den auf diesem System geltenden Farbnummern) bestimmt wurden, damit unter Windows und Linux bzw. macOS die Farben gleich dargestellt werden.

Ein vierter Interpretationsmodus legt fest, ob an bestimmten Stellen in Makroanweisungen ein Name als Schlüsselwort oder als Variable interpretiert werden soll. Er kann mit folgenden Makroanweisungen eingestellt werden:

```
$$ MODE KEYWORD
```

bzw.

```
$$ MODE VARIABLE
```

Achtung: Aus Kompatibilitätsgründen ist der Modus `KEYWORD` voreingestellt.

Bei Makros, die neu erstellt werden, sollten grundsätzlich die Modi `TUSCRIPT` und `{ }` eingestellt werden:

```
$$ MODE TUSCRIPT, { }
```

Damit werden die Modi `VARIABLE` und `STATEMENT` (s. o.), für die Steuerzeichen »Dollar«, »Klammern« und »Apostroph«, das Dollarzeichen, die geschweiften Klammern bzw. der Apostroph (siehe Kapitel »Steuerzeichen ändern« ab Seite 417) sowie der Modus `»{ }«` (s. o.) eingestellt.

Von der Einstellung `KEYWORD / VARIABLE` betroffen sind

- Wertzuweisungen: `$$ SET variable = name`

Folgt nach dem Gleichheitszeichen nur noch ein Name, so wird dieser im Modus `KEYWORD` als Name (Schlüsselwort) einer Makrofunktion, im Modus `VARIABLE` als Name einer Variablen interpretiert.

Soll im Modus `VARIABLE` der Name nach dem Gleichheitszeichen nicht als Name einer Variablen, sondern als Name einer Makrofunktion interpretiert werden, so muss hinter dem Namen `()` angegeben werden.

Bei Makrofunktionen, die keine Argumente haben, kann hinter dem Namen der Makrofunktion grundsätzlich `()` angegeben werden; in diesem Fall wird der Name unabhängig vom eingestellten Modus als Name einer Makrofunktion interpretiert.

- Bedingungen: `name1.EQ.name2`, `name1.NE.name2`, usw.

Steht links und/oder rechts von `.EQ.`, `.NE.` usw. ein Name, so wird dieser im Modus `KEYWORD` als Schlüsselwort, im Modus `VARIABLE` als Name einer Variablen interpretiert.

Soll im Modus `VARIABLE` ein Name nicht als Variable, sondern als Schlüsselwort interpretiert werden, so muss der Name zwischen Apostrophen stehen.

Wenn z. B. geprüft werden soll, ob eine Variable einen legalen Dateinamen enthält, kann dies im Modus `VARIABLE` mit folgender Makroanweisung geschehen:

```
$$ IF (variable.EQ.'FILE_NAME') ...
```

Im Modus `KEYWORD` müsste die Anweisung so formuliert werden:

```
$$ IF ("<variable>".EQ.FILE_NAME) ...
```

Bei dieser Anweisung wird zunächst der Name der Variablen einschließlich der spitzen Klammern durch den Inhalt der entsprechenden Variablen ersetzt. Wenn die Variable den Wert `xdat` hat, ergibt sich:

```
$$ IF ("xdat".EQ.FILE_NAME) ...
```

Dann erst wird die Makroanweisung interpretiert und ausgeführt.

Wenn nun aber der Inhalt der Variablen ein oder mehrere Anführungszeichen enthält (z. B. `xy"datei`), führt dieses Vorgehen zu einer unzulässigen Makroanweisung, obwohl die Anweisung korrekt formuliert wurde:

```
$$ IF ("xy"datei".EQ.FILE_NAME) ...
```

In diesem Beispiel endet die zu prüfende Zeichenfolge beim zweiten Anführungszeichen. Danach folgt das Wort `datei`, das an dieser Stelle nicht vorgesehen ist.

Um zu vermeiden, dass korrekt geschriebenen Makroanweisungen zu illegalen Anweisungen führen, sollte der Modus `VARIABLE` eingestellt werden. Dann kann die Abfrage (wie oben schon angegeben) so formuliert werden:

```
$$ IF (variable.EQ.'FILE_NAME') ...
```

Bei dieser Makroanweisung muss nicht zuerst der Inhalt der Variablen eingesetzt werden; die Makroanweisung kann in der angegebenen Form interpretiert und ausgeführt werden.

Wäre jedoch zuvor nicht der Modus auf `VARIABLE` eingestellt worden, würde bei der so formulierten Anweisung der Name der Variablen als Schlüsselwort interpretiert; die Makroanweisung wäre dann entweder illegal oder hätte zumindest nicht die gewünschte Wirkung.

Würden im Modus `VARIABLE` vor und nach `FILE_NAME` der Apostroph nicht angegeben, so würde `FILE_NAME` als Name einer Variablen interpretiert; d. h. es würden die Inhalte von zwei Variablen auf Übereinstimmung geprüft, falls eine Variable mit dem Namen `FILE_NAME` definiert ist, oder es würde eine Fehlermeldung ausgegeben, dass die Variable noch nicht definiert sei.



## Steuerzeichen ändern

Die im folgenden beschriebene Anweisung zum Ändern der Steuerzeichen wird immer sofort ausgeführt (auch wenn sie z. B. zwischen `IF` und `ENDIF` steht und die Bedingung nicht erfüllt ist); die Steuerzeichen gelten jeweils für alle nachfolgenden Anweisungen im jeweiligen Makro bzw. im jeweiligen Segment.

Es gibt drei Steuerzeichen, die verändert werden können:

- »Dollar«: Mit ihm werden die Makroanweisungen gekennzeichnet,
- »Klammern«: Sie schließen den Namen einer Variablen ein, falls an der betreffenden Stelle der Inhalt der Variablen eingesetzt werden soll, bevor die Makroanweisung interpretiert wird,
- »Apostroph«: Er trennt die einzelnen Teilzeichenfolgen innerhalb einer Zeichenfolge.

Nach der Makroanweisung

```
$$= * {} /
```

werden der Stern als Steuerzeichen »Dollar«, die geschweiften Klammern als Steuerzeichen »Klammern« und der Schrägstrich als Steuerzeichen »Apostroph« interpretiert.

Als Steuerzeichen

- »Dollar« ist jedes Sonderzeichen außer dem Minuszeichen erlaubt,
- »Klammern« sind die eckigen, die runden, die spitzen und die geschweiften Klammern erlaubt,
- »Apostroph« ist jedes Sonderzeichen erlaubt.

Achtung: Mit der Makroanweisung

```
$$= -
```

wird aus Kompatibilitätsgründen nicht das Minuszeichen als Steuerzeichen »Dollar«, sondern der Modus `STATEMENT` eingestellt, in dem nicht gekennzeichnete Zeilen als Makroanweisungen angesehen werden und Datenzeilen mit `DATA` gekennzeichnet werden müssen (vgl. Kapitel »Interpretationsmodi einstellen« ab Seite 413).

In dieser Beschreibung der Makros wird davon ausgegangen, dass die Steuerzeichen nicht verändert werden. Wenn also z. B. in der Beschreibung »... mit zwei Dollarzeichen beginnen ...« steht, so muss dafür »... mit zwei Klammeraffen beginnen ...« gelesen werden, falls der Klammeraffe für das Steuerzeichen »Dollar« eingestellt worden ist. Entsprechendes gilt für die anderen Steuerzeichen.

Nach dem Verändern der Steuerzeichen mit obiger Anweisung können die voreingestellten Steuerzeichen mit folgender Anweisung wieder eingestellt werden:

```
**= $ <> '
```

Sollen die Steuerzeichen »Dollar« und/oder »Apostroph« nicht verändert werden, können sie weggelassen werden; soll nur das Steuerzeichen »Dollar« verändert werden, genügt die Angabe des neuen Steuerzeichens für »Dollar«.

Für Makros zur Verarbeitung von Daten, die Tags enthalten (insbesondere auch für

CGI-Makros), ist es meist sinnvoll, als »Klammern« die geschweiften Klammern einzustellen. Sonst würde ein im Makro stehendes Tag der Form »<name>« entweder durch den Inhalt einer Variablen ersetzt, falls eine gleichnamige Makrovariable definiert ist, oder es würde eine Fehlermeldung ausgegeben, dass die Variable noch nicht definiert sei.

## Definieren von Variablen – Wertzuweisungen

Eine Variable kann auch mit einer der folgenden Makroanweisungen definiert und mit einem Wert belegt werden:

```
$$ SET variablenname = "Zeichenfolge"
```

Die zwischen den Anführungszeichen stehende Zeichenfolge wird der angegebenen Variablen zugewiesen. Die angegebene Zeichenfolge darf keine Anführungszeichen enthalten.

```
$$ SET variablenname = "Anfang der Zeichenfolge" ...
$$                      "Fortsetzung der Zeichenfolge" ...
$$                      "Schluss der Zeichenfolge"
```

Damit auch Wertzuweisungen mit langen Zeichenfolgen übersichtlich bleiben, kann die Zeichenfolge auf mehrere Zeilen aufgeteilt werden. In diesem Fall muss jede Zeichenfolge in Anführungszeichen eingeschlossen werden. Außerdem müssen als Kennzeichnung, dass die Zeichenfolge in der nächsten Zeile fortgesetzt wird, drei Punkte am Zeilenende eingefügt werden.

```
$$ SET variablenname = Rechenanweisung
```

Die Rechenanweisung wird ausgeführt und das Ergebnis (ein Zahlenwert) der angegebenen Variablen zugewiesen. Die möglichen Rechenanweisungen sind unten ab Seite 477 beschrieben.

```
$$ SET variablenname = Makrofunktion
```

Die Makrofunktion wird ausgeführt und der Funktionswert der angegebenen Variablen zugewiesen. Die möglichen Makrofunktionen sind unten ab Seite 479 beschrieben.

```
$$ SET variablenname1 = variablenname2
```

Wurde mit der `MODE`-Anweisung (siehe ab Seite 413) der Modus `VARIABLE` oder `TUSCRIPT` eingestellt, kann mit dieser Makroanweisung der Inhalt der rechts vom Gleichheitszeichen stehenden Variablen unverändert der links vom Gleichheitszeichen stehenden zugewiesen werden. Ist jedoch zuvor nicht der Modus auf `VARIABLE` oder `TUSCRIPT` eingestellt worden, so wird der Name rechts vom Gleichheitszeichen als Schlüsselwort (hier als Name einer Makrofunktion) interpretiert; die Makroanweisung ist dann entweder illegal oder hat zumindest nicht die gewünschte Wirkung.

```
$$ SET var1 = var2 = var3 = ...
```

Soll mehreren Variablen derselbe Wert zugewiesen werden, kann dies mit einer einzigen `SET`-Anweisung geschehen; alle Variablen erhalten den Wert, der durch die Angabe nach dem letzten Gleichheitszeichen bestimmt wird.

```
$$ SET var1 = "Zflg", var2 = Rechenanw, var3 = Makrofunktion
```

Folgen mehrere der oben beschriebenen SET-Anweisungen unmittelbar aufeinander, können diese in einer Zeile zusammengefasst werden, wobei die einzelnen Wertzuweisungen durch Komma getrennt werden müssen. Dabei ist zu beachten, dass zuerst alle in spitzen Klammern stehenden Variablen durch ihren Wert ersetzt und dann erst die Wertzuweisungen vorgenommen werden. Die nachfolgend beschriebene SET-Anweisung darf mit keiner anderen zusammengefasst werden.

```
$$ SET variablenname = *
```

Mit dieser Makroanweisung wird eine Sternvariable definiert. Der Variablen werden die Zeilen zugewiesen, die zwischen dieser Makroanweisung und der auf sie folgenden Makroanweisung (= nächste Zeile, die mit \$\$ beginnt) stehen.

```
$$ SET/DATA variablenname
...
$$ ENDDATA
```

Mit diesen Makroanweisungen wird ebenfalls eine Sternvariable definiert. Der Variablen werden die Zeilen zugewiesen, die zwischen der Makroanweisung SET/DATA und der auf sie folgenden Makroanweisung ENDDATA stehen und keine Makroanweisungen sind (d. h. nicht mit \$\$ beginnen). Damit ist es möglich, z. B. mit IF-Anweisungen eine Auswahl der Zeilen zu treffen, die der Variablen zugewiesen werden sollen.

```
$$ SET @variablenname = ...
```

Wird in einer SET-Anweisung die links von einem Gleichheitszeichen stehende Variable mit »@« markiert, erfolgt die Wertzuweisung nicht auf die angegebene Variable, sondern auf die Variable, deren Namen in der angegebenen Variablen steht.

Submakros (siehe Seite 451) haben »eigene« Variablen. In Submakros sind nur diejenigen Variablen bekannt, die innerhalb des jeweiligen Submakros definiert werden. Ein Ausnahme bilden die mit der folgenden Makroanweisung zuvor definierten Variablen:

```
$$ DEFINE/Common variable = ...
```

Variablen, die so definiert wurden, sind danach auch in Submakros bekannt. Sie können wie jede andere Variable verwendet und auch mit einem anderen Wert belegt werden.

## Freigabe von Variablen

Wenn eine Variable nicht mehr benötigt wird, kann sie mit der folgenden Makroanweisung freigegeben werden:

```
$$ UNSET/VARIABLE variablenname
```

Es können auch mehrere Variablenamen durch Komma getrennt angegeben werden. Die Option VARIABLE ist nicht obligat; sie kann angegeben werden, um die Aufgabe der Anweisung zu verdeutlichen.

```
$$ UNSET/STRUCTURE strukturname
```

Mit dieser Makroanweisung werden alle Variablen freigegeben, die in der angegebenen Struktur (vgl. Seite 627) aufgeführt sind. Es können auch mehrere Strukturnamen durch Komma getrennt angegeben werden.

Beim Beenden eines Makros werden Variablen, die noch nicht freigegeben sind, automatisch freigegeben. Die Freigabe mit der Anweisung `UNSET` ist insbesondere für Variablen sinnvoll, die umfangreiche Daten enthalten, damit der von ihnen belegte Platz wieder für andere Variablen zur Verfügung steht.

## Definieren von Trägerangaben

Beim Einrichten und Anmelden von Dateien sowie bei einigen Dateien betreffende Abfragen muss ein Träger (siehe Kapitel »Projekt und Träger« ab Seite 25) angegeben werden. Als Trägerangabe wird i.d.R. der Name einer System-Variablen, die einen Pfad enthält, erwartet. An Stelle des Namens einer System-Variablen kann als Träger jeweils auch ein mit der folgenden Anweisung definierter Name angegeben werden, ohne dass in der Beschreibung explizit darauf hingewiesen wird:

```
$$ DEFINE/VOLUME trägername = pfad
```

Die Variable `pfad` muss eine Pfadangabe enthalten. An Stelle dieser Variablen kann auch eine in Anführungszeichen eingeschlossene Pfadangabe angegeben werden.

Es können bis zu vier Trägernamen gleichzeitig definiert sein.

Mit der folgenden Makroanweisung kann die Definition eines Trägernamens wieder aufgehoben werden:

```
$$ REMOVE/VOLUME trägername
```

Beim Beenden eines Makros werden Trägernamen, deren Definition noch nicht aufgehoben wurde, automatisch aufgehoben.

## Zugriff auf Makrovariablen

Der Zugriff auf eine Makrovariable erfolgt im Regelfall direkt, indem der Name der Variablen angegeben wird. Es ist auch möglich, auf eine Variable indirekt zuzugreifen, indem der Name einer Variablen angegeben wird, die den Namen derjenigen Variablen enthält, auf die zugegriffen werden soll. Der Name der angegebenen Variablen muss in diesem Fall mit einem Klammeraffen unmittelbar vor dem Namen gekennzeichnet sein.

Beispiel: Statt der Anweisung

```
$$ SET neu = EXCHANGE (alt, ":x:y:")
```

könnten auch die Anweisungen

```
$$ SET name = "alt"
...
$$ SET neu = EXCHANGE (@name, ":x:y:")
```

oder die Anweisungen

```

$$ SET name = "neu"
...
$$ SET @name = EXCHANGE (alt, ":x:y:")

```

verwendet werden. Das Ergebnis ist bei allen dasselbe.

Wird in einer Anweisung eine Zeichenfolge erwartet, so kann sie in Anführungszeichen eingeschlossen direkt angegeben werden. In der Regel kann alternativ der Name einer Variablen angegeben werden, die diese Zeichenfolge enthält. Es gibt jedoch einige wenige Anweisungen, bei denen dies nicht möglich ist. Zum Beispiel wird mit der Anweisung

```

$$ DEFINE "name" = pfad

```

ein Alias-Name für eine Datei definiert (siehe Seite 423). Würde die Anweisung durch die Anweisungen

```

$$ SET alias = "name"
$$ DEFINE alias = pfad

```

ersetzt, so würde damit eine TUSTEP-Variable mit dem Namen »alias« definiert (siehe Seite 422). Deshalb kann an Stellen, an denen eine in Anführungszeichen eingeschlossene Zeichenfolge erwartet wird, gleichbedeutend auch eine mit einem Dollarzeichen unmittelbar vor dem Namen gekennzeichnete Variable angegeben werden. Auf diese Weise kann mit den Anweisungen

```

$$ SET alias = "name"
$$ DEFINE $alias = pfad

```

auch ein Alias-Name für eine Datei definiert werden, wenn der Alias-Name in einer Variablen steht.

## Zugriff auf Tabellen

Der Zugriff auf eine mit der BUILD-Anweisung (siehe Seite 440) definierte Such-, Austausch- oder Recherchier-Tabelle erfolgt im Regelfall direkt, indem der Name der Tabelle angegeben wird. Es ist auch möglich, auf eine Tabelle indirekt zuzugreifen, indem der Name einer Variablen angegeben wird, die den Namen der Tabelle enthält. Der Name der angegebenen Variablen muss in diesem Fall mit einem Klammeraffen unmittelbar vor dem Namen gekennzeichnet sein.

Beispiel: Nachdem mit der Anweisung

```

$$ BUILD X_TABLE x2y = ":x:y:"

```

eine Austausch-Tabelle definiert wurde, liefern die Anweisungen

```

$$ SET neu = EXCHANGE (alt, x2y)

```

und

```

$$ SET tausche = "x2y"
$$ ...
$$ SET neu = EXCHANGE (alt, @tausche)

```

dasselbe Ergebnis wie die Anweisung

```
$$ SET neu = EXCHANGE (alt, ":x:y:")
```

Im letzten Beispiel wird die Austausch-Tabelle als Zeichenfolge in Anführungszeichen angegeben. Würde statt der Zeichenfolge der Name einer Variablen angegeben, die diese Zeichenfolge enthält, so würde dieser Name nicht als Name einer Variablen, sondern als Name einer mit der BUILD-Anweisung definierten Austausch-Tabelle interpretiert. Damit der Name als Name einer Variablen interpretiert wird, muss er mit einem Dollarzeichen unmittelbar vor dem Namen gekennzeichnet werden:

```
$$ SET tausche = ":x:y:"
$$ SET neu = EXCHANGE (alt, $tausche)
```

## Zugriff auf TUSTEP-Variablen

TUSTEP-Variablen können mit dem Kommando #DEFINIERE (siehe Seite 132) und mit der Makroanweisung DEFINE definiert werden.

```
$$ DEFINE variablenname
```

Definiert eine TUSTEP-Variable mit dem angegebenen Namen und weist ihr den Wert (Inhalt) der gleichnamigen Makrovariablen zu. Wird der Inhalt der Makrovariablen anschließend wieder verändert, so hat dies keinen Einfluss mehr auf die gleichnamige TUSTEP-Variable.

```
$$ DEFINE tustepvariablenname = "Zeichenfolge"
```

Definiert eine TUSTEP-Variable mit dem angegebenen Namen und weist ihr die zwischen den Anführungszeichen stehende Zeichenfolge zu.

```
$$ DEFINE/TUSTEP tustepvariablenname = makrovariablenname
```

Definiert eine TUSTEP-Variable mit dem angegebenen Namen und weist ihr den Wert der Makrovariablen zu.

Bei der dritten Form der DEFINE-Anweisung ist die Angabe der Option TUSTEP erforderlich, da sonst eine System-Variable definiert würde. Bei den ersten beiden Formen kann die Option TUSTEP angegeben werden, um die Aufgabe der Anweisung zu verdeutlichen.

```
$$ FETCH variablenname
```

Mit dieser Makroanweisung wird eine Makrovariable mit dem angegebenen Namen definiert. Wenn eine gleichnamige TUSTEP-Variable definiert ist, wird der Makrovariablen der Wert (Inhalt) dieser TUSTEP-Variablen zugewiesen; wenn nicht, wird der Makrovariablen eine leere Zeichenfolge zugewiesen.

```
$$ FETCH variablenname = "Zeichenfolge"
```

Mit dieser Makroanweisung wird eine Makrovariable mit dem angegebenen Namen definiert. Wenn eine gleichnamige TUSTEP-Variable definiert ist, wird der Makrovariablen der Wert (Inhalt) dieser TUSTEP-Variablen zugewiesen; wenn nicht, wird der Makrovariablen die zwischen den Anführungszeichen stehende Zeichenfolge zugewiesen.

```
$$ FETCH/TUSTEP makrovariablenname = tustepvariablenname
```

Mit dieser Makroanweisung wird eine Makrovariable mit dem angegebenen Namen definiert. Wenn die angegebene TUSTEP-Variable definiert ist, wird der Makrovariablen der Wert dieser TUSTEP-Variablen zugewiesen; wenn nicht, wird der Makrovariablen eine leere Zeichenfolge zugewiesen.

Bei der dritten Form der FETCH-Anweisung ist die Angabe der Option TUSTEP erforderlich, da sonst eine System-Variable gesucht würde. Bei den ersten beiden Formen kann die Option TUSTEP angegeben werden, um die Aufgabe der Anweisung zu verdeutlichen.

```
$$ REMOVE variablenname
```

Mit dieser Makroanweisung wird die Definition der TUSTEP-Variable mit dem angegebenen Namen gelöscht. Die Option TUSTEP kann wie bei den obigen Anweisungen angegeben werden, um die Aufgabe der Anweisung zu verdeutlichen.

## Zugriff auf definierte Dateinamen

Ein Dateiname kann mit dem Kommando #DEFINIERE (siehe Seite 132) und mit der Makroanweisung DEFINE definiert werden (siehe »Definierte Dateinamen« Seite 29).

Hinweis: Beim Einrichten bzw. Anmelden einer Datei mit einem definierten Dateinamen muss als Träger ein Minuszeichen angegeben werden.

Mit »Dateibezeichnung« ist im folgenden die auf Betriebssystemebene erforderliche Bezeichnung (Pfad mit Dateiname) für eine Datei gemeint.

```
$$ DEFINE "dateiname" = "dateibezeichnung"
```

Definiert den angegebenen Dateinamen und ordnet ihm die angegebene Dateibezeichnung zu.

```
$$ DEFINE "dateiname" = variablenname
```

Definiert den angegebenen Dateinamen und ordnet ihm die in der angegebene Variablen enthaltene Dateibezeichnung zu.

Bei den beiden vorangehend aufgeführten Anweisungen kann an Stelle des in Anführungszeichen angegebenen Dateinamens auch der Name einer Variablen angegeben werden, die den Dateinamen enthält. Der Name der Variablen muss in diesem Fall mit einem Dollarzeichen unmittelbar vor dem Namen gekennzeichnet sein; ohne Dollarzeichen würde eine TUSTEP-Variable mit diesem Namen definiert (vgl. Seite 422).

```
$$ FETCH "dateiname" variablenname = "Zeichenfolge"
```

Mit dieser Makroanweisung wird eine Makrovariable mit dem angegebenen Namen definiert. Wenn der angegebene Dateiname mit einem »definierten Dateinamen« übereinstimmt, wird der Makrovariablen die dazugehörige Dateibezeichnung zugewiesen; wenn nicht, wird der Makrovariablen die zwischen den Anführungszeichen stehende Zeichenfolge zugewiesen.

```
$$ FETCH "dateiname" variablenname1 = variablenname2
```

Mit dieser Makroanweisung wird eine Makrovariable mit dem links vom Gleichheitszeichen stehenden Variablennamen definiert. Wenn der angegebene Dateiname mit einem »definierten Dateinamen« übereinstimmt, wird dieser Variablen die dazugehörige Datei-Bezeichnung zugewiesen; wenn nicht, wird dieser Variablen der Wert der rechts vom Gleichheitszeichen stehenden Variablen zugewiesen.

Bei den beiden vorangehend aufgeführten Anweisungen kann an Stelle des in Anführungszeichen angegebenen Dateinamens auch der Name einer Variablen angegeben werden, die den Dateinamen enthält. Der Name der Variablen muss in diesem Fall mit einem Dollarzeichen unmittelbar vor dem Namen gekennzeichnet sein; ohne Dollarzeichen wäre die Anweisung syntaktisch falsch.

```
$$ REMOVE "dateiname"
```

Mit dieser Makroanweisung wird die Definition des angegebenen Dateinamens gelöscht.

## Zugriff auf System-Variablen

System-Variablen können mit dem Makro \*DESI für jede einzelne TUSTEP-Sitzung vorgegeben werden. Sie werden dann bei nachfolgenden Aufrufen (Initialisierung oder Fortsetzung) der jeweiligen Sitzung automatisch definiert.

System-Variablen können auch mit der Makroanweisung DEFINE definiert werden. Diese Definitionen gelten jedoch nur für den jeweiligen TUSTEP-Aufruf. Bei nachfolgenden Aufrufen sind die mit der Makroanweisung DEFINE definierten System-Variablen betriebssystembedingt nicht mehr bekannt. Dies gilt auch, wenn eine Sitzung unterbrochen und wieder fortgesetzt wird.

```
$$ DEFINE/SYSTEM variablenname
```

definiert eine System-Variable mit dem angegebenen Namen und weist ihr den Wert (Inhalt) der gleichnamigen Makrovariablen zu. Wird der Inhalt der Makrovariablen anschließend wieder verändert, so hat dies keinen Einfluss mehr auf die gleichnamige System-Variable.

```
$$ DEFINE/SYSTEM systemvariablenname = "Zeichenfolge"
```

definiert eine System-Variable mit dem angegebenen Namen und weist ihr die zwischen den Anführungszeichen stehende Zeichenfolge zu.

```
$$ DEFINE systemvariablenname = makrovariablenname
```

definiert eine System-Variable mit dem angegebenen Namen und weist ihr den Wert der Makrovariablen zu.

Bei den ersten beiden Formen der DEFINE-Anweisung ist die Angabe der Option SYSTEM erforderlich, da sonst eine TUSTEP-Variable definiert würde. Bei der dritten Form kann die Option SYSTEM angegeben werden, um die Aufgabe der Anweisung zu verdeutlichen.

```
$$ FETCH/SYSTEM variablenname
```

Mit dieser Makroanweisung wird eine Makrovariable mit dem angegebenen Namen



definiert. Wenn eine gleichnamige System-Variable definiert ist, wird der Makrovariablen der Wert (Inhalt) dieser System-Variablen zugewiesen; wenn nicht, wird der Makrovariablen eine leere Zeichenfolge zugewiesen.

```
$$ FETCH/SYSTEM variablenname = "Zeichenfolge"
```

Mit dieser Makroanweisung wird eine Makrovariable mit dem angegebenen Namen definiert. Wenn eine gleichnamige System-Variable definiert ist, wird der Makrovariablen der Wert (Inhalt) dieser System-Variablen zugewiesen; wenn nicht, wird der Makrovariablen die zwischen den Anführungszeichen stehende Zeichenfolge zugewiesen.

```
$$ FETCH makrovariablenname = systemvariablenname
```

Mit dieser Makroanweisung wird eine Makrovariable mit dem angegebenen Namen definiert. Wenn die angegebene System-Variable definiert ist, wird der Makrovariablen der Wert dieser System-Variablen zugewiesen; wenn nicht, wird der Makrovariablen eine leere Zeichenfolge zugewiesen.

Bei den ersten beiden Formen der `FETCH`-Anweisung ist die Angabe der Option `SYSTEM` erforderlich, da sonst eine `TUSTEP`-Variable gesucht würde. Bei der dritten Form kann die Option `SYSTEM` angegeben werden, um die Aufgabe der Anweisung zu verdeutlichen.

```
$$ REMOVE/SYSTEM variablenname
```

Unter Windows kann mit dieser Makroanweisung die Definition der System-Variable mit dem angegebenen Namen gelöscht werden. Wurde die System-Variable nicht von `TUSTEP` definiert, so bleibt die Definition der System-Variablen außerhalb von `TUSTEP` erhalten. Die Option `SYSTEM` ist obligat, da sonst die `TUSTEP`-Variable mit dem angegebenen Namen gelöscht würde.

## Zugriff auf die Zwischenablage

Mit der folgenden Makroanweisung wird der Inhalt einer Variablen in die Zwischenablage kopiert:

```
$$ DEFINE/CLIPBOARD variablenname
```

Enthält die Zwischenablage schon Daten, werden diese zuvor gelöscht.

```
$$ FETCH/CLIPBOARD variablenname
```

Mit dieser Makroanweisung wird eine Sternvariable mit dem angegebenen Namen definiert und der Inhalt der Zwischenablage dieser Variablen zugewiesen.

## Zugriff auf Editor-Informationen

```
$$ DEFINE/EDIT_FILE "dateiname"
```

Mit dieser Makroanweisung wird die angegebene Datei als die zuletzt mit dem Editor editierte Datei definiert.

```
$$ FETCH/EDIT_FILE variablenname
```

Mit dieser Makroanweisung wird der Name der zuletzt mit dem Editor edierten Datei der angegebenen Variablen zugewiesen.

Hinweis: Die Makrofunktion `EDIT_INFO` (siehe Seite 494) liefert Informationen zum Inhalt einer TUSTEP-Datei.

## Anfragen

Mit der folgenden Makroanweisung wird eine Variable definiert und mit einem Wert belegt. Dabei wird im Dialog-Modus eine Meldung ausgegeben und eine Antwort erwartet:

```
$$ ASK "Meldung": variablenname = "Zeichenfolge"
```

Die Zeichenfolge, die als Antwort eingegeben wird (max. eine Zeile), wird der angegebenen Variablen zugewiesen. Falls eine leere Antwort eingegeben wird oder das Makro im Batch-Modus aufgerufen wird, wird der Variablen die nach dem Gleichheitszeichen zwischen den Anführungszeichen angegebene Zeichenfolge zugewiesen.

```
$$ ASK "Meldung": variablenname = "Zflg_1", "Zflg_2"
```

Werden nach dem Gleichheitszeichen zwei Zeichenfolgen durch Komma getrennt angegeben, wird die erste der Variablen zugewiesen, falls eine leere Antwort eingegeben wird, und die zweite, falls »\*EOF« eingegeben oder die Eingabe mit der Escape-Taste oder der Tastenkombination `Ctrl+D` bzw. `Strg+D` abgeschlossen wird.

Bei allen ASK-Anweisungen kann nach dem Schlüsselwort ASK eine Option angegeben werden, mit der die Farbe der Meldung festgelegt wird:

```
$$ ASK/WARNING "Meldung": variablenname = ...
```

gibt sie in der Form und Farbe einer Warnung aus;

```
$$ ASK/ERROR "Meldung": variablenname = ...
```

gibt sie in der Form und Farbe einer Fehlermeldung aus;

```
$$ ASK/COLOR:xx "Meldung": variablenname = ...
```

gibt sie in der mit dem Hexadecimal-Code angegebenen Farbe aus. Die möglichen Codes können der Tabelle entnommen werden, die im Editor durch die Tastenkombination `Ctrl+F` bzw. `Strg+F` angezeigt wird.

Außerdem kann bei allen ASK-Anweisungen ggf. zusätzlich die Option `SAVE` angegeben werden:

```
$$ ASK/SAVE "Meldung": variablenname = ...
```

nimmt die Antwort auf die Anfrage in die Liste der seither auf Kommandoebene eingegebenen Zeilen auf (siehe Seite 113).

## Dateneingabe im Dialog-Modus

Mit der folgenden Makroanweisung wird eine Sternvariable definiert; Daten, die hinter dem Aufruf des Makros stehen, werden eingelesen und der Sternvariablen, deren Name in der Makroanweisung angegeben ist, zugewiesen:

```
$$ ASK "Meldung": variablenname = *
```

Die Daten müssen mit »\*EOF« abgeschlossen werden. Bei jedem Abarbeiten einer solchen Makroanweisung werden jeweils die nachfolgenden Daten bis zum nächsten »\*EOF« eingelesen.

Falls hinter dem Aufruf des Makros keine Daten vorhanden sind, werden die Daten mit der in der Makroanweisung angegebenen Meldung angefordert.

## Dateneingabe im Batch-Modus

Mit der folgenden Makroanweisung wird eine Variable definiert; alle Daten der Standard-Eingabe werden eingelesen und der Variablen, deren Name in der Makroanweisung angegeben ist, zugewiesen:

```
$$ FETCH variablenname = -STD-
```

Enthält die Standard-Eingabe mehrere Zeilen, so werden alle Zeilen zu einer einzigen Zeichenfolge zusammengefasst und zwischen den Zeilen jeweils ein Leerzeichen ergänzt. Wurde das Makro im Dialog-Modus aufgerufen, wird nichts eingelesen; der angegebenen Variablen wird eine leere Zeichenfolge zugewiesen.

Mit der folgenden Makroanweisung wird eine Sternvariable definiert; alle Daten der Standard-Eingabe werden eingelesen und der Sternvariablen, deren Name in der Makroanweisung angegeben ist, zugewiesen:

```
$$ FETCH variablenname = *
```

Jede Zeile der Standard-Eingabe ergibt eine Zeile in der Sternvariablen. Wurde das Makro im Dialog-Modus aufgerufen, wird nichts eingelesen; der angegebenen Variablen werden 0 Zeilen zugewiesen.

## Dateneingabe von Dateien

Mit der Makrofunktion `FILE` (siehe Seite 494) können Daten von einer Datei eingelesen und in eine Variable gespeichert werden.

Zur Dateneingabe von Dateien stehen außerdem noch die ab Seite 597 beschriebenen »Dateizugriffe« mit vielfältigen Steuermöglichkeiten zur Verfügung.

## Datenausgabe in Dateien

Zur Datenausgabe in Dateien stehen neben der nachfolgend beschriebenen Makroanweisung noch die ab Seite 597 beschriebenen »Dateizugriffe« mit vielfältigen Steuermöglichkeiten zur Verfügung.

Mit der Makroanweisung

```
$$ FILE "dateiname" = variablenname
```

wird der Inhalt der angegebenen Variablen als nächster Satz ans Ende der angegebenen Datei geschrieben. Falls die Variable eine Sternvariable ist, wird für jede Zeile der Variablen ein Satz ans Dateiende geschrieben.

Hinter dem Schlüsselwort `FILE` können jeweils nach einem Schrägstrich noch Optionen in der Reihenfolge angegeben werden, in der sie im Folgenden beschrieben sind:

Wird die Option `ERASE` angegeben, so werden die schon in der Datei stehenden Daten zuerst gelöscht.

Wenn die angegebene Datei eine TUSTEP-Datei ist, werden die Sätze im Textmodus nummeriert. Sollen sie im Programmmodus nummeriert werden, kann die Option `PROGRAM` angegeben werden.

Wenn die angegebene Datei eine Fremd-Datei ist, kann als Option der Code angegeben werden, in den die Daten vom TUSTEP-Code umcodiert werden sollen; vorgesehen sind die Codes `ISO` (= ISO-8859-1), `UTF8` und `UTF16`. Voreingestellt ist `ISO`. Nach den Optionen `UTF8` und `UTF16` kann noch die Option `BOM` angegeben werden um die Byte-Order-Mark am Dateianfang zu unterdrücken. Wird statt eines Codes die Option `BINARY` angegeben, werden die Daten nicht umcodiert. Wird nach der Option `BINARY` noch die Option `NLF` angegeben, so werden nur die Daten und keine Codes für Zeilenwechsel (`CR+LF` unter Windows, `LF` unter Linux und macOS) in die Datei ausgegeben.

Wird die Option `PRINT` angegeben, so wird eine Meldung mit Angaben zu den ausgegebenen Daten ins Ablaufprotokoll ausgegeben.

Die obigen Formen der Makroanweisung `FILE` sollten nur verwendet werden, wenn der Inhalt einer einzigen Variablen in eine Datei ausgegeben werden soll; andernfalls sollte die nachfolgend unter »Ausgabe in eine Datei umleiten« beschriebene Form verwendet werden.

Mit der Makroanweisung

```
$$ FILE/TITLE "dateiname" = variablenname
```

wird der Inhalt der angegebenen Variablen der angegebenen TUSTEP-Datei als Dateititel zugewiesen.

Mit der Makroanweisung

```
$$ FILE/FILE "dateiname" = variablenname
```

wird der in der angegebenen Variablen stehende Dateiname (andere Daten sind nicht erlaubt) der angegebenen TUSTEP-Datei als der zuletzt im Editor eingestellter Dateiname zugewiesen.

### Mit der Makroanweisung

```
$$ FILE/SEGMENT "dateiname" = variablenname
```

wird der in der angegebenen Variablen stehende Segmentname (andere Daten sind nicht erlaubt) der angegebenen TUSTEP-Datei als der zuletzt im Editor eingestellter Segmentname zugewiesen.

### Mit der Makroanweisung

```
$$ SEGMENT/ERASE "dateiname", "segmentname" = variablenname
```

wird der Inhalt der angegebenen Variablen als Segment mit dem angegebenen Namen in die angegebene Datei geschrieben. Falls die Variable eine Sternvariable ist, wird für jede Zeile der Variablen ein Satz in das Segment geschrieben.

Die Option `ERASE` ist obligat; existiert in der Datei schon ein Segment mit dem angegebenen Namen, so wird dieses zuerst gelöscht. Wird hinter der Option `ERASE` nach einem Schrägstrich die Option `PRINT` angegeben, so wird eine Meldung mit Angaben zu den ausgegebenen Daten ins Ablaufprotokoll ausgegeben.

### Mit der Makroanweisung

```
$$ SEGMENT/TITLE "dateiname", "segmentname" = variablenname
```

wird der Inhalt der angegebenen Variablen dem angegebenen Segment als Titel zugewiesen.

Bei allen `SEGMENT`-Anweisungen kann an Stelle des Variablennamens auch eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben werden.

Die `FILE`- und die `SEGMENT`-Anweisung müssen auf die Datei schreibend zugreifen. Dies ist jedoch nicht möglich, wenn schon ein Programm diese Datei mit lesendem oder schreibendem Zugriff belegt hat; die Bearbeitung des Makros wird in diesem Fall abgebrochen. Soll erst eine bestimmte Zeit gewartet werden, bis die Datei wieder frei ist, kann dies durch einen der `FILE`- bzw. `SEGMENT`-Anweisung vorangehenden Aufruf der Makrofunktion `LOCK` (siehe Seite 490) erreicht werden. Dabei wird die Datei gleichzeitig für andere Zugriffe gesperrt; sie muss durch einen der `FILE`- bzw. `SEGMENT`-Anweisung folgenden Aufruf der Makrofunktion `UNLOCK` wieder freigegeben werden.

### Die Makroanweisung

```
$$ FILE/ERASE/BINARY "dateiname" = REQUEST (url, query)
```

ruft eine HTML-Seite von einem WWW-Server ab oder ruft ein CGI-Script auf einem WWW-Server auf. Die Antwort des WWW-Servers wird unverändert in die angegebene Datei geschrieben. Die schon in der Datei stehenden Daten werden zuvor gelöscht.

Die Argumente und weiter Einzelheiten der Makrofunktion `REQUEST` sind auf Seite 508 beschrieben.

## Ausgabe in eine Datei umleiten

Beim Abarbeiten von Makros werden Zeilen, die mit zwei Dollarzeichen beginnen, als Makroanweisungen interpretiert. Alle anderen Zeilen werden als Daten angesehen und in die zusammenzustellende Kommandofolge eingefügt, falls sie nicht auf Grund einer vorangehenden Makroanweisung übergangen werden oder eine besondere Bedeutung haben (z. B. Bestandteil der Definition einer Sternvariablen sind).

Mit der folgenden Makroanweisung kann die Ausgabe der Zeilen, die in die Kommandofolge eingefügt würden, in eine Datei umgeleitet werden:

```
$$ FILE "dateiname"
```

Die angegebene Datei darf eine TUSTEP- oder eine ASCII-Datei sein. In TUSTEP-Dateien werden die Sätze im Textmodus nummeriert. Mit der Makroanweisung

```
$$ FILE/PROGRAM "dateiname"
```

werden sie im Programmmodus nummeriert. Mit der Makroanweisung

```
$$ FILE/ERASE "dateiname"
```

werden die schon in der Datei stehenden Daten zuerst gelöscht. Die Optionen ERASE und PROGRAM können auch zusammen (in dieser Reihenfolge) angegeben werden.

Die Umleitung der Daten in die angegebene Datei erfolgt so lange, bis dies mit der Makroanweisung

```
$$ ENDFILE
```

wieder aufgehoben wird und damit die Bearbeitung der in der vorangehenden FILE-Anweisung angegebenen Datei abgeschlossen wird. Alternativ kann die Makroanweisung

```
$$ ENDFILE/PRINT
```

verwendet werden; mit ihr wird zusätzlich eine Meldung mit Angaben zu den ausgegebenen Daten ins Ablaufprotokoll ausgegeben.

Damit möglichst keine Daten (z. B. auf Grund eines Stromausfalls) verloren gehen, sollte die Dateibearbeitung jeweils abgeschlossen werden, bevor auf Benutzereingaben gewartet wird.

Während die Daten in eine Datei umgeleitet werden, können mit der Makroanweisung

```
$$ FILE/APPEND "dateiname"
```

aus einer beliebigen anderen Datei (auch aus einer ASCII-Datei) Daten unverändert übernommen werden.

Um die Daten auf eine Datei umleiten zu können, muss auf diese Datei schreibend zugegriffen werden. Dies ist jedoch nicht möglich, wenn schon ein Programm diese Datei mit lesendem oder schreibendem Zugriff belegt hat; die Bearbeitung des Makros wird in diesem Fall abgebrochen. Soll erst eine bestimmte Zeit gewartet werden, bis die Datei wieder frei ist, kann dies durch einen der FILE-Anweisung vorangehenden Aufruf der Makrofunktion LOCK (siehe Seite 490) erreicht werden. Dabei wird die

Datei gleichzeitig für andere Zugriffe gesperrt; sie muss durch einen der `ENDFILE`-Anweisung folgenden Aufruf der Makrofunktion `UNLOCK` wieder freigegeben werden.

## Ausgabe automatisch modifizieren

Mit der folgenden Makroanweisung können in Zeilen, die in die Kommandofolge eingefügt bzw. auf Grund der `FILE`-Anweisung (siehe Seite 430) in eine Datei ausgegeben werden, automatisch Zeichenfolgen ausgetauscht werden:

```
$$ EXCHANGE xtab
```

Danach werden jeweils die in der Austausch-Tabelle `xtab` angegebenen Suchzeichenfolgen durch die dazugehörigen Ersatzzeichenfolgen ersetzt.

Für `xtab` kann der Name einer mit der `BUILD`-Anweisung definierten Austausch-Tabelle oder direkt eine in Anführungszeichen eingeschlossene Austausch-Tabelle angegeben werden.

Das Austauschen der Zeichenfolgen erfolgt so lange, bis dies mit der Makroanweisung

```
$$ ENDEXCHANGE
```

wieder aufgehoben wird.

## Kommentare

Mit der Makroanweisung

```
$$- Kommentar
```

können Kommentare in ein Makro eingefügt werden. Sie werden beim Abarbeiten des Makros ignoriert.

Kommentar, der am Anfang eines Makros vor der Makroanweisung `$$!` (siehe Seite 413) steht, gilt als Beschreibung des Makros und kann mit dem Kommando `#INFORMIERE` (siehe Seite 167) ausgegeben werden.

Außerdem kann hinter jeder Makroanweisung (aber nicht in Datenzeilen) durch eine Tilde getrennt ein Kommentar angegeben werden.

## Meldungen

Mit den beiden folgenden Anweisungen kann z. B. vor Anfragen die Information ausgegeben werden, welche Antworten möglich sind und welche Wirkung sie im einzelnen haben:

```
$$+ Meldung
```

Der als Meldung angegebene Text wird während des Abarbeitens des Makros ins Ablaufprotokoll ausgegeben.

Enthält die Meldung in spitze Klammern eingeschlossene Variablennamen, so werden diese durch den Inhalt der entsprechenden Variablen ersetzt; bei Sternvariablen wird jedoch statt des Variablen-Inhalts nur ein Stern eingesetzt, wie dies bei Makroanweisungen der Regel entspricht. Von dieser Regel kann abgewichen werden, indem die Meldung mit folgender Makroanweisung ausgegeben wird:

```
$$* Meldung
```

Wenn bei dieser Makroanweisung die Meldung eine Sternvariable enthält, so wird die Meldung so oft ausgegeben, wie die Sternvariable Zeilen enthält; dabei wird der Name der Sternvariablen (einschließlich der Klammern) jeweils durch eine Zeile der Sternvariablen ersetzt; in einer solchen Meldung darf nur eine einzige Sternvariable vorkommen.

Für beide Arten von Meldungen gilt folgende Regelung, falls mit dem Kommando #PROTOKOLL (siehe Seite 209) der Protokoll-Modus auf PORTIONIERT oder FREILAUFEND eingestellt worden ist:

- Beginnt die Meldung mit »@@@« , so wird sie in der Farbe ausgegeben, die für Warnungen eingestellt ist.
- Beginnt die Meldung mit »@@@@@« , so wird sie in der Farbe ausgegeben, die für Fehlermeldungen eingestellt ist.
- Alle anders beginnenden Meldungen (auch solche, bei denen nach »+« bzw. »\*« mehr als ein Leerzeichen folgt) werden in der Farbe ausgegeben, die für Parameterausgabe eingestellt ist.

Hinweis: Die Farben können mit dem Kommando #DEFINIERE (siehe Seite 132) eingestellt werden.

#### Die Makroanweisung

```
$$ SHOW "Zeichenfolge"
```

zeigt die Zeichenfolge als Meldung im TUSTEP-Fenster an, jedoch nur so lange, bis sie durch eine andere Meldung überschrieben wird;

```
$$ SHOW/n "Zeichenfolge"
```

zeigt die Zeichenfolge als Meldung für mindestens n Sekunden im TUSTEP-Fenster an, danach so noch lange, bis sie durch eine andere Meldung überschrieben wird;

```
$$ PRINT "Zeichenfolge"
```

gibt die Zeichenfolge als Meldung ins Ablaufprotokoll aus;

```
$$ PRINT/WARNING "Zeichenfolge"
```

gibt sie in der Form und Farbe einer Warnung aus;

```
$$ PRINT/ERROR "Zeichenfolge"
```

gibt sie in der Form und Farbe einer Fehlermeldung aus.

```
$$ PRINT/COLOR:xx "Zeichenfolge"
```

gibt sie in der mit dem Hexadecimal-Code angegebenen Farbe aus. Die möglichen Codes können der Tabelle entnommen werden, die im Editor durch die Tastenkombination Ctrl+F bzw. Strg+F angezeigt wird.



```
$$ PRINT/FILE "Zeichenfolge"
```

gibt die Zeichenfolge als Meldung ins Ablaufprotokoll aus; falls jedoch die Makroanweisung `FILE` (siehe Seite 430) aktiviert ist, wird die Zeichenfolge in die entsprechende Datei ausgegeben.

```
$$ PRINT/DATA "Zeichenfolge"
```

fügt die Zeichenfolge in die zusammenzustellende Kommandofolge ein; falls jedoch die Makroanweisung `FILE` (siehe Seite 430) aktiviert ist, wird die Zeichenfolge in die entsprechende Datei ausgegeben.

Bei der `SHOW`-Anweisung und allen `PRINT`-Anweisungen kann an Stelle der in Anführungszeichen eingeschlossenen Zeichenfolge auch eine Variable oder eine Folge von Zeichenfolgen und Variablen durch Komma getrennt angegeben werden.

## Fehlermeldungen und Fehlerflag

Wird bei der Interpretation oder Ausführung einer Makroanweisung ein Fehler festgestellt, so wird eine Fehlermeldung ausgegeben und das Fehlerflag gesetzt.

Ist das Fehlerflag gesetzt,

- werden Schleifen nur noch maximal einmal durchlaufen,
- werden die Anweisungen `BROWSE`, `CALL`, `DISPLAY`, `DO`, `DOWNLOAD`, `EXECUTE`, `INCLUDE`, `UPLOAD` sowie `FILE`, `ACCESS`, `READ`, `WRITE`, `ERASE`, `COUNT`, `FIND` nicht mehr ausgeführt,
- werden die Funktionen `ASK`, `CLICK`, `DISPLAY`, `EDIT` sowie `CREATE`, `OPEN`, `CHECK`, `ERASE`, `CLOSE`, `DELETE`, `RENAME`, `LOCK`, `UNLOCK` nicht mehr ausgeführt,
- wird nach Abarbeiten des Makros die eventuell zusammengestellte Kommando-  
folge nicht ausgeführt.

Mit folgender Makroanweisung kann (z. B. weil das Makro mit nicht vorgesehenen Spezifikationswerten aufgerufen wurde) eine Fehlermeldung ausgegeben und das Fehlerflag gesetzt werden:

```
$$ ERROR "Zeichenfolge"
```

Wird die Zeichenfolge weggelassen, wird keine Fehlermeldung ausgegeben; es wird nur das Fehlerflag gesetzt.

Wenn das Fehlerflag gesetzt wird, während Makrofenster angezeigt werden, so werden diese automatisch geschlossen. Das Abarbeiten des Makros selbst wird durch die `ERROR`-Anweisung nicht abgebrochen. Soll das Abarbeiten des Makros beendet werden, kann dies durch die Makroanweisung

```
$$ ERROR/STOP "Zeichenfolge"
```

oder durch eine nachfolgende `STOP`-Anweisung erreicht werden.

An Stelle der in Anführungszeichen eingeschlossenen Zeichenfolge kann auch eine Variable oder eine Folge von Zeichenfolgen und Variablen durch Komma getrennt angegeben werden. Falls die als letzte angegebene Variable die Zeichenfolge `»OK«` enthält, wird keine Fehlermeldung ausgegeben und das Fehlerflag nicht gesetzt.

An Stelle der letzten Variablen kann auch eine der Makrofunktionen zur Dateiverwaltung CREATE, OPEN, CHECK, ERASE, DELETE, CLOSE, REARRANGE, COPY oder RENAME angegeben werden, z. B.:

```
$$ ERROR/STOP "QUELLE: ", OPEN (datei, READ, -STD-)
```

Falls die angegebene Makrofunktion als Funktionswert keine Fehlermeldung, sondern »OK« liefert, wird keine Fehlermeldung ausgegeben und das Fehlerflag nicht gesetzt.

Mit der Makroanweisung

```
$$ ERROR/RESET
```

kann das Fehlerflag gelöscht werden.

Hinweis: Mit der Makroanweisung PRINT/ERROR (siehe Seite 432) kann eine Fehlermeldung ausgegeben werden, ohne dass das Fehlerflag gesetzt wird.

## Signalton

Mit der folgenden Makroanweisung kann ein Signalton ausgegeben werden, um z. B. auf Fehlermeldungen aufmerksam zu machen:

```
$$ BEEP
```

Die Tonhöhe und Tonlänge kann mit dem Kommando #DEFINIERE (siehe Seite 132) eingestellt werden.

Mit der folgenden Makroanweisung kann eine Tonfolge ausgegeben werden:

```
$$ BEEP "tonfolge"
```

Die einzelnen Töne der Tonfolge werden in der gleichen Form wie beim Kommando #SIGNAL (siehe Seite 222) erwartet. An Stelle der in Anführungszeichen eingeschlossenen Tonfolge kann auch der Name einer Variablen angegeben werden, die die Tonfolge enthält.

## Warten

Mit den folgenden Makroanweisungen kann vor der Ausführung der nächsten Anweisung gewartet werden: Die Anweisung

```
$$ WAIT n
```

wartet n Sekunden, die Anweisung

```
$$ WAIT "Meldung"
```

gibt den als Meldung angegebenen Text ins Ablaufprotokoll aus und wartet bis die Return-Taste gedrückt wird.

## Schleifen

Die folgenden Makroanweisungen ermöglichen es, dass eine oder mehrere aufeinander folgende Zeilen mehrmals ausgeführt werden.

```
$$ LOOP variablenname1 = variablenname2
...
$$ ENDLLOOP
```

Falls die rechts vom Gleichheitszeichen angegebene Variable eine »normale« Variable ist, werden die zwischen LOOP und ENDLLOOP stehenden Zeilen für jede in dieser Variablen enthaltene Teilzeichenfolge ausgewertet. Dabei wird jeweils am Anfang der Schleife die aktuelle Teilzeichenfolge der links vom Gleichheitszeichen angegebenen Variablen zugewiesen. Teilzeichenfolgen sind Teile einer Zeichenfolge, die durch Apostroph getrennt sind. Enthält die rechts vom Gleichheitszeichen angegebene Variable eine leere Zeichenfolge, werden die zwischen LOOP und ENDLLOOP stehenden Zeilen nicht ausgewertet.

Falls die rechts vom Gleichheitszeichen angegebene Variable eine Sternvariable ist, werden die zwischen LOOP und ENDLLOOP stehenden Zeilen für jede in dieser Variablen enthaltene Zeile ausgewertet. Dabei wird jeweils am Anfang der Schleife die aktuelle Zeile der links vom Gleichheitszeichen angegebenen Variablen zugewiesen. Enthält die rechts vom Gleichheitszeichen angegebene Variable keine Zeilen, werden die zwischen LOOP und ENDLLOOP stehenden Zeilen nicht ausgewertet.

Rechts vom Gleichheitszeichen kann an Stelle einer Variablen auch eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben werden.

Nachdem die Schleife abgearbeitet ist, wird der links vom Gleichheitszeichen angegebenen Variablen eine leere Zeichenfolge zugewiesen. Falls die Schleife jedoch vorzeitig mit der EXIT-Anweisung (s. u.) verlassen wird, enthält sie noch die aktuelle Teilzeichenfolge bzw. Zeile.

```
$$ LOOP/CLEAR variablenname1 = variablenname2
...
$$ ENDLLOOP
```

Wird die Option CLEAR angegeben, wird der Inhalt der rechts vom Gleichheitszeichen angegebenen Variablen gelöscht, nachdem ihr Inhalt für die einzelnen Schleifendurchgänge gemerkt wurde. Dadurch kann sie innerhalb der Schleife (z. B. mit der Makrofunktion APPEND) wieder neu belegt werden.

```
$$ LOOP name1 = wertel, name2 = werte2, ...
...
$$ ENDLLOOP
```

Je Schleifendurchgang können auch mehreren Variablen jeweils ein Teilwert bzw. eine Zeile zugewiesen werden. Für name muss jeweils ein Variablenname, für werte jeweils eine in Anführungszeichen eingeschlossene Zeichenfolge oder ein Variablenname angegeben werden. Die Anzahl der Schleifendurchgänge richtet sich nach der Zeichenfolge bzw. der Variablen, die am meisten Teilzeichenfolgen bzw. Zeilen enthält. Für eine fehlende Teilzeichenfolge bzw. Zeile wird jeweils eine leere Zeichenfolge bzw. eine leere Zeile angenommen.

```

$$ LOOP variable = Anfangswert TO Endwert STEP Schrittweite
...
$$ ENDLOOP

```

oder

```

$$ LOOP variable = Anfangswert, Endwert, Schrittweite
...
$$ ENDLOOP

```

Die zwischen `LOOP` und `ENDLOOP` stehenden Zeilen werden für jeden Wert des Schleifenzählers ausgewertet. Dabei wird jeweils am Anfang der Schleife der Inhalt des Schleifenzählers der angegebenen Variablen zugewiesen. Der Schleifenzähler durchläuft die Werte vom angegebenen `Anfangswert` bis zum angegebenen `Endwert` in der angegebenen `Schrittweite`. Für `Anfangswert`, `Endwert` und `Schrittweite` dürfen Zahlenwerte (ggf. mit Vorzeichen) oder Variablen, die eine Zahl enthalten, angegeben werden. Die `Schrittweite` kann einschließlich des davor stehenden Kommas bzw. des Schlüsselwortes `STEP` weggelassen werden; in diesem Fall wird als `Schrittweite 1` angenommen.

Nachdem die Schleife abgearbeitet ist, wird der angegebenen Variablen eine leere Zeichenfolge zugewiesen. Falls die Schleife jedoch vorzeitig mit der `EXIT`-Anweisung (s. u.) verlassen wird, enthält sie noch den letzten Wert des Schleifenzählers.

```

$$ LOOP
...
$$ ENDLOOP

```

Die zwischen `LOOP` und `ENDLOOP` stehenden Zeilen werden so lange wiederholt ausgewertet, bis die Schleife durch die Makroanweisung

```

$$ EXIT

```

verlassen wird. Diese Makroanweisung kann bei allen vier Arten von Schleifen verwendet werden, wenn das Abarbeiten einer Schleife vorzeitig beendet werden soll. Das Abarbeiten des Makros wird dann nach der `ENDLOOP`-Anweisung fortgesetzt.

Sind Schleifen verschachtelt, so wird mit der `EXIT`-Anweisung nur die innerste Schleife, in der die Anweisung steht, beendet. Sollen nächst äußere Schleifen ebenfalls beendet werden, so kann in der gleichen Zeile hinter `EXIT` für jede weitere zu beendende Schleife ein zusätzliches `EXIT` durch Komma getrennt angegeben werden.

Die Makroanweisung

```

$$ CYCLE

```

bewirkt, dass in einem Schleifendurchgang die restlichen Zeilen bis zum `ENDLOOP` nicht mehr abgearbeitet werden, sondern mit dem nächsten Schleifendurchgang fortgefahren (d. h. das Abarbeiten des Makros nach der `LOOP`-Anweisung fortgesetzt) wird bzw. die Schleife beendet wird, falls sie abgearbeitet ist.

Bei verschachtelten Schleifen kann `CYCLE` auch durch Komma getrennt hinter `EXIT` angegeben werden. Für jedes `EXIT` wird dann eine Schleife beendet, und in der nächst äußeren Schleife werden die folgenden Zeilen bis zum `ENDLOOP` übergangen.

```

$$ LOOP/n
...
$$ ENDLOOP

```

Damit Endlosschleifen vermieden bzw. möglichst früh erkannt werden, vermutet TUSTEP nach 1000 Schleifendurchgängen eine Endlosschleife und beendet die Ausführung der Schleife mit der Fehlermeldung »Vermutlich Endlosschleife«. Soll eine Schleife mindestens *n*-mal durchlaufen werden können, bevor eine Endlosschleife vermutet wird, kann die Zahl *n* bei allen vier Arten der LOOP-Anweisungen durch einen Schrägstrich getrennt hinter dem Schlüsselwort LOOP angegeben werden.

```

$$ LOOP variablenname, ...
...
$$ ENDLOOP

```

Wird hinter dem Schlüsselwort LOOP bzw. hinter LOOP/*n* zusätzlich eine Variable angegeben, so wird ihr bei jedem Schleifendurchgang die laufende Nummer des jeweiligen Schleifendurchgangs zugewiesen. Nach Beenden der Schleife enthält die Variable die Anzahl der ausgeführten Schleifendurchgänge. Folgen in der LOOP-Anweisung hinter dieser Variablen noch weitere Angaben, müssen diese durch ein Komma von ihr getrennt werden.

Hinweis: Damit Schleifen zeitsparend ausgeführt werden, sollten sie mit der COMPILER-Anweisung (siehe Seite 453) compiliert werden. Außerdem sollten nur diejenigen Anweisungen innerhalb einer Schleife angegeben werden, die unbedingt innerhalb der Schleife ausgeführt werden müssen. Insbesondere sollten ACCESS-, FILE- und BUILD-Anweisungen vor der LOOP-Anweisung, ENDACCESS-, ENDFILE-, und RELEASE-Anweisungen nach der ENDLOOP-Anweisung statt innerhalb einer Schleife ausgeführt werden, soweit dies möglich ist.

## Auswahl über beliebige Bedingungen

Die folgende Makroanweisung ermöglicht es, dass eine andere Makroanweisung nur ausgeführt wird, wenn eine bestimmte Bedingung erfüllt ist.

```

$$ IF (Bedingung) Makroanweisung

```

Die möglichen Bedingungen sind ab Seite 461 beschrieben.

Folgende Makroanweisungen dürfen angegeben werden: ASK, BEEP, BREAK, BROWSE, BUILD, CALL, CANCEL, CHECK, CONTINUE, COUNT, CYCLE, DATA, DEFINE, DICTIONARY, DISPLAY, DO, DOWNLOAD, EDIT, ENDEXCHANGE, ENDFILE, ERASE, ERROR, EXCHANGE, EXECUTE, EXIT, FETCH, FILE, FIND, FINISH, GOTO, INCLUDE, MODIFY, KEY, PRINT, READ, RELEASE, REMOVE, RETURN, SET, SHOW, STACK, STOP, TERMINATE, TRACE, UNSET, UPLOAD, WAIT, WRITE.

Die folgenden Makroanweisungen ermöglichen es, dass eine oder mehrere aufeinander folgende Zeilen nur unter bestimmten Bedingungen ausgewertet werden:

```
$$ IF (Bedingung_1) THEN
...
$$ ELSEIF (Bedingung_2) THEN
...
$$ ELSEIF (Bedingung_3) THEN
...
...
...
$$ ELSEIF (Bedingung_n) THEN
...
$$ ELSE
...
$$ ENDIF
```

Die angegebenen Bedingungen werden der Reihe nach abgefragt. Ist eine Bedingung erfüllt, werden die nachfolgenden Zeilen bis zum nächsten `ELSEIF`, `ELSE` oder `ENDIF` ausgewertet. Alle anderen Zeilen zwischen `IF` und `ENDIF` werden übergangen. Ist keine Bedingung erfüllt, werden die Zeilen zwischen `ELSE` und `ENDIF` ausgewertet, falls `ELSE` angegeben ist.

Die Anzahl der `ELSEIF`-Anweisungen richtet sich nach der Anzahl der Bedingungen, die nacheinander abgefragt werden sollen. Soll nur eine einzige Bedingung abgefragt werden, entfallen die `ELSEIF`-Anweisungen. Ebenso kann die `ELSE`-Anweisung entfallen, falls es zwischen `IF` und `ENDIF` keine Zeilen gibt, die dann ausgewertet werden sollen, wenn keine der Bedingungen erfüllt ist.

Die Makroanweisung

```
$$ CONTINUE
```

bewirkt, dass die restlichen Zeilen bis zum `ENDIF` übergangen werden.

Sind `IF`-Anweisungen verschachtelt, so werden mit der `CONTINUE`-Anweisung nur die Zeilen bis zum `ENDIF` der innersten `IF`-Anweisung, in der die `CONTINUE`-Anweisung steht, übergangen. Sollen die Zeilen bis zum `ENDIF` einer äußeren `IF`-Anweisung übergangen werden, so kann in der gleichen Zeile hinter `CONTINUE` für jede weitere `IF`-Anweisung, deren restliche Zeilen übergangen werden sollen, ein zusätzliches `CONTINUE` durch Komma getrennt angegeben werden.

## Auswahl über Zeichenfolgen oder Zahlenwerte

Die folgenden Makroanweisungen ermöglichen es, dass eine oder mehrere aufeinander folgende Zeilen nur ausgewertet werden, wenn eine Zeichenfolge bzw. Zahl mit einer vorgegebenen Zeichenfolge bzw. Zahl übereinstimmt.

```

$$ SELECT variablenname
...
$$ CASE "Zflg_1"
...
$$ CASE "Zflg_2"
...
$$ CASE "Zflg_3", "Zflg_4"
...
...
...
$$ CASE "Zflg_n"
...
$$ DEFAULT
...
$$ ENDSELECT

```

In der `SELECT`-Anweisung kann an Stelle des Variablennamens auch eine in Anführungszeichen eingeschlossene Zeichenfolge stehen.

Falls es sich bei den zu vergleichenden Zeichenfolgen um Zahlen handelt, dürfen die Anführungszeichen sowohl in der `SELECT`-Anweisung als auch in den `CASE`-Anweisungen weggelassen werden.

Nach der Zeile mit der `SELECT`-Anweisung folgt in der Regel unmittelbar die Zeile mit der ersten `CASE`-Anweisung. Falls dazwischen jedoch andere Zeilen eingefügt sind, werden diese ausgeführt. Dann wird die in der `SELECT`-Anweisung stehende Zeichenfolge der Reihe nach mit den in den einzelnen `CASE`-Anweisungen stehenden verglichen. Wird eine Übereinstimmung gefunden, werden die jeweils nachfolgenden Zeilen bis zum nächsten `CASE`, `DEFAULT` oder `ENDSELECT` ausgewertet. Alle anderen Zeilen zwischen dem ersten `CASE` bis zum `ENDSELECT` werden übergangen. Stimmt keine in einer `CASE`-Anweisung angegebene Zeichenfolge mit der in der `SELECT`-Anweisung stehenden überein, werden die Zeilen zwischen `DEFAULT` und `ENDSELECT` ausgewertet, falls `DEFAULT` angegeben ist.

Die Anzahl der `CASE`-Anweisungen richtet sich nach der Anzahl der Zeichenfolgen, die nacheinander mit der hinter `SELECT` stehenden Zeichenfolge verglichen werden sollen. Falls für verschiedene Zeichenfolgen die gleichen Zeilen ausgewertet werden sollen, können am Ende der Zeile mit dem dazugehörigen `CASE` die entsprechenden Zeichenfolgen durch Komma getrennt zusätzlich angegeben werden. Die `DEFAULT`-Anweisung kann entfallen, falls es zwischen `SELECT` und `ENDSELECT` keine Zeilen gibt, die nur dann ausgewertet werden sollen, wenn keine der Zeichenfolgen übereinstimmt.

Beim Vergleich der Zeichenfolgen werden Groß- und Kleinbuchstaben nicht unterschieden, sie gelten als gleichwertig. Sollen sie bei einzelnen `CASE`-Anweisungen unterschieden werden, so kann nach dem Schlüsselwort `CASE` durch einen Schrägstrich getrennt die Option `EXACT` angegeben werden; sollen sie bei allen `CASE`-Anweisungen unterschieden werden, so kann nach dem Schlüsselwort `SELECT` durch einen Schrägstrich getrennt die Option `EXACT` angegeben werden.

## Die Makroanweisung

```
$$ BREAK
```

bewirkt, dass die restlichen Zeilen bis zum `ENDSELECT` übergangen werden.

Sind `SELECT`-Anweisungen verschachtelt, so werden mit der `BREAK`-Anweisung nur die Zeilen bis zum `ENDSELECT` der innersten `SELECT`-Anweisung, in der die `BREAK`-Anweisung steht, übergangen. Sollen die Zeilen bis zum `ENDSELECT` einer äußeren `SELECT`-Anweisung übergangen werden, so kann in der gleichen Zeile hinter `BREAK` für jede weitere `SELECT`-Anweisung, deren restliche Zeilen übergangen werden sollen, ein zusätzliches `BREAK` durch Komma getrennt angegeben werden.

## Such-Tabellen, Austausch-Tabellen, Recherchier-Tabellen

Mit den folgenden Makroanweisungen können Tabellen definiert werden. Dabei werden ihnen Namen zugewiesen, über die in anderen Makroanweisungen auf sie Bezug genommen werden kann.

```
$$ BUILD S_TABLE tabellenname = "Such-Tabelle"
```

```
$$ BUILD X_TABLE tabellenname = "Austausch-Tabelle"
```

```
$$ BUILD R_TABLE/option tabellenname = "Recherchier-Tabelle"
```

Der Tabellenname ist frei wählbar. Er kann aus 1 bis 12 Zeichen (Buchstaben, Ziffern und `»_«`) bestehen, muss mit einem Buchstaben beginnen und darf nicht mit `»_«` enden.

Die Such-Tabelle muss die Zeichenfolgen enthalten, nach denen gesucht werden soll. Für die Angabe dieser Such-Tabelle gelten die gleichen Regeln wie für das Informationsfeld von Parametern der Parameterart IX. Diese ist für die Konvention `»{ }«` ab Seite 714 und für die Konvention `»<>«` ab Seite 746 beschrieben. Nach welcher Konvention die Tabellen interpretiert werden, kann mit der `MODE`-Anweisung (siehe ab Seite 413) eingestellt werden.

Die Austausch-Tabelle muss die Zeichenfolgenpaare enthalten, deren jeweils erste Zeichenfolge die Suchzeichenfolge ist, die durch die jeweils zweite Zeichenfolge, die Ersatzzeichenfolge, ersetzt werden soll. Für die Angabe dieser Austausch-Tabelle gelten die gleichen Regeln wie für das Informationsfeld von Parametern der Parameterart X. Diese ist für die Konvention `»{ }«` ab Seite 721 und für die Konvention `»<>«` ab Seite 752 beschrieben. Nach welcher Konvention die Tabellen interpretiert werden, kann mit der `MODE`-Anweisung (siehe ab Seite 413) eingestellt werden.

Die Recherchier-Tabelle muss die Zeichenfolgen enthalten, nach denen gesucht werden soll. Für die Angabe dieser Recherchier-Tabelle gelten die gleichen Regeln wie für das Informationsfeld von Parametern der Parameterart XII. Diese ist für die Konvention `»{ }«` ab Seite 724 und für die Konvention `»<>«` ab Seite 756 beschrieben. Nach welcher Konvention die Tabellen interpretiert werden, kann mit der `MODE`-Anweisung (siehe ab Seite 413) eingestellt werden.

Beim Definieren einer Recherchier-Tabelle können nach dem Schlüsselwort `R_TABLE` jeweils durch Schrägstrich getrennt zusätzliche Optionen angegeben wer-



den. Falls in der Recherchier-Tabelle mehr als eine Zeichenfolge angegeben ist, muss mindestens eine der drei Optionen OR, AND oder USER angegeben werden; die Optionen OR und AND dürfen jedoch nicht zusammen angegeben werden. Ist die Option USER angegeben, sind die Besonderheiten der Parameterart XII-b zu beachten, andernfalls die Besonderheiten der Parameterart XII-a.

Für Recherchier-Tabellen sind folgende Optionen möglich:

– BLANK

Am Anfang und am Ende der Recherchier-Tabelle wird logisch ein Leerzeichen als Begrenzungszeichen ergänzt.

Setzt sich die Recherchier-Tabelle aus mehreren Zeilen zusammen (siehe unten am Ende dieses Kapitels), so wird für jeden Zeilenwechsel ebenfalls ein Leerzeichen eingefügt.

– EXACT

Groß- und Kleinbuchstaben werden als solche interpretiert, es sei denn, dass durch ein vor dem Buchstaben stehenden Backslash ausdrücklich der entsprechende Klein- bzw. Großbuchstabe verlangt wird.

Ist diese Option nicht angegeben, so ist bei Buchstaben, die nicht mit einem Backslash gekennzeichnet sind, jeweils der entsprechende Groß- und Kleinbuchstabe gemeint (gleichgültig, ob der Buchstabe groß oder klein geschrieben ist).

– n

Um eine zu suchende Zeichenfolge als übereinstimmend mit einer gefundenen gelten zu lassen, darf sie bis zu  $n$  ( $n = 0$  bis  $9$ ) Unterschiede (Ersetzungen, Auslassungen oder Einfügungen von je 1 Zeichen) aufweisen.

– n:m

Wie Option  $n$ , jedoch ist die Anzahl der erlaubten Unterschiede von der Länge der zu suchenden Zeichenfolge abhängig. Sie beträgt bei bis zu  $n$  Zeichen 0 Unterschiede, bis zu  $n+m$  Zeichen 1 Unterschied, bis zu  $n+m+m$  Zeichen 2 Unterschiede usw.

– WORD

Die zu suchenden Zeichenfolgen müssen in der zu durchsuchenden Zeichenfolge an einer Wortgrenze beginnen und enden. Als Wortgrenze gilt jedes Sonderzeichen (einschließlich Leerzeichen) sowie der Anfang und das Ende der zu durchsuchenden Zeichenfolge.

Hinweis: Da Akzente mit Sonderzeichen codiert werden (z. B. %/e für ein e mit Akut), müssen Akzent-Codierungen zuvor über eine Austausch-Tabelle eliminiert werden. Entsprechend müssen Sonderbuchstaben, die mit »#.« codiert sind, ersetzt werden (z. B. #. i durch i).

- TEXT

Die zu suchenden Zeichenfolgen müssen in der zu durchsuchenden Zeichenfolge an der Textgrenze (= Anfang und Ende der zu durchsuchenden Zeichenfolge) beginnen und enden.

- OR

Falls in der Recherchier-Tabelle mehr als eine Zeichenfolge angegeben ist, genügt es, wenn eine davon in der zu durchsuchenden Zeichenfolge vorkommt.

- AND

Falls in der Recherchier-Tabelle mehr als eine Zeichenfolge angegeben ist, müssen sie alle in der zu durchsuchenden Zeichenfolge vorkommen.

- USER

Falls in der Recherchier-Tabelle mehr als eine Zeichenfolge angegeben ist, können dazwischen logische Operatoren angegeben werden. Wird zwischen zwei Zeichenfolgen kein logischer Operator angegeben, so wird ein logisches ODER angenommen; dieses logische ODER hat Vorrang vor allen angegebenen logischen Operatoren.

Wenn die Option OR zusätzlich angegeben ist, so wird an Stellen, an denen zwischen zwei Zeichenfolgen kein logischer Operator angegeben ist, ein logisches ODER ohne Vorrang angenommen.

Wenn die Option AND zusätzlich angegeben ist, so wird an Stellen, an denen zwischen zwei Zeichenfolgen kein logischer Operator angegeben ist, ein logisches UND angenommen.

Wenn eine schon definierte Tabelle neu definiert werden soll, muss sie zuvor mit einer der folgenden Makroanweisungen freigegeben werden:

```
$$ RELEASE S_TABLE tabellenname
```

```
$$ RELEASE X_TABLE tabellenname
```

```
$$ RELEASE R_TABLE tabellenname
```

Es können auch mehrere Tabellennamen durch Komma getrennt angegeben werden.

Wenn eine Tabelle nicht mehr benötigt wird, kann sie ebenfalls freigegeben werden. Beim Beenden eines Makros werden Tabellen, die noch nicht freigegeben sind, automatisch freigegeben.

Wird eine Tabelle aus Benutzereingaben zusammengestellt, so ist es sinnvoll, sie erst auf Syntaxfehler zu prüfen, bevor sie mit der BUILD-Anweisung definiert wird. Dies kann mit folgenden Makroanweisungen geschehen:

```
$$ CHECK S_TABLE position+fehler = "Such-Tabelle"
```

```
$$ CHECK X_TABLE position+fehler = "Austausch-Tabelle"
```

```
$$ CHECK R_TABLE/option position+fehler = "Recherchier-Tabelle"
```

Falls in der Tabelle kein Syntaxfehler festgestellt wurde, wird der Variablen `position` der Wert 0 (Null) zugewiesen, andernfalls die laufende Nummer des Zeichens innerhalb der Tabelle, bei dem der Fehler festgestellt wird.

Falls in der Tabelle ein Syntaxfehler festgestellt wurde und mit der `MODE`-Anweisung (siehe Seite 413 ff.) die Konvention `»{ }«` eingestellt wurde, wird der Variablen `fehler` ein Fehlermeldungstext zugewiesen, andernfalls eine leere Zeichenfolge. Falls der Fehlermeldungstext nicht benötigt wird, kann die Variable `fehler` einschließlich des vorangehenden Pluszeichens weggelassen werden.

Beim Prüfen einer Recherchier-Tabelle mit der `CHECK`-Anweisung können nach dem Schlüsselwort `R_TABLE` die gleichen Optionen angegeben werden wie beim Definieren einer Recherchier-Tabelle mit der `BUILD`-Anweisung. Falls in der Recherchier-Tabelle mehr als eine Zeichenfolge angegeben ist, muss eine der Optionen `OR` oder `AND` angegeben werden.

In den Tabellen kann an Stelle eines einzelnen Zeichens einer Zeichenfolge auch die Kennung einer zuvor definierten Zeichengruppe stehen; in Such- und Austausch-Tabellen dürfen auch Kennungen von Stringgruppen verwendet werden. Diese Gruppen können mit den folgenden Makroanweisungen definiert werden:

```
$$ BUILD C_GROUP gruppenname = "Zeichengruppe"
```

```
$$ BUILD S_GROUP gruppenname = "Stringgruppe"
```

Ein Gruppenname ist ein aus zwei Zeichen bestehender Name, wobei `x` ein Buchstabe und `y` ein Buchstabe oder eine Ziffer sein muss. Groß- und Kleinschreibung wird nicht unterschieden.

Falls jedoch mit dem der `MODE`-Anweisung (siehe ab Seite 413) nicht `MODUS={ }` eingestellt wurde, muss in der `BUILD`-Anweisung und in der nachfolgend beschriebenen `RELEASE`-Anweisung an Stelle eines Gruppennamens eine Gruppenkennung angegeben werden. Gruppenkennungen für Zeichengruppen haben die Form `»>[xy]«` und Gruppenkennungen für Stringgruppen haben die Form `»<[xy]«` (jeweils ohne Anführungszeichen). Dabei ist `xy` der Gruppenname. Daneben sind noch Gruppenkennungen für Zeichen- und Stringgruppen der Form `»>n«` und `»<n«` (ohne Anführungszeichen) möglich, wobei `n` jeweils durch eine Ziffer zu ersetzen ist. Es können auf diese Weise also bis zu 20 Zeichengruppen und bis zu 20 Stringgruppen zusätzlich definiert werden.

Die Zeichengruppe bzw. Stringgruppe muss die Zeichen bzw. Strings (Zeichenfolgen) enthalten, für die die Gruppe stellvertretend stehen soll. Für die Angabe der Zeichen bzw. Stringgruppe gelten die gleichen Regeln wie für das Informationsfeld von Parametern der Parameterart `V`. Diese ist für die Parameter-Konvention `»{ }«` ab Seite 721 und für die Parameter-Konvention `»<>«` ab Seite 752 beschrieben. Nach welcher Konvention die Parameter interpretiert werden, kann mit dem Kommando `#PARAMETER` (siehe Seite 207) eingestellt werden.

Die Definition einer Gruppe gilt jeweils für alle nachfolgenden `BUILD`- und `CHECK`-Anweisungen, bis die Gruppe neu definiert oder mit einer der folgenden Makroanweisungen freigegeben wird:

```
$$ RELEASE C_GROUP gruppenname
```

```
$$ RELEASE S_GROUP gruppenname
```

Es können auch mehrere Gruppennamen durch Komma getrennt angegeben werden.

Wenn eine Gruppe nicht mehr benötigt wird, kann sie freigegeben werden. Beim Beenden eines Makros werden die Gruppen, die noch nicht freigegeben sind, automatisch freigegeben.

Wird eine Zeichen- oder Stringgruppe aus Benutzereingaben erstellt, so ist es sinnvoll, sie erst auf Syntaxfehler zu prüfen, bevor sie mit der `BUILD`-Anweisung definiert wird. Dies kann mit einer der folgenden Makroanweisungen geschehen:

```
$$ CHECK C_GROUP position+fehler = "Zeichengruppe"
$$ CHECK S_GROUP position+fehler = "Stringgruppe"
```

Falls in der Zeichen- bzw. Stringgruppe kein Syntaxfehler festgestellt wurde, wird der Variablen `position` der Wert 0 (Null) und der Variablen `fehler` eine leere Zeichenfolge zugewiesen, andernfalls die laufende Nummer des Zeichens innerhalb der Gruppe, bei dem der Fehler festgestellt wurde, und ein Fehlermeldungstext. Falls der Fehlermeldungstext nicht benötigt wird, kann die Variable `fehler` einschließlich des vorangehenden Pluszeichens weggelassen werden.

Bei den oben beschriebenen `BUILD`- und `CHECK`-Anweisungen kann die jeweils zwischen den Anführungszeichen angegebene Zeichenfolge auch auf mehrere Zeilen aufgeteilt werden; diese müssen unmittelbar nach der `BUILD`- bzw. `CHECK`-Anweisung folgen; sie enden vor der nächsten Makroanweisung (= nächste Zeile, die mit `$$` beginnt). In der `BUILD`- bzw. `CHECK`-Anweisung muss in diesem Fall an Stelle der in Anführungszeichen eingeschlossenen Zeichenfolge ein Stern angegeben werden. Der besseren Lesbarkeit wegen empfiehlt es sich, die Zeileneinteilung so zu wählen, dass jede Zeile mit einem Begrenzungszeichen anfängt und mit einem Begrenzungszeichen endet.

Die beiden folgenden Beispiele sind gleichwertig:

```
$$ BUILD X_TABLE tausch = "/((/#1+))/#1-/"
$$ BUILD X_TABLE tausch = *
/((/#1+
/))/#1-/"
```

Bei den `BUILD`- und `CHECK`-Anweisungen kann nach dem Gleichheitszeichen auch der Name einer Variablen angegeben werden, die die entsprechende Tabelle enthält.

Wird nach dem Gleichheitszeichen eine Stern oder eine Sternvariable angegeben, so muss in Fortsetzungszeilen das gleiche Begrenzungszeichen wie in der ersten Zeile verwendet werden. Ein Begrenzungszeichen am Anfang einer Fortsetzungszeile (ggf. nach `DATA`) und ein Begrenzungszeichen am Ende der vorhergehenden Zeile gelten zusammen als nur ein Begrenzungszeichen.

Der besseren Lesbarkeit wegen empfiehlt es sich, diese Eigenschaft auszunutzen und die Zeileneinteilung so zu wählen, dass jede Zeile (ggf. nach `DATA`) mit einem Begrenzungszeichen anfängt und mit einem Begrenzungszeichen endet.

Wird eine Zeichenfolge in der nächsten Zeile fortgesetzt, so darf jedoch weder am Ende der Zeile noch am Anfang der nächsten Zeile (ggf. nach `DATA`) ein Begrenzungszeichen stehen. In diesem Fall ist zu beachten, dass Leerzeichen am Zeilenende ignoriert werden, nicht aber am Zeilenanfang.

## Wörterbücher

Ein Wörterbuch ist ein Speicher, in dem zu frei wählbaren Schlüsselwörtern jeweils ein beliebiger Textteil abgelegt werden kann. Jedes Schlüsselwort kann nur einmal im Wörterbuch stehen. Über das Schlüsselwort kann der dazugehörige Textteil wieder abgefragt werden. Schlüsselwort und Textteil bilden einen Wörterbucheintrag.

TUSTEP bietet ein erweitertes Wörterbuch. Jeder Wörterbucheintrag enthält die Information, als wievielter er ins Wörterbuch aufgenommen wurde und wie oft auf ihn schon zugegriffen wurde. Außerdem können zu jedem Schlüsselwort null, ein oder zwei Textteile abgelegt werden.

```
$$ DICTIONARY name CREATE anzahl
```

richtet ein Wörterbuch mit dem angegebenen Namen ein.

Mit dem Argument `anzahl` kann festgelegt werden, wieviele Einträge das Wörterbuch maximal aufnehmen kann. Für `anzahl` kann eine Zahl oder eine Variable, die eine Zahl enthält, angegeben werden. Falls `anzahl` nicht angegeben wird, kann das Wörterbuch bis zu 1000 Einträge aufnehmen.

Beim Prüfen, ob ein Schlüsselwort schon im Wörterbuch steht, werden Groß- und Kleinschreibung nicht unterschieden. Sollen sie unterschieden werden, muss hinter dem Schlüsselwort `CREATE` durch einen Schrägstrich getrennt die Option `EXACT` angegeben werden.

Hinter dem Schlüsselwort `CREATE` kann auch die Option `TAGS` angegeben werden. In diesem Fall gelten folgende Sonderregeln: Ein »Schlüsselwort« darf nur ein oder mehrere Anfangs-Tags ohne Attribute (vgl. »Makrofunktionen für Tags« Seite 559) enthalten. In Tag-Namen kann »?« als Stellvertreter für ein beliebiges Zeichen stehen, jedoch nicht im letzten Tag; an Stelle eines Tag-Namens kann auch »\*« als Stellvertreter für einen beliebigen Tag-Namen stehen. Vor dem ersten Tag und zwischen den Tags kann »\*« als Stellvertreter für null oder beliebig viele Anfangs-Tags stehen. Weitere Trennzeichen oder Angaben sind im »Schlüsselwort« nicht erlaubt. Die Option `TAGS` ist erforderlich, wenn die weiter unten beschriebene Aktion `MATCH` verwendet wird. Nur bei dieser Aktion werden »?« und »\*« als Stellvertreter angesehen; bei allen anderen Aktionen haben diese beiden Zeichen keine Sonderfunktion.

Hinter dem Schlüsselwort `CREATE` kann auch die Option `INFIX` angegeben werden. In diesem Fall gelten folgende Sonderregeln: In den zum jeweiligen Schlüsselwort gehörenden Textteilen müssen sämtliche Zeichen des Schlüsselwortes in der gleichen Reihenfolge enthalten sein. Es dürfen an jeder Stelle zusätzliche Zeichen eingefügt werden, jedoch muss erkenntlich bleiben, welche Zeichen aus dem Schlüsselwort übernommen wurden und welche Zeichen eingefügt wurden. Es ist gleichgültig, ob im Schlüsselwort, das in das Wörterbuch eingefügt wird, die Buchstaben groß oder klein geschrieben werden. Jeder Klein- bzw. Großbuchstabe muss aber in den dazugehörenden Textteilen in gleicher Weise als Klein- bzw. Großbuchstabe angegeben werden. Die Option `INFIX` ist erforderlich, wenn die weiter unten beschriebene Aktion `INFIX` verwendet wird.

```
$$ DICTIONARY name action/option key, num, cnt, value1, value2
```

Nach der Ausführung der Anweisung gibt die Variable `num` an, als wievielteltes das in der Variablen `key` enthaltene Schlüsselwort ins Wörterbuch eingefügt wurde, die Variable `cnt` nennt den Stand des Zugriffszählers. Der Zugriffszähler beginnt bei Null; er wird bei jedem Zugriff auf das Schlüsselwort um eins erhöht, falls die Option `COUNT` angegeben ist, und um eins erniedrigt, falls die Option `DISCOUNT` angegeben ist.

Jede der Variablen `num`, `cnt`, `value1` und `value2` kann einschließlich des jeweils davor stehenden Kommas weggelassen werden, wenn sie als letzte in der Anweisung stehen würde und die dazugehörige Information nicht benötigt wird.

Für `action` sind folgende Angaben vorgesehen:

- **ADD:** Fügt das in der Variablen `key` stehende Schlüsselwort mit den Textteilen, die in den Variablen `value1` und `value2` stehen, hinzu. Steht das Schlüsselwort schon im Wörterbuch, wird nur das Fehlerflag gesetzt und eine Fehlermeldung ausgegeben. Soll beides unterbleiben, so muss die Option `QUIET` angegeben werden. An Stelle der Variablen `key`, `value1` und `value2` kann jeweils auch eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben werden.
- **UPDATE:** Sucht das in der Variablen `key` stehende Schlüsselwort und ersetzt die dazugehörigen Textteile durch die in den Variablen `value1` und `value2` stehenden Textteile. Steht das Schlüsselwort nicht im Wörterbuch, wird nur das Fehlerflag gesetzt und eine Fehlermeldung ausgegeben. Soll beides unterbleiben, so muss die Option `QUIET` angegeben werden. An Stelle der Variablen `key`, `value1` und `value2` kann jeweils auch eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben werden.
- **APPEND:** Sucht das in der Variablen `key` stehende Schlüsselwort und ergänzt die dazugehörigen Textteile durch die in den Variablen `value1` und `value2` stehenden Textteile. Zwischen die schon vorhandenen und die hinzugefügten Textteile wird zusätzlich ein Apostroph als Trennzeichen ergänzt. Soll eine andere Zeichenfolge als Trennzeichen ergänzt werden, so kann diese in Anführungszeichen eingeschlossen hinten in der `DICTIONARY`-Anweisung nach einem Strichpunkt angegeben werden; an Stelle dieser Zeichenfolge kann auch eine Variable angegeben werden, die die entsprechende Zeichenfolge enthält. Steht das Schlüsselwort nicht im Wörterbuch, wird nur das Fehlerflag gesetzt und eine Fehlermeldung ausgegeben. Soll beides unterbleiben und das Schlüsselwort mit den Textteilen wie bei `ADD` in das Wörterbuch eingetragen werden, so muss die Option `QUIET` angegeben werden. An Stelle der Variablen `key`, `value1` und `value2` kann jeweils auch eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben werden.
- **LOOKUP:** Sucht das in der Variablen `key` stehende Schlüsselwort und speichert die dazugehörigen Textteile in die Variablen `value1` und `value2`. Steht das Schlüsselwort nicht im Wörterbuch, wird diesen beiden Variablen eine leere Zeichenfolge und den Variablen `num` und `cnt` der Wert 0 (Null) zugewiesen. An Stelle der Variablen `key` kann auch eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben werden.
- **ALTER:** Wie `LOOKUP`; jedoch bleiben die beiden Variablen `value1` und `value2` unverändert, falls das Schlüsselwort nicht im Wörterbuch steht.

- EXCHANGE: Wie LOOKUP; jedoch wird den beiden Variablen `value1` und `value2` das Schlüsselwort zugewiesen, falls das Schlüsselwort nicht im Wörterbuch steht.
- MATCH: Wie LOOKUP mit folgenden Sonderregeln: Das in der Variablen `key` stehende »Schlüsselwort« darf nur ein oder mehrere Anfangs-Tags ohne Attribute enthalten. Im Wörterbuch wird nach einem »Schlüsselwort« gesucht, das die gleichen Tags enthält. Jedoch dürfen an Stellen, an denen ein im Wörterbuch stehendes Schlüsselwort ein »?« enthält, in dem in der Variablen `key` stehenden Schlüsselwort ein beliebiges Zeichen stehen, und an den Stellen, an denen ein im Wörterbuch stehendes Schlüsselwort »<\*>« enthält, in dem in der Variablen `key` stehenden Schlüsselwort ein beliebiges Anfangs-Tag stehen, und an den Stellen, an denen ein im Wörterbuch stehendes Schlüsselwort einen »\*« außerhalb der Tags enthält, in dem in der Variablen `key` stehenden Schlüsselwort null oder beliebig viele zusätzliche Tags stehen. Falls das Wörterbuch mehrere zutreffende »Schlüsselwörter« enthält, gilt folgende Regelung: Alle Muster, bei denen das letzte Tag die Form »<name>« hat, haben Vorrang vor den Mustern, bei denen das letzte Tag die Form »<\*>« hat; innerhalb dieser beiden Gruppen hat das zuerst ins Wörterbuch eingefügte Vorrang.
- INFIX: Wie EXCHANGE mit folgender Sonderregel: Wenn das in der Variablen `key` stehende Schlüsselwort im Wörterbuch enthalten ist, werden vor der Rückgabe eines Textteils die Buchstaben, die vom Schlüsselwort in den Textteil übernommen werden, in der Form (d. h. als Groß- bzw. Kleinbuchstabe) eingesetzt, in der sie in der Variablen `key` (also nicht wie im Wörterbuch) stehen.
- RECALL: Sucht das Schlüsselwort, das als `n`-tes in das Wörterbuch eingefügt wurde, speichert das Schlüsselwort in die Variable `key` und die dazugehörigen Textteile in die Variablen `value1` und `value2`; die Nummer `n` des Schlüsselwortes muss mit der Variablen `num` angegeben werden. Enthält das Wörterbuch weniger als `n` Schlüsselwörter, wird den Variablen `key`, `value1` und `value2` eine leere Zeichenfolge und der Variablen `cnt` der Wert 0 (Null) zugewiesen.
- UNLOAD: Weist der Variablen `key` alle Schlüsselwörter in der Reihenfolge zu, in der sie in das Wörterbuch eingefügt wurden. In der Variablen `num` wird die Anzahl der Wörterbucheinträge gespeichert. Parallel zu den Schlüsselwörtern in der Variablen `key` werden der Variablen `cnt` die jeweiligen Zugriffszähler und den Variablen `value1` und `value2` die jeweiligen Textteile zugewiesen.
- LOAD: Fügt die Schlüsselwörter, die in der Sternvariablen `key` stehen, mit den Zugriffszählern, die parallel zu den Schlüsselwörtern in der Sternvariablen `cnt` stehen, und mit den Textteilen, die parallel zu den Schlüsselwörtern in den Sternvariablen `value1` und `value2` stehen, hinzu. Der Variable `num` wird die Anzahl der eingefügten Schlüsselwörter zugewiesen. Steht ein Schlüsselwort schon im Wörterbuch, wird das Fehlerflag gesetzt und eine Fehlermeldung ausgegeben; die in der Variablen `key` nachfolgenden Schlüsselwörter werden nicht mehr eingefügt.

```
$$ DICTIONARY name UNLOAD/FILE "dateiname", "trenner"
```

Schreibt den Inhalt des Wörterbuchs in die angegebene Datei; falls die Datei schon Daten enthält, werden diese zuvor gelöscht. Jedes Schlüsselwort wird zusammen mit den Textteilen in einen eigenen Satz geschrieben. Falls nach der Option `FILE` noch

die Option `COUNTER` angegeben ist, wird der Zugriffszähler jeweils zwischen dem Schlüsselwort und den Textteilen eingefügt. Zwischen den einzelnen Werten wird die mit `trenner` angegebene Zeichenfolge ergänzt. Diese muss so gewählt werden, dass sie eindeutig ist. Die Schlüsselwörter werden in der Reihenfolge in die Datei geschrieben, in der sie in das Wörterbuch eingefügt wurden.

```
$$ DICTIONARY name LOAD/FILE "dateiname", "trenner"
```

Fügt die Schlüsselwörter, die in der angegebenen Datei stehen, ins Wörterbuch ein. In jedem Satz der Datei wird ein Schlüsselwort mit den dazugehörigen Textteilen erwartet. Falls nach der Option `FILE` noch die Option `COUNTER` angegeben ist, wird jeweils zwischen dem Schlüsselwort und den Textteilen noch der Wert des Zugriffszählers erwartet. Schlüsselwort, Zugriffszähler und die Textteile müssen jeweils durch die mit `trenner` angegebene Zeichenfolge getrennt sein. Fehlende Textteile werden wie leere Textteile, fehlende Zugriffszähler wie 0 (Null) gewertet. Steht ein Schlüsselwort schon im Wörterbuch, wird das Fehlerflag gesetzt und eine Fehlermeldung ausgegeben.

```
$$ DICTIONARY name SIZE anz
```

speichert die Anzahl der im Wörterbuch enthaltenen Einträge in der Variablen `anz`.

```
$$ DICTIONARY name RESET key
```

sucht das in der Variablen `key` stehende Schlüsselwort und setzt den dazugehörigen Zugriffszähler auf Null. Steht das Schlüsselwort nicht im Wörterbuch, wird das Fehlerflag gesetzt und eine Fehlermeldung ausgegeben. Soll beides unterbleiben, so muss hinter dem Schlüsselwort `RESET` durch einen Schrägstrich getrennt die Option `QUIET` angegeben werden. An Stelle der Variablen `key` kann auch eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben werden.

```
$$ DICTIONARY name RESET
```

setzt alle Zugriffszähler auf Null.

```
$$ DICTIONARY name CLEAR
```

löscht alle im Wörterbuch enthaltenen Einträge.

```
$$ DICTIONARY name DELETE
```

löscht das Wörterbuch.

Der Anweisungsname `DICTIONARY` kann mit `DICT` abgekürzt werden.

## Stapelspeicher

Bei einem Stapelspeicher können Werte, d. h. Inhalte von Variablen, oben auf den (zu Anfang leeren) Stapel abgelegt werden. Die Namen der Variablen werden nicht mit abgelegt. Auf den Stapel abgelegte Werte können in umgekehrter Reihenfolge von oben wieder entfernt und den gleichen oder anderen Variablen zugewiesen werden.

TUSTEP bietet einen erweiterten Stapelspeicher. Bei ihm können Werte auch (wie bei einer Warteschlange in der Reihenfolge, in der sie auf den Stapel gelegt wurden) von unten vom Stapel entfernt und Variablen zugewiesen werden.



```
$$ STACK name CREATE anzahl
```

richtet einen Stapelspeicher mit dem angegebenen Namen ein.

Mit dem Argument `anzahl` kann festgelegt werden, wieviele Werte der Stapelspeicher maximal aufnehmen kann. Für `anzahl` kann eine Zahl oder eine Variable, die eine Zahl enthält, angegeben werden. Falls `anzahl` nicht angegeben wird, kann der Stapelspeicher bis zu 1000 Werte aufnehmen.

```
$$ STACK name PUSH var
```

legt den Inhalt der Variablen `var` auf den Stapel ab. An Stelle der Variablen `var` kann auch eine in Anführungszeichen eingeschlossene Zeichenfolge oder eine Folge von Variablen und Zeichenfolgen jeweils durch Komma getrennt angegeben werden. Die Variablen können auch Sternvariablen sein; sie werden jeweils als ein Wert abgelegt.

Wird nach dem Schlüsselwort `PUSH` durch einen Schrägstrich getrennt die Option `CLEAR` angegeben, so wird der Inhalt der angegebenen Variablen gelöscht, nachdem er auf den Stapel gelegt wurde.

```
$$ STACK name SPLIT var
```

wirkt wie die zuvor beschriebene Anweisung, jedoch wird bei Sternvariablen jede Zeile einzeln als Wert abgelegt.

Wird nach dem Schlüsselwort `PUSH` durch einen Schrägstrich getrennt die Option `CLEAR` angegeben, so wird der Inhalt der angegebenen Variablen gelöscht, nachdem er auf den Stapel gelegt wurde.

```
$$ STACK name POP var
```

wirkt wie die nachfolgend beschriebene Anweisung.

```
$$ STACK name LIFO var
```

nimmt den obersten Wert vom Stapel (last in, first out) und weist ihn der Variablen `var` zu. Werden mehrere Variablen jeweils durch Komma getrennt angegeben, wird für jede Variable, beginnend mit der letzten, ein Wert oben vom Stapel genommen und ihr zugewiesen.

```
$$ STACK name FIFO var
```

nimmt den untersten Wert vom Stapel (first in, first out) und weist ihn der Variablen `var` zu. Werden mehrere Variablen jeweils durch Komma getrennt angegeben, wird für jede Variable, beginnend mit der ersten, ein Wert unten vom Stapel genommen und ihr zugewiesen.

```
$$ STACK name PEEK var
```

liest den obersten Wert vom Stapel, ohne ihn zu entfernen, und weist ihn der Variablen `var` zu. Werden mehrere Variablen jeweils durch Komma getrennt angegeben, wird für jede Variable, beginnend mit der letzten, der nächste Wert vom Stapel gelesen und ihr zugewiesen. Der Stapelspeicher bleibt in jedem Fall unverändert.

```
$$ STACK name LOOK var
```

liest den untersten Wert vom Stapel, ohne ihn zu entfernen, und weist ihn der Variablen `var` zu. Werden mehrere Variablen jeweils durch Komma getrennt angege-

ben, wird für jede Variable, beginnend mit der ersten, der nächste Wert vom Stapel gelesen und ihr zugewiesen. Der Stapelspeicher bleibt in jedem Fall unverändert.

```
$$ STACK name SIZE anz
```

speichert die Anzahl der im Stapelspeicher enthaltenen Werte in der Variablen *anz*.

```
$$ STACK name CLEAR
```

löscht alle im Stapelspeicher enthaltenen Werte.

```
$$ STACK name DELETE
```

löscht den Stapelspeicher.

Falls mit `POP`, `FIFO`, `LIFO`, `PEEK` oder `LOOK` ein Wert von einem leeren Stapelspeicher gelesen werden soll, erfolgt eine Fehlermeldung und das Fehlerflag wird gesetzt. Beides kann vermieden werden, indem hinter den genannten Schlüsselwörtern durch einen Schrägstrich getrennt die Option `QUIET` angegeben wird. Innerhalb einer Schleife kann statt der Option `QUIET` die Option `EXIT` angegeben werden; dann wird automatisch eine `EXIT`-Anweisung ausgeführt (d. h. die Schleife wird automatisch verlassen), wenn ein Wert von einem (inzwischen) leeren Stapelspeicher gelesen werden soll.

## Sektionen

Um Makros übersichtlicher gestalten zu können, kann eine Folge von Makroanweisungen als Sektion definiert werden:

```
$$ SECTION sektionsname  
...  
$$ ENDSECTION
```

Die zwischen `SECTION` und `ENDSECTION` stehenden Makroanweisungen werden noch nicht bei der Definition der Sektion ausgeführt, sondern erst, wenn die Sektion mit der Makroanweisung

```
$$ DO sektionsname
```

aufgerufen wird.

Eine Sektion hat keine »eigenen« Variablen. In einer Sektion sind diejenigen Variablen bekannt, die bis zum Aufruf definiert wurden bzw. die in der Sektion definiert werden. Nach dem Abarbeiten einer Sektion behalten die Variablen, die in der Sektion geändert oder definiert wurden, ihren neuen Inhalt.

Sollen die Makroanweisungen in einer Sektion nicht bis zum Ende der Sektion abgearbeitet werden, kann mit der Makroanweisung

```
$$ RETURN
```

vorzeitig zu der auf die `DO`-Anweisung folgenden Makroanweisung zurückgekehrt werden.

## Submakros

Um Makros übersichtlicher gestalten zu können, kann eine Folge von Makroanweisungen als Submakro definiert werden:

```
$$ SUBMACRO submakroname
$$! spezifikationen
...
$$ ENDSUBMACRO
```

Die zwischen `SUBMACRO` und `ENDSUBMACRO` stehenden Makroanweisungen werden noch nicht bei der Definition des Submakros ausgeführt, sondern erst, wenn das Submakro mit der Makroanweisung

```
$$ CALL submakroname (spezifikationsangaben)
```

aufgerufen wird.

Ein Submakro hat »eigene« Variablen. In einem Submakro sind nur diejenigen Variablen bekannt, die entweder im Submakro definiert werden oder die mit der Makroanweisung `DEFINE/COMMON` (siehe Seite 419) zuvor definiert wurden. Variablen können in Submakros in gleicher Weise wie in Hauptmakros auch durch Spezifikationen definiert werden (siehe »Spezifikationen« Seite 413). Eine so definierte Variable hat den gleichen Namen wie die jeweilige Spezifikation; sie wird im Folgenden Spezifikationsvariable genannt.

In der `CALL`-Anweisung folgen nach dem Namen des Submakros in der Regel noch Spezifikationsangaben, falls im entsprechenden Submakro Spezifikationen definiert sind. Die Spezifikationsangaben müssen in Klammern eingeschlossen und durch Komma getrennt werden.

Eine Spezifikationsangabe kann die Form einer Wertzuweisung haben. Dabei steht links vom Gleichheitszeichen der Spezifikationsname und rechts davon der Spezifikationswert. Der Spezifikationswert kann eine Zahl, eine in Anführungszeichen eingeschlossene Zeichenfolge oder ein Variablenname sein. Beim Aufruf des Submakros wird die angegebene Zahl bzw. die angegebene Zeichenfolge bzw. der Inhalt der Variablen in die entsprechende Spezifikationsvariable des Submakros übertragen. Falls im Aufruf als Spezifikationswert eine Variable angegeben ist, wird nach dem Abarbeiten des Submakros in diese Variable der zuletzt gültige Inhalt der entsprechenden Spezifikationsvariablen zurückübertragen. Wird zu einer Spezifikation keine Angabe gemacht, so gilt für diese Spezifikation der voreingestellte Wert.

Die Reihenfolge der Spezifikationen ist im Submakro festgelegt. Der Spezifikationsname und das Gleichheitszeichen können im Aufruf weggelassen werden, falls diese Reihenfolge eingehalten wird. Wenn der Spezifikationsname angegeben wird, muss die Reihenfolge nicht eingehalten werden. Innerhalb eines Aufrufs können beide Möglichkeiten gemischt auftreten.

Nach dem Abarbeiten eines Submakros werden alle Variablen des Submakros gelöscht. Soll der Inhalt einer im Submakro verwendeten Variablen weiter verwendet werden, so muss diese im Submakro als Spezifikation definiert werden; beim Aufruf des Submakros muss als Spezifikationswert eine Variable angegeben werden; diese Variable erhält nach dem Abarbeiten des Submakros den letzten Wert der entsprechenden Variablen im Submakro.

Außer den Variablen sind alle Definitionen (z. B. Such- und Austausch-Tabellen) in Submakros bekannt, die zum Zeitpunkt des Aufrufs definiert sind bzw. die im Submakro definiert werden. Nach dem Abarbeiten eines Submakros behalten diese Definitionen weiter ihre Gültigkeit, falls eine Definition nicht explizit im Submakro gelöscht wurde.

Sollen die Makroanweisungen in einem Submakro nicht bis zum Ende des Submakros abgearbeitet werden, kann mit der Makroanweisung

```
$$ RETURN
```

vorzeitig zu der auf die CALL-Anweisung folgenden Makroanweisung zurückgekehrt werden.

## Einfügen von Segmenten/Dateien

Werden die gleichen Sektionen, Submakros oder Makrofenster (s. u.) in mehreren Makros benötigt, so können diese Sektionen, Submakros und Makrofenster in eigenen Segmenten der selben Makro-Datei abgespeichert und dann mit der Makroanweisung

```
$$ INCLUDE segmentname
```

in das jeweilige Makro (logisch) eingefügt werden. Dabei kann ein Segment eine Sektion, ein Submakro, ein Makrofenster oder auch mehrere Sektionen, Submakros und/oder Makrofenster enthalten. Es empfiehlt sich, diese INCLUDE-Anweisungen am Anfang des Makros anzugeben, um einen Überblick über die in einem Makro benötigten Segmente zu erhalten.

Ein mit der INCLUDE-Anweisung eingefügtes Segment darf auch eine beliebige Folge von Makroanweisungen und Daten enthalten. Die entsprechende INCLUDE-Anweisung muss in diesem Fall an der Stelle im Makro stehen, an der diese Makroanweisungen abgearbeitet werden sollen. Außerdem kann in diesem Fall das eingefügte Segment auch in einer anderen Segment-Datei stehen als das Makro, das die INCLUDE-Anweisung enthält; diese Segment-Datei muss in der INCLUDE-Anweisung angegeben werden:

```
$$ INCLUDE "dateiname" segmentname
```

Wird bei der INCLUDE-Anweisung nur ein Dateiname und kein Segmentname angegeben, so wird die ganze Datei eingefügt.

Wird in der INCLUDE-Anweisung ein Dateiname angegeben und enthält das eingefügte Segment bzw. die eingefügte Datei Sektionen, Submakros oder Makrofenster, so können diese nur während des Abarbeitens dieses Segments bzw. dieser Datei aufgerufen werden. Danach können sie nicht mehr aufgerufen werden. Waren vor der Ausführung dieser INCLUDE-Anweisung gleichnamige Sektionen, Submakros oder Makrofenster definiert, können diese ebenfalls nicht mehr aufgerufen werden, bevor sie nicht neu definiert werden.

Für jedes eingefügte Segment bzw. für jede eingefügte Datei gelten der voreingestellte Modus und die voreingestellten Steuerzeichen. Makroanweisungen müssen also mit zwei Dollarzeichen gekennzeichnet sein; in spitze Klammern eingeschlossene Vari-

ablennamen werden durch den Inhalt der entsprechenden Variablen ersetzt; der Apostroph gilt als Trennzeichen zwischen Teilzeichenfolgen. Mit der `MODE`-Anweisung (siehe ab Seite 413) und der Makroanweisung `$$=` (siehe Seite 417) können jedoch ein anderer Modus bzw. andere Steuerzeichen festgelegt werden.

Wenn in einem einzufügenden Segment bzw. in einer einzufügenden Datei nur Daten (d. h. keine Makroanweisungen) stehen und in diesen Daten auch keine in Klammern eingeschlossenen Variablennamen ersetzt werden sollen, kann zusätzlich die Option `DATA` angegeben werden:

```
$$ INCLUDE/DATA ...
```

Die Daten des angegebenen Segments werden damit unverändert in die Kommando­folge eingefügt bzw. auf Grund der `FILE`-Anweisung (siehe Seite 430) in eine Datei ausgegeben.

Die beiden folgenden Anweisungen sind nur in einem CGI-Makro erlaubt, d. h. wenn `TUSTEP` über `WWW` durch einen `WWW`-Server im Batch-Modus aufgerufen wird:

```
$$ INCLUDE/BINARY/FILE "dateiname"
```

Die in der angegebenen Datei enthaltenen Daten werden binär, d. h. unverändert in die Standard-Ausgabe eingefügt.

```
$$ INCLUDE/BINARY/VARIABLE variablenname
```

Die in der angegebenen Variablen enthaltenen Daten werden binär, d. h. unverändert in die Standard-Ausgabe eingefügt.

## Compilieren von Makroanweisungen

Beim Abarbeiten eines Makros wird eine Makroanweisung nach der anderen interpretiert und jeweils sofort ausgeführt. Dies hat zur Folge, dass Makroanweisungen, die mehrmals abgearbeitet werden (z. B. innerhalb von Schleifen), auch jedes Mal neu interpretiert werden. Um das wiederholte (zeitaufwändige) Interpretieren zu sparen, können Makroanweisungen »compiliert« werden. Dabei werden aus den Makroanweisungen interne Anweisungen erzeugt, die dann beim Abarbeiten nur noch ausgeführt werden müssen.

Eine Folge von Anweisung, die compiliert werden soll, muss mit der Makroanweisung

```
$$ COMPILER
```

eingeleitet und mit der Makroanweisung

```
$$ ENDCOMPILER
```

abgeschlossen werden.

Makrovariablen, die in spitzen Klammern stehen, werden durch ihren Inhalt ersetzt. Dies geschieht jeweils unmittelbar vor dem Interpretieren einer Anweisung. Dadurch wird für jede Variable der Wert eingesetzt, den sie zu diesem Zeitpunkt hat. Dies gilt auch beim Compilieren von Anweisungen. Jedoch ist zu beachten, dass alle

Anweisungen zwischen `COMPILE` und `ENDCOMPILE` zuerst kompiliert und dann erst ausgeführt werden. Für Makrovariablen, die innerhalb dieser Anweisungen verändert werden, werden deshalb nicht die veränderten Werte eingesetzt, sondern die Werte, die die Makrovariablen zu Beginn des Kompilierens hatten. Deshalb dürfen nur solche Makrovariablen in spitzen Klammern angegeben werden, die zwischen `COMPILE` und `ENDCOMPILE` nicht verändert werden, oder bei denen jeweils die Werte eingesetzt werden sollen, die sie zu Beginn des Kompilierens hatten.

In den Anweisungen

```
$$ SET namen = "Maier'Müller'Schulze"
$$ LOOP nummer = 1, 3
$$   SET name = SELECT (namen, <nummer>)
$$   PRINT "<nummer>: <name>"
$$ ENDLOOP
```

wird die Variable `nummer` in der 2. Zeile mit einem Wert belegt, der danach in der 3. und 4. Zeile eingesetzt wird; die Variable `name` erhält in der 3. Zeile einen Wert, der in der 4. Zeile eingesetzt wird. Die Anweisungen geben folgende Zeilen ins Ablaufprotokoll aus:

```
1: Maier
2: Müller
3: Schulze
```

Beim Kompilieren der Anweisungen

```
$$ COMPILE
$$ SET namen = "Maier'Müller'Schulze"
$$ LOOP nummer = 1, 3
$$   SET name = SELECT (namen, <nummer>)
$$   PRINT "<nummer>: <name>"
$$ ENDLOOP
$$ ENDCOMPILE
```

muss in der 4. und 5. Zeile der Inhalt der Variablen `nummer` eingesetzt werden. Falls diese Variable schon vorher einen Wert erhalten hat, wird dieser jeweils eingesetzt, andernfalls wird für beide Zeilen eine Fehlermeldung ausgegeben, dass diese Variable noch nicht definiert sei; das gleiche gilt für die Variable `name`, die in der 5. Zeile eingesetzt werden soll. Im Fehlerfall wird keine der Anweisungen zwischen `COMPILE` und `ENDCOMPILE` ausgeführt.

Hat z. B. die Variable `nummer` von einer vorangehenden Zuweisung noch den Wert »123« und die Variable `name` noch den Wert »Beispiel«, werden nach dem Ersetzen der in spitzen Klammern stehenden Variablennamen folgende Anweisungen kompiliert:

```
$$ SET namen = "Maier'Müller'Schulze"
$$ LOOP nummer = 1, 3
$$   SET name = SELECT (namen, 123)
$$   PRINT "123: Beispiel"
$$ ENDLOOP
```

Nach dem Compilieren werden die Anweisungen ausgeführt; dabei werden folgende Zeilen ins Ablaufprotokoll ausgegeben:

```
123: Beispiel
123: Beispiel
123: Beispiel
```

Um das gleiche Ergebnis wie oben zu erhalten, müssen die Anweisungen so umformuliert werden, dass keine Variablen in spitzen Klammern mehr vorkommen.

```
$$ COMPILE
$$ SET namen = "Maier'Müller'Schulze"
$$ LOOP nummer = 1, 3
$$   SET name = SELECT (namen, #nummer)
$$   PRINT nummer, ":", name
$$ ENDLOOP
$$ ENDCOMPILE
```

Dabei wäre es nahe liegend, in der 4. Zeile nur die spitzen Klammern wegzulassen, ohne ein Nummernzeichen zu ergänzen. Das würde in vielen Fällen genügen. Die Wirkung der Makrofunktion `SELECT` ist jedoch unterschiedlich, je nachdem, ob als zweites Argument eine Zahl oder ein Name angegeben wird. Ist als zweites Argument eine Zahl `n` angegeben, wird der `n`-te Teilwert aus der Variablen `namen` ausgewählt. Würde als zweites Argument nur `nummer` angegeben, so würde `nummer` als Name einer Such-Tabelle interpretiert und das Ergebnis wäre nicht das gewünschte. Damit der Variablenname `nummer` nicht als Tabellename, sondern als Name einer Variablen interpretiert wird, die die Zahl `n` enthält, muss der Variablenname mit einem Nummernzeichen gekennzeichnet werden.

Wenn als Argument einer Makrofunktion eine Zahl erwartet wird, kann an ihrer Stelle auch ein mit einem Nummernzeichen gekennzeichneteter Name einer Variablen angegeben werden, die die Zahl enthält.

Wenn als Argument einer Makrofunktion eine Zeichenfolge erwartet wird, kann auch ein mit einem Dollarzeichen gekennzeichneteter Name einer Variablen angegeben werden, die die Zeichenfolge enthält. Falls keine Verwechslung möglich ist (weil z. B. an dieser Stelle kein Tabellename vorgesehen ist), kann die Kennzeichnung des Variablennamens mit Nummernzeichen bzw. Dollarzeichen entfallen.

Wenn in einer Makroanweisung ein in Anführungszeichen eingeschlossener Dateiname erwartet wird, kann auch ein mit einem Dollarzeichen gekennzeichneteter Name einer Variablen (ohne Anführungszeichen) angegeben werden, die den Dateinamen enthält.

Wenn als Argument einer Makrofunktion der Name einer System-Variablen erwartet wird, kann auch ein mit einem Klammeraffen gekennzeichneteter Name einer Makrovariablen angegeben werden, die den Namen der System-Variablen enthält.

Sektionen und Submakros können jeweils auch als ganze Einheit compiliert werden. Es genügt, hinter dem Schlüsselwort `SECTION` bzw. `SUBMACRO` durch einen Schrägstrich getrennt die Option `COMPILE` anzugeben. Würde stattdessen in einer Sektion bzw. in einem Submacro als erste Anweisung `COMPILE` und als letzte `ENDCOMPILE` angegeben, so würden die dazwischen stehenden Anweisungen bei jedem Aufruf erst

compiliert und dann ausgeführt. Entsprechendes gilt beim Compilieren einer Schleife. Obwohl nur die Anweisungen zwischen `LOOP` und `ENDLOOP` mehrfach ausgeführt werden, muss die `COMPILE`-Anweisung vor der `LOOP`-Anweisung und die `ENDCOMPILE`-Anweisung nach der `ENDLOOP`-Anweisung stehen, da die Anweisungen in der Schleife sonst bei jedem Schleifendurchgang erst compiliert und dann ausgeführt würden.

Außer durch Compilieren von Schleifen kann auch dadurch Zeit gespart werden, dass nur diejenigen Anweisungen innerhalb einer Schleife angegeben werden, die unbedingt innerhalb der Schleife ausgeführt werden müssen. Insbesondere sollten `ACCESS`-, `FILE`- und `BUILD`-Anweisungen vor der `LOOP`-Anweisung, `ENDACCESS`-, `ENDFILE`- und `RELEASE`-Anweisungen nach der `ENDLOOP`-Anweisung statt innerhalb einer Schleife ausgeführt werden, soweit dies möglich ist. Das gleiche gilt auch für Sektionen und Submakros, die häufig aufgerufen werden.

## Ausführen von Kommandos

Kommandos, die von einem Makro erzeugt werden, werden erst ausgeführt, nachdem das ganze Makro abgearbeitet oder mit der Makroanweisung `STOP` vorzeitig beendet worden ist. Mit den folgenden Makroanweisungen können Kommandos (mit Ausnahme der Kommandos `#TUE` und `#MAKRO`) jedoch sofort während des Arbeitens des Makros ausgeführt werden:

```
$$ EXECUTE #kommando
```

Falls die Ablauf-Protokollierung nicht mit dem Kommando `#PROTOKOLL` (siehe Seite 209) ausgeschaltet ist, wird die Ausführung des Kommandos protokolliert.

```
$$ EXECUTE/QUIET #kommando
```

Wird die Option `QUIET` angegeben, so wird die Ausführung des Kommandos nicht protokolliert.

```
$$ EXECUTE #name
```

Kommandos, die ein Makro aufrufen (auf `#` folgt in diesem Fall `*` oder `$`), sind nur für solche Makros erlaubt, die ihre Leistung allein durch Ausführen von Makroanweisungen erbringen; Makros, die eine Kommandofolge zusammenstellen, die anschließend automatisch ausgeführt werden soll, dürfen mit `EXECUTE` nicht aufgerufen werden.

Einschränkungen:

Die `EXECUTE`-Anweisung kann nicht ausgeführt werden, solange ein Makrofenster (s. u.) angezeigt wird.

In Kommandos, die mit der `EXECUTE`-Anweisung ausgeführt werden, darf mit `»*«` als Spezifikationswert nicht angegeben werden, dass Daten (z. B. Parameter) auf das Kommando folgen. Die Daten müssen in diesem Fall (z. B. mit Hilfe der `FILE`-Anweisung) zuvor in eine Hilfsdatei geschrieben werden. Dann kann im Kommando der Name dieser Hilfsdatei als Spezifikationswert angegeben werden.



## Ausführen von System-Kommandos

Mit der EXECUTE-Anweisung können auch System-Kommandos aufgerufen werden:

```
$$ EXECUTE "systemkommando"
```

Das zwischen den Anführungszeichen angegebene System-Kommando kann auch auf mehrere Zeilen aufgeteilt werden; diese müssen unmittelbar nach der EXECUTE-Anweisung folgen; sie enden vor der nächsten Makroanweisung. In der EXECUTE-Anweisung muss in diesem Fall an Stelle der in Anführungszeichen eingeschlossenen Zeichenfolge ein Stern angegeben werden.

Dabei werden diejenigen Zeilen als zum System-Kommando gehörend angesehen, die zwischen der EXECUTE-Anweisung und der auf sie folgenden Makroanweisung (= nächste Zeile, die mit \$\$ beginnt) stehen.

Die beiden folgenden Beispiele sind gleichwertig:

```
$$ EXECUTE "echo Triviales Beispiel"

$$ EXECUTE *
echo Triviales
Beispiel
```

Einschränkungen:

Mit der EXECUTE-Anweisung dürfen nur solche System-Kommandos aufgerufen werden, die keine Eingabe von der Tastatur oder mit der Maus erwarten.

Führt das System-Kommando zu Ausgaben im TUSTEP-Fenster, erfolgen diese in gleicher Weise von TUSTEP unkontrolliert, wie wenn mit dem Kommando #PROTOKOLL (siehe Seite 209) der Protokoll-Modus auf SYSTEM (Voreinstellung) eingestellt ist.

## Ausführen von Programmen

Unter Windows können mit der EXECUTE-Anweisung auch Programme, die in EXE- oder BAT-Dateien gespeichert sind, aufgerufen und in einem eigenen Fenster ausgeführt werden:

```
$$ EXECUTE "pfad" "parameter"
```

Das Makro wird erst weiter abgearbeitet, nachdem das aufgerufene Programm beendet wurde. Soll nicht gewartet werden, bis das Programm beendet wurde, so kann dies mit der Option CONTINUE erreicht werden:

```
$$ EXECUTE/CONTINUE "pfad" "parameter"
```

Die Angabe pfad muss den vollständigen Pfad samt Dateiname der Datei enthalten, in der das Programm gespeichert ist. Dieser Pfad kann in der Regel mit der Makrofunktion FULL\_NAME (siehe Seite 498) abgefragt werden.

Die Parameter, die das Programm erwartet, können in Anführungszeichen eingeschlossen angegeben werden. Sie können aber auch auf mehrere Zeilen aufgeteilt werden; diese müssen unmittelbar nach der EXECUTE-Anweisung folgen; sie enden

vor der nächsten Makroanweisung. In der EXECUTE-Anweisung muss in diesem Fall an Stelle der in Anführungszeichen eingeschlossenen Parameter ein Stern angegeben werden. Erwartet das Programm keine Parameter, muss eine leere Zeichenfolge ("" ) angegeben werden.

### Starten einer TUSTEP-Sitzung

```
$$ START SESSION "name"
```

startet die mit dem Makro \*DESI definierte TUSTEP-Sitzung mit dem angegebenen Namen.

### Starten des TUSTEP-Editors

```
$$ START EDIT "dateiname"
```

startet den Editor mit der angegebenen Datei in einer eigenen TUSTEP-Sitzung.

Der angegebene Dateiname kann ein Dateiname nach der für TUSTEP üblichen Konvention oder nach der für das jeweilige Betriebssystem üblichen Konvention sein (unter Windows mit Laufwerksbuchstabe oder Tilde beginnend, unter Linux und macOS mit Schrägstrich oder Tilde beginnend).

### Starten und Beenden von Anwendungen

Mit der folgenden BROWSE-Anweisung kann für eine Datei, deren Name eine Endung hat, der im Betriebssystem eine Anwendung (Programm) zugeordnet ist, die entsprechende Anwendung gestartet werden:

```
$$ BROWSE "dateiname"
```

Nach dem Starten der Anwendung wird das Makro weiter abgearbeitet. Soll damit gewartet werden, bis die gestartete Anwendung beendet wird, so kann dies unter Windows mit der Option WAIT erreicht werden:

```
$$ BROWSE/WAIT "dateiname"
```

Der in der BROWSE-Anweisung angegebene Dateiname kann ein Dateiname nach der für TUSTEP üblichen Konvention oder nach der für das jeweilige Betriebssystem üblichen Konvention sein (unter Windows mit Laufwerksbuchstabe oder Tilde beginnend, unter Linux und macOS mit Schrägstrich oder Tilde beginnend).

Enthält z. B. eine Datei mit dem Namen »test.html« eine HTML-Seite, so kann mit der BROWSE-Anweisung diese Seite mit dem im Betriebssystem voreingestellten Browser angezeigt werden.

An Stelle des Dateinamens kann auch eine URL für das WWW (z. B. »http://www.tustep.uni-tuebingen.de«) angegeben werden.

```
$$ BROWSE/CANCEL "titel"
```

beendet unter Windows Anwendungen, deren Fenster-Titel mit dem angegebenen Titel übereinstimmen.

## Fehlersuche

Bei der Suche nach logischen Fehler in Makros können die folgenden Makroanweisungen hilfreich sein:

```
$$ TRACE ON
```

bewirkt, dass alle Zeilen, die das Makro ausführt, ins Ablaufprotokoll ausgegeben werden. Die Ausgabe erfolgt, nachdem die in spitzen Klammern angegebenen Variablennamen durch den aktuellen Inhalt der jeweiligen Variablen ersetzt wurden.

```
$$ TRACE OFF
```

beendet dieses Protokollierung wieder.

```
$$ TRACE "Zeichenfolge"
```

gibt die angegebene Zeichenfolge ins Ablaufprotokoll aus.

```
$$ TRACE "Zeichenfolge" variablenname
```

gibt zusätzlich den Inhalt der angegebenen Variablen aus; es können auch mehrere Variablennamen durch Komma getrennt angegeben werden.

```
$$ TRACE * variablenname
```

gibt die Nummer des Satzes und den Namen der Datei, in der die TRACE-Anweisung steht, sowie den Inhalt der angegebenen Variablen aus; es können auch mehrere Variablennamen durch Komma getrennt angegeben werden.

```
$$ TRACE + variablenname
```

gibt den Namen und den Inhalt der angegebenen Variablen aus und bewirkt, dass bei jeder nachfolgenden Änderung der angegebenen Variablen automatisch Name und Inhalt protokolliert wird; es können auch mehrere Variablennamen durch Komma getrennt angegeben werden. Wird kein Variablenname angegeben, so wird die automatische Protokollierung für alle Variablen aktiviert.

```
$$ TRACE - variablenname
```

beendet diese Protokollierung für die angegebene Variable wieder; es können auch mehrere Variablennamen durch Komma getrennt angegeben werden. Wird kein Variablenname angegeben, so wird die automatische Protokollierung für alle Variablen beendet.

```
$$ TRACE $
```

Jede Änderung der Steuerzeichen (siehe »Steuerzeichen ändern« Seite 398) wird protokolliert.

```
$$ TRACE
```

entspricht der Makroanweisung TRACE ON; zusätzlich werden bei sämtlichen Wertzuweisungen (auch wenn der Inhalt der Variablen dabei nicht verändert wird) automatisch Name und Inhalt der Variablen protokolliert.

```
$$ TRACE ALL
```

entspricht der Makroanweisung TRACE (ohne weiteren Angaben); jedoch werden

alle Zeilen, die das Makro abarbeitet (auch solche, die nur interpretiert aber nicht ausgeführt werden), ins Ablaufprotokoll ausgegeben.

```
$$ TRACE NOTHING
```

beendet jegliche mit der Makroanweisung TRACE eingestellte Protokollierung.

```
$$ ENDTRACE
```

entspricht der Makroanweisung TRACE NOTHING.

## Anstehende Kommandos löschen

Mit der Makroanweisung

```
$$ CANCEL
```

können alle eventuell noch zur Ausführung anstehenden Kommandos gelöscht werden.

## Abarbeiten des Makros beenden

Mit der Makroanweisung

```
$$ STOP
```

kann das Abarbeiten des Makros vorzeitig beendet werden. Sie bewirkt kein Setzen des Fehlerflags. Soll das Fehlerflag gesetzt werden, muss zuvor die ERROR-Anweisung ausgeführt werden. Ist das Fehlerflag bei der Ausführung der STOP-Anweisung nicht gesetzt, so wird die eventuell bereits zusammengestellte Kommandofolge anschließend ausgeführt.

## TUSTEP-Sitzung beenden

Mit der Makroanweisung

```
$$ TERMINATE
```

kann eine TUSTEP-Sitzung beendet werden. Ein eventuell in der INI-Datei (siehe Seite 89) vorhandenes Segment mit dem Namen TERM wird dabei nicht ausgeführt.

## Bedingungen

Im Folgenden sind die Bedingungen beschrieben, die innerhalb der Klammern in Auswahlanweisungen (siehe Seite 437) verwendet werden können.

Zahl\_1 und Zahl\_2 steht jeweils für eine Zahl (= Ziffernfolge) mit oder ohne Vorzeichen; Zflg, Zflg\_1, Zflg\_2, . . . steht jeweils für eine beliebige Zeichenfolge.

Wurde mit der MODE-Anweisung (siehe ab Seite 413) der Modus VARIABLE oder TUSCRIPT eingestellt, kann an Stelle der abzufragenden Zahl oder Zeichenfolge auch der Name einer Variablen stehen.

Beispiel: \$\$ IF (antwort.ab."nein") STOP

Falls mit der MODE-Anweisung nicht der Modus VARIABLE oder TUSCRIPT eingestellt wurde, steht an Stelle der abzufragenden Zahl oder Zeichenfolge in der Regel der in spitze Klammern eingeschlossene Name einer Variablen. Die Prüfung, ob eine Bedingung erfüllt ist, erfolgt, nachdem die Namen der Variablen durch ihre aktuellen Werte ersetzt sind.

Beispiel: \$\$ IF ("<>antwort>" .ab."nein") STOP

Beim Vergleich bzw. Überprüfen von Zeichenfolgen werden Groß- und Kleinbuchstaben nicht unterschieden, sie gelten als gleichwertig, wenn bei der jeweiligen Bedingung nichts Gegenteiliges angegeben ist.

An Stelle einer Bedingung können auch mehrere Bedingungen, die jeweils durch einen der folgenden logischen Operatoren miteinander verbunden sind, angegeben werden.

- .AND. beide Bedingungen müssen erfüllt sein
- .OR. mindestens eine Bedingung muss erfüllt sein
- .XOR. genau eine Bedingung muss erfüllt sein
- .EQV. entweder darf keine oder müssen beide erfüllt sein

Werden mehr als zwei Bedingungen mit logischen Operatoren verbunden, so erfolgt die Auswertung der Operatoren in der Reihenfolge, in der sie oben aufgeführt sind. Von dieser Regelung kann abgewichen werden, indem die Priorität durch Setzen von Klammern festgelegt wird.

Beispiel: \$\$ IF (n.EQ.0 .AND. (w.LT.1 .OR. w.GT.9)) STOP

Damit auch mehrere Bedingungen, die durch logische Operatoren verbunden sind, übersichtlich bleiben, kann nach jedem logischen Operator eine neue Zeile begonnen werden.

Beispiel: \$\$ IF (antwort.AB."ja" .OR.  
 \$\$           antwort.AB."yes") STOP

Für die logischen Operatoren

.AND. und .OR.

können auch die Formen

&& und ||

verwendet werden.

Für die Vergleichsoperatoren

`.EQ. .NE. .LE. .LT. .GE. .GT.`

können auch die Formen

`== != <= < >= >`

verwendet werden.

## Vergleichen von Zahlen

Wurde mit der `MODE`-Anweisung (siehe ab Seite 413) der Modus `VARIABLE` oder `TUSCRIPT` eingestellt, kann an Stelle einer Zahl auch der Name einer Variablen angegeben werden. In diesem Fall wird die in der angegebenen Variablen enthaltene Zahl verglichen.

Achtung: Falls für beide Zahlen Variablenamen angegeben werden, muss mindestens einer der Variablenamen mit einem vorangestellten »#« gekennzeichnet werden, damit die Inhalte der Variablen als Zahlenwert verglichen werden; wird keiner der beiden Variablenamen mit »#« gekennzeichnet, werden die Inhalte der Variablen als Zeichenfolge alphabetisch verglichen (»9« wäre in diesem Fall größer als »10«, »1« größer als »02«).

- **Zahl\_1 .EQ. Zahl\_2**

Zahl\_1 ist gleich Zahl\_2

- **Zahl\_1 .EQ. Zahl\_2, Zahl\_3, ...**

Zahl\_1 stimmt mit mindestens einer der Zahlen Zahl\_2, Zahl\_3, ... überein.

- **Zahl\_1 .NE. Zahl\_2**

Zahl\_1 ist nicht gleich Zahl\_2

- **Zahl\_1 .NE. Zahl\_2, Zahl\_3, ...**

Zahl\_1 stimmt mit keiner der Zahlen Zahl\_2, Zahl\_3, ... überein.

- **Zahl\_1 .LT. Zahl\_2**

Zahl\_1 ist kleiner als Zahl\_2

- **Zahl\_1 .GT. Zahl\_2**

Zahl\_1 ist größer als Zahl\_2

- **Zahl\_1 .LE. Zahl\_2**

Zahl\_1 ist kleiner oder gleich Zahl\_2

- **Zahl\_1 .GE. Zahl\_2**

Zahl\_1 ist größer oder gleich Zahl\_2

## Vergleichen von Zeichenfolgen

Wurde mit der `MODE`-Anweisung (siehe ab Seite 413) der Modus `VARIABLE` oder `TUSCRIPT` eingestellt, kann an Stelle einer in Anführungszeichen eingeschlossenen Zeichenfolge auch der Name einer Variablen angegeben werden. In diesem Fall wird die in der angegebenen Variablen enthaltene Zeichenfolge verglichen.

Beim Vergleichen von Zeichenfolgen wird die Standard-Sortierfolge (siehe Seite 841) zugrunde gelegt. Für Zeichenfolgen, die sich auf Grund der Standard-Sortierfolge nicht unterscheiden, wird zusätzlich die Sonder-Sortierfolge (siehe Seite 841) zugrunde gelegt.

Groß- und Kleinbuchstaben gelten als gleichwertig und abschließende Leerstellen werden ignoriert, wenn in der jeweiligen Bedingung nicht verlangt wird, dass die Zeichenfolgen »identisch« bzw. »nicht identisch« sind.

- `"Zflg_1" .EQ. "Zflg_2"`  
Zflg\_1 und Zflg\_2 sind gleich.
- `"Zflg_1" .ID. "Zflg_2"`  
Zflg\_1 und Zflg\_2 sind identisch.
- `"Zflg_1" .EQ. "Zflg_2", "Zflg_3", ...`  
Zflg\_1 stimmt mit mindestens einer der Zeichenfolgen Zflg\_2, Zflg\_3, ... überein.
- `"Zflg_1" .ID. "Zflg_2", "Zflg_3", ...`  
Zflg\_1 ist mit mindestens einer der Zeichenfolgen Zflg\_2, Zflg\_3, ... identisch
- `"Zflg_1" .NE. "Zflg_2"`  
Zflg\_1 und Zflg\_2 sind nicht gleich.
- `"Zflg_1" .NI. "Zflg_2"`  
Zflg\_1 und Zflg\_2 sind nicht identisch.
- `"Zflg_1" .NE. "Zflg_2", "Zflg_3", ...`  
Zflg\_1 stimmt mit keiner der Zeichenfolgen Zflg\_2, Zflg\_3, ... überein.
- `"Zflg_1" .NI. "Zflg_2", "Zflg_3", ...`  
Zflg\_1 ist mit keiner der Zeichenfolgen Zflg\_2, Zflg\_3, ... identisch.
- `"Zflg_1" .LT. "Zflg_2"`  
Zflg\_1 steht bei alphabetischer Ordnung vor Zflg\_2
- `"Zflg_1" .GT. "Zflg_2"`  
Zflg\_1 steht bei alphabetischer Ordnung nach Zflg\_2

- 
- **"Zflg\_1" .LE. "Zflg\_2"**  
Zflg\_1 und Zflg\_2 sind gleich oder Zflg\_1 steht bei alphabetischer Ordnung vor Zflg\_2
  - **"Zflg\_1" .GE. "Zflg\_2"**  
Zflg\_1 und Zflg\_2 sind gleich oder Zflg\_1 steht bei alphabetischer Ordnung nach Zflg\_2
  - **"Zflg\_1" .SW. "Zflg\_2"**  
Zflg\_1 beginnt mit Zflg\_2.
  - **"Zflg\_1" .SI. "Zflg\_2"**  
Zflg\_1 beginnt identisch mit Zflg\_2.
  - **"Zflg\_1" .EW. "Zflg\_2"**  
Zflg\_1 endet mit Zflg\_2.
  - **"Zflg\_1" .EI. "Zflg\_2"**  
Zflg\_1 endet identisch mit Zflg\_2.
  - **"Zflg\_1" .PA. "Zflg\_2"**  
Zflg\_1 ist in Zflg\_2 enthalten.
  - **"Zflg\_1" .NP. "Zflg\_2"**  
Zflg\_1 ist nicht in Zflg\_2 enthalten.
  - **"Zflg\_1" .SS. "Zflg\_2"**  
Zflg\_1 ist als Teilzeichenfolge in Zflg\_2 enthalten.
  - **"Zflg\_1" .NS. "Zflg\_2"**  
Zflg\_1 ist nicht als Teilzeichenfolge in Zflg\_2 enthalten.
  - **"Zflg\_1" .AB. "Zflg\_2"**  
Zflg\_1 und Zflg\_2 sind gleich oder Zflg\_1 ist eine Abkürzung von Zflg\_2. Ein am Ende von Zflg\_1 evtl. vorhandener Abkürzungspunkt wird ignoriert.
  - **"Zflg\_1" .AB. "Zflg\_2", "Zflg\_3", ...**  
Zflg\_1 stimmt mit mindestens einer der Zeichenfolgen Zflg\_2, Zflg\_3, ... überein oder Zflg\_1 ist eine Abkürzung von mindestens einer der Zeichenfolgen Zflg\_2, Zflg\_3, ... Ein am Ende von Zflg\_1 evtl. vorhandener Abkürzungspunkt wird ignoriert.
  - **"Zflg\_1" .NA. "Zflg\_2"**  
Zflg\_1 und Zflg\_2 sind weder gleich noch ist Zflg\_1 eine Abkürzung von Zflg\_2. Ein am Ende von Zflg\_1 evtl. vorhandener Abkürzungspunkt wird ignoriert.



- **"Zflg\_1" .NA. "Zflg\_2", "Zflg\_3", ...**

Zflg\_1 stimmt weder mit einer der Zeichenfolgen Zflg\_2, Zflg\_3, ... überein noch ist Zflg\_1 eine Abkürzung von einer der Zeichenfolgen Zflg\_2, Zflg\_3, ... Ein am Ende von Zflg\_1 evtl. vorhandener Abkürzungspunkt wird ignoriert.

- **"Zflg\_1" .HN. "Zflg\_2"**

Zflg\_1 ist ein Tag, das den Namen Zflg\_2 hat.

- **"Zflg\_1" .HA. "Zflg\_2"**

Zflg\_1 ist ein Tag, das ein Attribut Zflg\_2 hat.

Ein Tag ist eine in spitzen Klammern eingeschlossene Zeichenfolge; siehe »Makrofunktionen für Tags« Seite 559.

## Prüfen von Zeichenfolgen mit Hilfe von Schlüsselwörtern

Die Bedingungen zum Prüfen von Zeichenfolgen haben die Form:

`"Zeichenfolge" .EQ. Schlüsselwort`

Dabei kann eines der nachfolgend genannten Schlüsselwörter angegeben werden. An Stelle von `.EQ.` kann auch `.NE.` stehen. Mit `.NE.` ist eine Bedingung dann erfüllt, wenn sie mit `.EQ.` nicht erfüllt ist und umgekehrt.

Wurde mit der `MODE`-Anweisung (siehe ab Seite 413) der Modus `VARIABLE` oder `TUSCRIPT` eingestellt, muss das Schlüsselwort mit Apostrophen gekennzeichnet werden; an Stelle der in Anführungszeichen eingeschlossenen Zeichenfolge kann auch der Name einer Variablen angegeben werden:

`Variablenname .EQ. 'Schlüsselwort'`

In diesem Fall wird die in der angegebenen Variablen enthaltene Zeichenfolge entsprechend geprüft.

Folgende Schlüsselwörter sind vorgesehen:

- **DIGITS** oder **ZIFFERN**

Die Zeichenfolge besteht nur aus Ziffern (0 bis 9).

- **NUMBER** oder **ZAHL**

Die Zeichenfolge ist eine arabische Zahl (Zeichenfolge aus Ziffern von 0 bis 9) mit oder ohne Vorzeichen.

- **ROMAN**

Die Zeichenfolge ist eine römische Zahl (gebildet aus den Groß- oder Kleinbuchstaben I, V, X, L, C, D und M).

**– NAME**

Die Zeichenfolge entspricht den Konventionen für einen Namen (z. B. Variablennamen, Segmentnamen) in TUSTEP.

**– VARIABLE**

Die Zeichenfolge ist der Name einer zuvor definierten und noch nicht wieder freigegebenen Makrovariablen.

**– C\_GROUP**

Die Zeichenfolge ist der Name einer zuvor mit der BUILD-Anweisung definierten und noch nicht wieder freigegebenen Zeichengruppe.

**– S\_GROUP**

Die Zeichenfolge ist der Name einer zuvor mit der BUILD-Anweisung definierten und noch nicht wieder freigegebenen Stringgruppe.

**– S\_TABLE**

Die Zeichenfolge ist der Name einer zuvor mit der BUILD-Anweisung definierten und noch nicht wieder freigegebenen Such-Tabelle.

**– X\_TABLE**

Die Zeichenfolge ist der Name einer zuvor mit der BUILD-Anweisung definierten und noch nicht wieder freigegebenen Austausch-Tabelle.

**– R\_TABLE**

Die Zeichenfolge ist der Name einer zuvor mit der BUILD-Anweisung definierten und noch nicht wieder freigegebenen Recherchier-Tabelle.

**– DICTIONARY**

Die Zeichenfolge ist der Name eines zuvor definierten und noch nicht wieder freigegebenen Wörterbuchs.

**– STACK**

Die Zeichenfolge ist der Name eines zuvor definierten und noch nicht wieder freigegebenen Stapelspeichers.

**– SECTION**

Die Zeichenfolge ist der Name einer zuvor definierten Sektion.

**– SUBMACRO**

Die Zeichenfolge ist der Name eines zuvor definierten Submakros.

**– PRINTER oder DRUCKER**

Die Zeichenfolge ist der Name einer System-Variablen, die den Namen eines Druckers enthält; falls keine System-Variable mit diesem Namen definiert ist, entspricht die Zeichenfolge dem Namen eines Druckers; d. h. die Zeichenfolge ist eine legale Angabe zur Spezifikation GERAET beim Kommando #DRUCKE (siehe Seite 142).

Einschränkung: Diese Abfrage ist z. Z. nur unter Windows möglich.

– **TITLE**

Die Zeichenfolge ist der Titel eines Fensters.

Einschränkung: Diese Abfrage ist z. Z. nur unter Windows möglich.

## Prüfen von Zeichenfolgen mit Hilfe von Tabellen

Die Bedingungen zum Prüfen von Zeichenfolgen mit Tabellen haben die Form:

```
"Zeichenfolge" .xx. Tabelle
```

Wurde mit der `MODE`-Anweisung (siehe ab Seite 413) der Modus `VARIABLE` oder `TUSCRIPT` eingestellt, kann an Stelle der in Anführungszeichen eingeschlossenen Zeichenfolge auch der Name einer Variablen angegeben werden:

```
Variablenname .xx. Tabelle
```

In diesem Fall wird die in der angegebenen Variablen enthaltene Zeichenfolge entsprechend geprüft.

Für »Tabelle« kann entweder der Name einer mit der `BUILD`-Anweisung definierten Tabelle oder direkt eine in Anführungszeichen eingeschlossenen Tabelle angegeben werden:

```
Variablenname .xx. ":Zflg_1:Zflg_2:...:"
```

Wird auf diese Weise eine Recherchier-Tabelle angegeben, so werden automatisch die Optionen `TEXT` und `OR` angenommen. Falls andere Optionen (siehe Seite 441) erforderlich sind, muss die Tabelle mit der `BUILD`-Anweisung definiert und der Name der Tabelle angegeben werden.

– **"Zflg" .CT. Such-Tabelle**

`Zflg` enthält mindestens eine der in der Such-Tabelle angegebenen Zeichenfolgen.

– **"Zflg" .CT. Such-Tabelle\_1, Such-Tabelle\_2, ...**

`Zflg` enthält mindestens eine der in den Such-Tabellen angegebenen Zeichenfolgen.

– **"Zflg" .NC. Such-Tabelle**

`Zflg` enthält keine der in der Such-Tabelle angegebenen Zeichenfolgen.

– **"Zflg" .NC. Such-Tabelle\_1, Such-Tabelle\_2, ...**

`Zflg` enthält keine der in den Such-Tabellen angegebenen Zeichenfolgen.

– **"Zflg" .MA. Recherchier-Tabelle**

`Zflg` stimmt mit mindestens einer der in der Recherchier-Tabelle angegebenen Zeichenfolgen überein.

- **"Zflg" .MA. Recherchier-Tabelle\_1, Recherchier-Tabelle\_2, ...**  
Zflg stimmt mit mindestens einer der in den Recherchier-Tabellen angegebenen Zeichenfolgen überein.
- **"Zflg" .NM. Recherchier-Tabelle**  
Zflg stimmt mit keiner der in der Recherchier-Tabelle angegebenen Zeichenfolgen überein.
- **"Zflg" .NM. Recherchier-Tabelle\_1, Recherchier-Tabelle\_2, ...**  
Zflg stimmt mit keiner der in den Recherchier-Tabellen angegebenen Zeichenfolgen überein.
- **"Zflg" .CO. Such-Tabelle**  
Zflg enthält genau eine der in der Such-Tabelle angegebenen Zeichenfolgen.
- **"Zflg" .CO. Such-Tabelle\_1, Such-Tabelle\_2, ...**  
Zflg enthält in mindestens einer der Such-Tabellen genau eine der angegebenen Zeichenfolgen.
- **"Zflg" .CA. Such-Tabelle**  
Zflg enthält alle der in der Such-Tabelle angegebenen Zeichenfolgen.
- **"Zflg" .CA. Such-Tabelle\_1, Such-Tabelle\_2, ...**  
Zflg enthält in mindestens einer der Such-Tabellen alle angegebenen Zeichenfolgen.
- **"Zflg" .CE. Such-Tabelle**  
Zflg enthält jede in der Such-Tabelle angegebenen Zeichenfolgen genau ein Mal.
- **"Zflg" .CE. Such-Tabelle\_1, Such-Tabelle\_2, ...**  
Zflg enthält in mindestens einer der Such-Tabellen jede der in der Such-Tabelle angegebenen Zeichenfolgen genau ein Mal.
- **"Zflg" .CS. Such-Tabelle**  
Zflg enthält jede der in der Such-Tabelle angegebenen Zeichenfolgen in der Reihenfolge, wie sie in der Such-Tabelle angegeben sind, genau ein Mal.
- **"Zflg" .CS. Such-Tabelle\_1, Such-Tabelle\_2, ...**  
Zflg enthält in mindestens einer der Such-Tabellen jede der in der Such-Tabelle angegebenen Zeichenfolgen in der Reihenfolge, wie sie in der Such-Tabelle angegeben sind, genau ein Mal.

## Prüfen von Projektnamen, Dateinamen und Dateien

Die Bedingungen zum Prüfen von Projektnamen, Dateinamen und Dateien haben die Form:

`"Zeichenfolge" .EQ. Schlüsselwort`

Dabei kann eines der nachfolgend genannten Schlüsselwörter angegeben werden. An Stelle von `.EQ.` kann auch `.NE.` stehen. Mit `.NE.` ist eine Bedingung dann erfüllt, wenn sie mit `.EQ.` nicht erfüllt ist und umgekehrt.

Wurde mit der `MODE`-Anweisung (siehe ab Seite 413) der Modus `VARIABLE` oder `TUSCRIPT` eingestellt, muss das Schlüsselwort mit Apostrophen gekennzeichnet werden; an Stelle der in Anführungszeichen eingeschlossenen Zeichenfolge kann auch der Name einer Variablen angegeben werden:

`Variablenname .EQ. 'Schlüsselwort'`

In diesem Fall wird die in der angegebenen Variablen enthaltene Zeichenfolge entsprechend geprüft.

Wird für das angegebene Schlüsselwort als Zeichenfolge ein Dateiname erwartet, so kann dieser um einen Projektnamen ergänzt und in der Form »Projektname\*Dateiname« angegeben werden. Wird kein Projektname oder wird der Projektname nur teilweise angegeben, wird er automatisch ergänzt bzw. vervollständigt (siehe »Ergänzen des Projektnamens« Seite 26)

Zum Prüfen eines Dateiinhalts muss auf diesen zugegriffen werden. Dies ist jedoch nicht möglich, wenn schon ein Programm diese Datei mit schreibendem Zugriff belegt hat; die Bearbeitung des Makros wird in diesem Fall abgebrochen. Soll erst eine bestimmte Zeit gewartet werden, bis die Datei wieder frei ist, kann dies durch einen vorangehenden Aufruf der Makrofunktion `LOCK` (siehe Seite 490) erreicht werden. Dabei wird die Datei gleichzeitig für andere Zugriffe gesperrt; sie muss durch einen nachfolgenden Aufruf der Makrofunktion `UNLOCK` wieder freigegeben werden.

Folgende Schlüsselwörter sind vorgesehen:

– **VOLUME**

Die Zeichenfolge ist der Name einer System-Variablen, deren Inhalt einen existierenden Träger (Pfad) bezeichnet.

Unter Windows kann die Zeichenfolge auch ein Laufwerksbuchstabe eines existierenden Laufwerks sein.

– **PROJECT\_NAME** oder **PROJEKTNAME**

Die Zeichenfolge entspricht den Konventionen für einen Projektnamen in TUSTEP (unabhängig davon, ob ein Projekt mit diesem Namen existiert oder nicht).

– **PROJECT** oder **PROJEKT**

Die Zeichenfolge entspricht den Konventionen für einen Projektnamen in TUSTEP und ist der Name eines existierenden Projekts (unabhängig davon, ob Dateien dazu existieren oder nicht) auf dem mit der System-Variablen `TUSTEP_DSK` vorgegebenen Träger.

– **PROJECT .ON. traeger** oder **PROJEKT .ON. traeger**

Die Zeichenfolge entspricht den Konventionen für einen Projektnamen in TUSTEP und ist der Name eines existierenden Projekts (unabhängig davon, ob Dateien dazu existieren oder nicht) auf dem mit der System-Variablen `traeger` vorgegebenen Träger.

Für `traeger` muss der Name einer System-Variablen angegeben werden, die den Pfad für das abzufragende Projekt enthält. Unter Windows kann auch direkt ein Laufwerksbuchstabe angegeben werden, falls der Pfad keinen Verzeichnisnamen enthält.

– **FILE\_NAME** oder **DATEINAME**

Die Zeichenfolge entspricht den Konventionen für einen Dateinamen in TUSTEP (unabhängig davon, ob eine Datei mit diesem Namen existiert oder nicht).

– **FILE** oder **DATEI**

Die Zeichenfolge entspricht den Konventionen für einen Dateinamen in TUSTEP und ist der Name einer existierenden Datei (unabhängig davon, ob die Datei mit diesem Namen angemeldet ist oder nicht) auf dem Standard-Träger für permanente Dateien; dieser ist durch die System-Variable `TUSTEP_DSK` vorgegeben.

– **FILE .ON. traeger** oder **DATEI .ON. traeger**

Die Zeichenfolge entspricht den Konventionen für einen Dateinamen in TUSTEP und ist der Name einer existierenden Datei (unabhängig davon, ob die Datei mit diesem Namen angemeldet ist oder nicht) auf dem mit der System-Variablen `traeger` vorgegebenen Träger.

Für `traeger` muss der Name einer System-Variablen, die den Pfad für die abzufragende Datei enthält, oder ein Minuszeichen angegeben werden. Wird ein Minuszeichen angegeben, so muss der Name der abzufragenden Datei zuvor definiert worden sein (siehe `DEFINE`-Anweisung Seite 423).

Unter Windows kann für das Argument `traeger` auch direkt ein Laufwerksbuchstabe angegeben werden, falls der Pfad keinen Verzeichnisnamen enthält.

– **READ** oder **LESEN**

Die Zeichenfolge ist der Name einer Scratch-Datei oder der Name einer Datei, die (auf einem beliebigen Träger) zum Lesen oder zum Schreiben (und damit auch zum Lesen) angemeldet ist.

– **READ .ON. traeger**

Die Zeichenfolge ist der Name einer Datei, die auf dem mit der System-Variablen `traeger` vorgegebenen Träger zum Lesen oder zum Schreiben (und damit auch zum Lesen) angemeldet ist.

Für `traeger` muss der Name einer System-Variablen angegeben werden, die den Pfad für die abzufragende Datei enthält. Unter Windows kann auch direkt ein Laufwerksbuchstabe angegeben werden, falls der Pfad keinen Verzeichnisnamen enthält.

– **WRITE** oder **SCHREIBEN**

Die Zeichenfolge ist der Name einer Scratch-Datei oder der Name einer Datei, die (auf einem beliebigen Träger) zum Schreiben angemeldet ist.

– **WRITE .ON. traeger**

Die Zeichenfolge ist der Name einer Datei, die auf dem mit der System-Variablen `traeger` vorgegebenen Träger zum Schreiben angemeldet ist.

Für `traeger` muss der Name einer System-Variablen angegeben werden, die den Pfad für die abzufragende Datei enthält. Unter Windows kann auch direkt ein Laufwerksbuchstabe angegeben werden, falls der Pfad keinen Verzeichnisnamen enthält.

– **SCRATCH**

Die Zeichenfolge ist der Name einer Scratch-Datei (temporären Datei).

– **SEQ**

Die Zeichenfolge ist der Name einer Datei vom Typ `SEQ`, die angemeldet ist.

– **RAN**

Die Zeichenfolge ist der Name einer Datei vom Typ `RAN`, die angemeldet ist.

– **FDF**

Die Zeichenfolge ist der Name einer Datei vom Typ `FDF`, die angemeldet ist.

– **TAPE**

Die Zeichenfolge ist der Name einer Band-Datei, die angemeldet ist.

– **EMPTY** oder **LEER**

Die Zeichenfolge ist der Name einer leeren Datei, die angemeldet ist.

– **SORTED**

Zeichenfolge ist der Name einer TUSTEP-Datei, die angemeldet ist, und in der die Sätze nach der Satznummer in aufsteigender Reihenfolge sortiert sind, wobei keine Satznummer mehrfach vorkommt.

– **OK**

Die Zeichenfolge ist der Name einer TUSTEP-Datei, die angemeldet und korrekt abgeschlossen ist, oder einer Fremd-Datei, die angemeldet ist.

– **SEGMENT .IN. "datei"**

Die angegebene Datei ist eine Segment-Datei, die angemeldet und korrekt abgeschlossen ist, und die Zeichenfolge ist entweder leer oder ist der Name eines Segments in dieser Datei.

– **BROWSE**

Die Zeichenfolge ist der Name einer Datei, die angemeldet ist, oder der vollständige Pfad samt Name einer existierenden Datei, deren Name eine Endung hat, für die im Betriebssystem eine Anwendung definiert ist.

## Abfragen von Einstellungen

Die Bedingungen zum Abfragen von Einstellungen haben die Form:

Schlüsselwort `.EQ.` Schlüsselwort

Dabei kann eine der nachfolgend genannten Schlüsselwort-Kombinationen angegeben werden. An Stelle von `.EQ.` kann auch `.NE.` stehen. Mit `.NE.` ist eine Bedingung dann erfüllt, wenn sie mit `.EQ.` nicht erfüllt ist und umgekehrt.

Wurde mit der `MODE`-Anweisung (siehe ab Seite 413) der Modus `VARIABLE` oder `TUSCRIPT` eingestellt, müssen die Schlüsselwörter mit Apostrophen gekennzeichnet werden:

'Schlüsselwort' `.EQ.` 'Schlüsselwort'

Folgende Schlüsselwort-Kombinationen sind vorgesehen:

- **ERROR\_STOP .EQ. ON** oder **FEHLERHALT .EQ. EIN**  
Fehlerhalt ist auf `EIN` gestellt.
- **ERROR\_STOP .EQ. OFF** oder **FEHLERHALT .EQ. AUS**  
Fehlerhalt ist auf `AUS` gestellt.
- **ERROR\_STOP .EQ. SIGNAL** oder **FEHLERHALT .EQ. SIGNAL**  
Fehlerhalt ist auf `SIGNAL` gestellt.
- **ERROR\_STOP .EQ. DELETE** oder **FEHLERHALT .EQ. LOESCHEN**  
Fehlerhalt ist auf `LOESCHEN` gestellt.
- **PARAMETER .EQ. ON** oder **PARAMETER .EQ. EIN**  
Parameter-Protokollierung ist eingeschaltet.
- **PARAMETER .EQ. OFF** oder **PARAMETER .EQ. AUS**  
Parameter-Protokollierung ist ausgeschaltet.
- **PARAMETER .EQ. OLD** oder **PARAMETER .EQ. ALT**  
Alte Parameter-Interpretation ist eingestellt.
- **PARAMETER .EQ. NEW** oder **PARAMETER .EQ. NEU**  
Neue Parameter-Interpretation ist eingestellt.



- **PARAMETER .EQ. <>**  
Parameter-Interpretation »<>« ist eingestellt.
- **PARAMETER .EQ. {}**  
Parameter-Interpretation »{}« ist eingestellt.
- **JOURNAL .EQ. ON** oder **PROTOKOLL .EQ. EIN**  
Zweitprotokoll ist eingeschaltet.
- **JOURNAL .EQ. OFF** oder **PROTOKOLL .EQ. AUS**  
Zweitprotokoll ist ausgeschaltet.
- **JOURNAL .EQ. SYSTEM** oder **PROTOKOLL .EQ. SYSTEM**  
Protokoll ist auf SYSTEM eingestellt.
- **JOURNAL .EQ. SCROLLING** oder **PROTOKOLL .EQ. FREILAUFEND**  
Protokoll ist auf FREILAUFEND eingestellt.
- **JOURNAL .EQ. PARTITIONED** oder **PROTOKOLL .EQ. PORTIONIERT**  
Protokoll ist auf PORTIONIERT eingestellt.
- **JOURNAL .EQ. +** oder **PROTOKOLL .EQ. +**  
Ablauf-Protokollierung ist eingeschaltet.
- **JOURNAL .EQ. -** oder **PROTOKOLL .EQ. -**  
Ablauf-Protokollierung ist ausgeschaltet.
- **JOURNAL .EQ. /** oder **PROTOKOLL .EQ. /**  
Ablauf-Protokollierung ist für das TUSTEP-Fenster ausgeschaltet, für das Zweitprotokoll eingeschaltet.

## Status-Abfragen

Bei den folgenden Bedingungen kann **.NOT.** vorangestellt werden, um die Bedingung umzukehren. Mit **.NOT.** ist eine Bedingung dann erfüllt, wenn sie sonst nicht erfüllt ist und umgekehrt.

Beispiel: `$$ IF (.NOT. ERROR) DEFINE var`

- **WSn**  
Wahlschalter n (n = 1 bis 7) gesetzt.
- **BATCH**  
Aufruf des Makros erfolgte im Batch-Modus.

- **DIALOG**  
Aufruf des Makros erfolgte im Dialog-Modus.
- **REMOTE**  
Aufruf des Makros erfolgte in einer REMOTE-Sitzung.
- **WINRMT**  
Aufruf des Makros erfolgte in einer REMOTE-Sitzung, die von Windows aus aufgerufen wurde.
- **CGI**  
Aufruf des Makros erfolgte als CGI-Makro (siehe Kapitel »Starten von TUSTEP durch einen WWW-Server« ab Seite 98).
- **CMD**  
Aufruf des Makros erfolgte als CMD-Makro (siehe Kapitel »Starten von TUSTEP als System-Kommando« ab Seite 89).
- **GERMAN**  
Kommandos und Editor-Anweisungen werden deutsch-sprachig erwartet.
- **ENGLISH**  
Kommandos und Editor-Anweisungen werden englisch-sprachig erwartet.
- **ERROR**  
Makroanweisung `ERROR` wurde schon ausgeführt oder eine Makroanweisung enthielt Fehler.
- **MODIFIED**  
Bei der zuletzt ausgeführten `EDIT`-Anweisung (siehe Seite 644) wurde der Feldinhalt verändert.
- **BACKTAB**  
Bei der zuletzt ausgeführten `EDIT`-Anweisung (siehe Seite 644) wurde die Eingabe mit `BACKTAB` abgeschlossen.
- **CANCEL**  
Bei der zuletzt ausgeführten `EDIT`-Anweisung (siehe Seite 644) wurde die Eingabe mit `CANCEL`, mit `CR` in einem Feld vom Typ `BUTTON/CANCEL` oder durch Anklicken eines Feldes vom Typ `BUTTON/CANCEL` abgeschlossen.
- **CLICK**  
Bei der zuletzt ausgeführten `EDIT`-Anweisung (siehe Seite 644) wurde die Eingabe durch Anklicken eines anderen Feldes (d. h. eines Feldes mit Ausnahme des gerade aktiven Feldes, in dem der Cursor steht), das nicht vom Typ `BUTTON/ENTER`, `BUTTON/CANCEL`, `BUTTON/HELP`, `BUTTON/FKEY` ist, abgeschlossen.

**– CR**

Bei der zuletzt ausgeführten EDIT-Anweisung (siehe Seite 644) wurde die Eingabe mit CR abgeschlossen.

**– DEL**

Bei der zuletzt ausgeführten EDIT-Anweisung (siehe Seite 644) wurde die Eingabe mit DEL abgeschlossen.

**– ENTER**

Eingabe wurde mit ENTER, mit CR in einem Feld vom Typ BUTTON/ENTER oder durch Anklicken eines Feldes vom Typ BUTTON/ENTER abgeschlossen.

**– FKEY**

Bei der zuletzt ausgeführten EDIT-Anweisung (siehe Seite 644) wurde die Eingabe mit einer Funktionstaste (F1 bis F60), mit CR in einem Feld vom Typ BUTTON/FKEY oder durch Anklicken eines Feldes vom Typ BUTTON/FKEY abgeschlossen.

**– Fn**

Bei der zuletzt ausgeführten EDIT-Anweisung (siehe Seite 644) wurde die Eingabe mit der Funktionstaste Fn (n = 1 bis 60) abgeschlossen.

**– HELP**

Bei der zuletzt ausgeführten EDIT-Anweisung (siehe Seite 644) wurde die Eingabe mit HELP, mit CR in einem Feld vom Typ BUTTON/HELP oder durch Anklicken eines Feldes vom Typ BUTTON/HELP abgeschlossen.

**– TAB**

Bei der zuletzt ausgeführten EDIT-Anweisung (siehe Seite 644) wurde die Eingabe mit TAB abgeschlossen.

**– WAIT**

Bei der vorangehenden ACCESS-Anweisung (siehe Seite 598) war der gewünschte Zugriff innerhalb des angegebenen Zeitlimits nicht möglich.

Hinweis: Eine Abfrage dieser Bedingung ist nur sinnvoll, wenn in der ACCESS-Anweisung ein Zeitlimit angegeben wurde (siehe Beispiel Seite 599).

**– SOR**

Die bei der vorangehenden READ-Anweisung gelesene Datenportion begann in der Datei am Satzanfang (Start Of Record).

**– EOR**

Die bei der vorangehenden READ-Anweisung gelesene Datenportion endete in der Datei am Satzende (End Of Record).

– **EOF**

Für die vorangehenden `READ`-Anweisung waren keine Daten mehr vorhanden bzw. bei der vorangehenden `FIND`- oder `COUNT`-Anweisung wurden keine entsprechenden Daten gefunden.

– **EXIST**

Bei der vorangehenden `READ`/`CHECK`-Anweisung waren entsprechende Daten vorhanden.

– **DONE**

Bei der vorangehenden `UPLOAD`- bzw. `DOWNLOAD`-Anweisung (siehe Kapitel »Datei-Transfer« ab Seite 639) konnte die Datei übertragen werden.

## Rechenanweisungen

Im Folgenden sind die Rechenanweisungen beschrieben, die bei Wertzuweisungen rechts vom Gleichheitszeichen verwendet werden können.

Beispiel: `$$ SET nummer = nummer + 1`

Das jeweilige Ergebnis der Rechenanweisung wird der links vom Gleichheitszeichen stehenden Variablen zugewiesen. Eine Rechenanweisung ist eine Rechenvorschrift, durch die ein numerischer Wert bestimmt wird. Sie besteht aus Operanden, arithmetischen Operatoren und Klammerpaaren.

Ein Operand kann eine (ganze) Zahl, der Name einer Variablen, die eine Zahl enthält, oder ein Funktionsaufruf sein. Im einfachsten Fall besteht eine Rechenanweisung nur aus einer dieser drei Angaben.

Es stehen folgende arithmetischen Operatoren zur Verfügung:

- + für die Addition
- für die Subtraktion
- \* für die Multiplikation
- / für die Division
- % für das Berechnen des Divisionsrestes

Bei der Auswertung der Rechenanweisung werden Multiplikation und Division vor Addition und Subtraktion ausgeführt. Folgen mehrere Multiplikationen und/oder Divisionen aufeinander, so werden diese von links nach rechts ausgewertet. Das gleiche gilt für aufeinander folgende Additionen und/oder Subtraktionen. Soll von dieser Regelung abgewichen werden, so kann die Reihenfolge durch Setzen von Klammern festgelegt werden.

Bei der Division ist zu beachten, dass grundsätzlich nur mit ganzen Zahlen gerechnet wird. Der Divisionsrest geht verloren; eine Rundung findet nicht statt. Dies hat zur Folge, dass z. B.  $3/2$  den Wert 1 (nicht 1,5) und  $3/2*4$  den Wert 4 (nicht 6) ergibt.

Steht (wie im obigen Beispiel) unmittelbar rechts vom Gleichheitszeichen die gleiche Variable wie links vom Gleichheitszeichen und danach ein arithmetischer Operator, so kann die Anweisung verkürzt geschrieben werden:

Beispiel: `$$ SET nummer += 1`

Bei den Operatoren für Multiplikation, Division und Divisionsrest muss jedoch beachtet werden, dass

```
$$ SET nummer *= basis + anzahl
```

gleichbedeutend ist zu

```
$$ SET nummer = nummer * (basis + anzahl)
```

und nicht zu

```
$$ SET nummer = nummer * basis + anzahl
```

Zur Unterstützung von Berechnungen stehen Funktionen zur Verfügung. Sie dürfen nicht mit Makrofunktionen verwechselt werden. Eine Funktion wird mit ihrem Na-

men und einem oder mehreren Argumenten aufgerufen. Die Argumente müssen durch Komma getrennt in Klammern hinter dem Funktionsnamen angegeben werden. Als Argumente dieser Funktionen werden nur Zahlen erwartet. Durch den Aufruf einer Funktion mit diesen Argumenten wird nach der Funktionsvorschrift ein Zahlenwert, der Funktionswert, errechnet.

Für Rechenanweisungen stehen folgende Funktionen zur Verfügung:

- Berechnen des Minimums: `MIN (arg1, arg2, arg3, ...)`

Der Funktionswert ist der kleinste Wert der Zahlenwerte `arg1, arg2, arg3, ...`

Die Anzahl der Argumente ist bei dieser Funktion beliebig.

Beispiel: Der Funktionswert von `MIN (-5, +3)` ist `-5`.

- Berechnen des Maximums: `MAX (arg1, arg2, arg3, ...)`

Der Funktionswert ist der größte Wert der Zahlenwerte `arg1, arg2, arg3, ...`. Die

Anzahl der Argumente ist bei dieser Funktion beliebig.

Beispiel: Der Funktionswert von `MAX (-5, +3)` ist `+3`.

- Berechnen des Divisionsrestes: `MOD (arg1, arg2)`

Der Funktionswert ist der Rest, der sich bei der Division von `arg1` durch `arg2` ergibt.

Beispiel: Der Funktionswert von `MOD (234, 10)` ist `4`.

- Intervall-Funktion: `INT (arg, arg1, arg2, arg3, ..., argn)`

Mit dieser Funktion kann festgestellt werden, in welches Intervall der Zahlenwert `arg` fällt. Der Funktionswert ist Null, falls `arg` kleiner als `arg1` ist; er ist 1, falls `arg` größer oder gleich `arg1` aber kleiner als `arg2` ist; er ist 2, falls `arg` größer oder gleich `arg2` aber kleiner als `arg3` ist, ..., den Wert `n`, falls `arg` größer oder gleich `argn` ist. Die Anzahl der Argumente ist bei dieser Funktion beliebig, jedoch muss `arg1` kleiner als `arg2` sein, `arg2` kleiner als `arg3` sein usw.

Beispiel: Der Funktionswert von `INT (16, 1, 10, 100, 1000)` ist `2`.

## Makrofunktionen

Im Folgenden sind die Makrofunktionen beschrieben, die bei Wertzuweisungen verwendet werden können.

Beispiel: `$$ SET dateinamen = FILES ()`

Der jeweilige Funktionswert der Makrofunktion wird der links vom Gleichheitszeichen stehenden Variablen zugewiesen. Rechts vom Gleichheitszeichen sind bei Angabe einer Makrofunktion außer den ggf. vorgesehenen Argumenten keine weiteren Angaben erlaubt.

Als Argumente werden Variablennamen, Tabellennamen, Zahlenwerte und in Anführungszeichen eingeschlossene Zeichenfolgen erwartet. Variablen muss zuvor ein Wert zugewiesen worden sein. Falls im Einzelfall nichts anderes angegeben ist, dürfen die Variablen keine Sternvariablen sein. Such-, Austausch-, und Recherchier-Tabellen können zuvor mit der `BUILD`-Anweisung definiert werden oder direkt in Anführungszeichen angegeben werden.

Achtung: Soll an Stelle eines Zahlenwertes der Inhalt einer Variablen verwendet werden, so kann der Name dieser Variablen mit einem Nummernzeichen unmittelbar vor dem Namen gekennzeichnet werden; falls jedoch für das entsprechende Argument auch der Name einer Tabelle angegeben werden darf, muss der Variablenname mit einem Nummernzeichen gekennzeichnet werden, damit er nicht als Tabellename interpretiert wird.

Einige Makrofunktionen interpretieren den Inhalt einer Variablen nicht als eine einzige Zeichenfolge, sondern als einzelne Teilzeichenfolgen. Dazu müssen die Teilzeichenfolgen durch Apostroph getrennt sein. Teilzeichenfolgen können auch leer sein.

### Makrofunktionen zur Dateiverwaltung

Die Leistungen von einigen der folgenden Makroanweisungen entsprechen den Leistungen der Kommandos `#DATEI`, `#ANMELDE`, `#ABMELDE`, `#LOESCHE` und `#AENDERE`, die mit der `EXECUTE`-Anweisung (siehe Seite 456) ausgeführt werden können. Bei Verwendung dieser Makroanweisungen kann einerseits bequemer abgefragt werden, ob die Leistung fehlerfrei erbracht wurde, und andererseits kann die Fehlermeldung ggf. in einem Makrofenster angezeigt werden, da sie nicht ins Ablaufprotokoll ausgegeben wird, sondern in eine Variable abgespeichert wird.

Falls beim Aufruf einer der folgenden Funktionen das Fehlerflag zuvor gesetzt wurde (z. B. explizit durch die Makroanweisung `ERROR` oder automatisch bei Auftreten eines Syntaxfehlers) wird die Leistung der Makrofunktion nicht erbracht; der Funktionswert liefert in diesem Fall eine entsprechende Fehlermeldung.

`CREATE (name, typ, traeger)`

Die Makrofunktion `CREATE` richtet ein Projekt (Verzeichnis) oder eine Datei ein.

Für das Argument `name` muss ein in Anführungszeichen eingeschlossener Datei- bzw. Projektname oder eine Variable angegeben werden, die den Datei- bzw. Projektnamen enthält.

Achtung: Bei dieser Makrofunktion wird der angegebene Dateiname nur dann als »definierter Dateiname« angesehen, wenn für das Argument `traeger` ein Minuszeichen angegeben ist.

Für das Argument `traeger` muss entweder der Name einer System-Variablen, die den Pfad für das einzurichtende Projekt bzw. die einzurichtende Datei enthält, oder `-STD-` oder ein Minuszeichen angegeben werden. Die Angabe `-STD-` ist beim Einrichten einer temporären Datei (Scratch-Datei) gleichbedeutend mit der Angabe der System-Variablen `TUSTEP_SCR`, die den Standard-Träger für temporäre Dateien enthält; andernfalls ist die Angabe `-STD-` gleichbedeutend mit der Angabe der System-Variablen `TUSTEP_DSK`, die den Standard-Träger für permanente Dateien enthält. Wird für das Argument `traeger` ein Minuszeichen angegeben, so muss der mit dem Argument `name` angegebene Dateiname zuvor definiert worden sein (siehe `DEFINE`-Anweisung Seite 423); beim Einrichten eines Projekts darf für das Argument `traeger` kein Minuszeichen angegeben werden. Das Argument `traeger` kann einschließlich des davor stehenden Kommas weggelassen werden; in diesem Fall wird als Träger `-STD-` angenommen.

Unter Windows kann für das Argument `traeger` auch direkt ein Laufwerksbuchstabe angegeben werden, falls der Pfad keinen Verzeichnisnamen enthält.

Ist für das Argument `typ` das Schlüsselwort `PROJECT` angegeben, so wird das mit dem Argument `name` angegebene Projekt eingerichtet. Für `name` kann ein in Anführungszeichen eingeschlossener Projektname oder eine Variable angegeben werden, die den Projektnamen enthält.

Falls das Projekt eingerichtet werden konnte, ist der Funktionswert »OK«; andernfalls wird als Funktionswert eine entsprechende Fehlermeldung geliefert.

```
Beispiel: $$ SET verz = "name\tst"
          $$ SET status = CREATE (verz, PROJECT, -STD-)
```

Ergebnis: status = "OK" oder Fehlermeldung

Ist für das Argument `typ` das Schlüsselwort `PROJECT` und mit dem Argument `name` kein Projektname, sondern »-« angegeben, so werden diejenigen Verzeichnisse eingerichtet, die für den mit dem Argument `traeger` angegebenen Pfad noch nicht existieren.

Ist für das Argument `typ` ein Dateityp mit einer Option angegeben (Dateityp und Option müssen durch ein Minuszeichen oder einen Schrägstrich getrennt sein), wird eine Datei eingerichtet und diese gleichzeitig zum Schreiben angemeldet. Für das Argument `name` muss in diesem Fall ein in Anführungszeichen eingeschlossener Dateiname oder eine Variable angegeben werden, die den Dateinamen enthält.

Als Dateityp kann das Schlüsselwort `SEQ` für TUSTEP-Dateien mit sequentiellem Zugriff, `RAN` für TUSTEP-Dateien mit wahlfreiem (random) Zugriff, `TAPE` für Band-Dateien oder `FDF` für Fremd-Dateien im Fremd-Daten-Format (z. B. ASCII-Dateien) angegeben werden.

Wird die Option `P` oder `T` angegeben, so wird eine permanente bzw. temporäre Datei eingerichtet.



Wird die Option `O` (= `OPEN`) angegeben, so wird erst versucht, eine permanente Datei mit dem angegebenen Namen zum Schreiben anzumelden; existiert noch keine permanente Datei mit dem angegebenen Namen, so wird eine solche eingerichtet.

Wird die Option `E` (= `ERASE`) angegeben, so wird erst versucht, in einer temporären Datei mit dem angegebenen Namen die Daten zu löschen; existiert noch keine temporäre Datei mit dem angegebenen Namen, so wird eine solche eingerichtet.

Falls die Datei eingerichtet (Option `O` oder `E`) bzw. angemeldet (Option `O`) werden konnte bzw. die Daten gelöscht werden konnten (Option `E`), ist der Funktionswert »OK«; andernfalls wird als Funktionswert eine entsprechende Fehlermeldung geliefert.

```
Beispiel: $$ SET datei = "testdaten"
          $$ SET status = CREATE (datei, SEQ-O, -STD-)

          Ergebnis: status = "OK" oder Fehlermeldung
```

`OPEN` (name, modus, traeger)

Die Makrofunktion `OPEN` meldet eine permanente Datei an.

Für das Argument `name` muss ein in Anführungszeichen eingeschlossener Dateiname oder eine Variable angegeben werden, die den Dateinamen enthält.

Achtung: Bei dieser Makrofunktion wird der angegebene Dateiname nur dann als »definierter Dateiname« angesehen, wenn für das Argument `traeger` ein Minuszeichen angegeben ist.

Für das Argument `modus` muss die Zeichenfolge `READ` angegeben werden, falls die Datei zum Lesen angemeldet werden soll, und die Zeichenfolge `WRITE`, falls die Datei zum Schreiben angemeldet werden soll.

Für das Argument `traeger` muss entweder der Name einer System-Variablen, die den Pfad für die anzumeldende Datei enthält, oder `-STD-` oder ein Minuszeichen angegeben werden. Die Angabe `-STD-` ist gleichbedeutend mit der Angabe der System-Variablen `TUSTEP_DSK`, die den Standard-Träger für permanente Dateien enthält. Wird für das Argument `traeger` ein Minuszeichen angegeben, so muss der mit dem Argument `name` angegebene Dateiname zuvor definiert worden sein (siehe `DEFINE`-Anweisung Seite 423). Das Argument `traeger` kann einschließlich des davor stehenden Kommas weggelassen werden; in diesem Fall wird als Träger `-STD-` angenommen.

Unter Windows kann für das Argument `traeger` auch direkt ein Laufwerksbuchstabe angegeben werden, falls der Pfad keinen Verzeichnisnamen enthält.

Falls die Datei angemeldet werden konnte, ist der Funktionswert »OK«; andernfalls wird als Funktionswert eine entsprechende Fehlermeldung geliefert.

```
Beispiel: $$ SET datei = "testdaten"
          $$ SET status = OPEN (datei, READ, -STD-)
```

Ergebnis: status = "OK" oder Fehlermeldung

Damit beim Anmelden einer Datei festgestellt werden kann, um welchen Dateityp es sich handelt, ist es notwendig, auf den Inhalt der Datei zuzugreifen. Dies ist jedoch nicht möglich, wenn schon ein Programm diese Datei mit schreibendem Zugriff belegt hat. In diesem Fall ist die Datei nach dem Aufruf der Makrofunktion nicht angemeldet; als Funktionswert wird eine entsprechende Fehlermeldung geliefert.

Soll in diesem Fall eine bestimmte Zeit gewartet werden, bis die Datei wieder frei ist, kann dies durch Angabe eines Zeitlimits erreicht werden. Das Zeitlimit gibt an, wieoft maximal versucht werden soll, auf die Datei zuzugreifen. Dabei wird vor jedem erneuten Versuch 1 Sekunde (bei großer Recherauslastung betriebssystembedingt etwas länger) gewartet. Das Zeitlimit muss nach dem Argument `modus` durch einen Schrägstrich getrennt angegeben werden. Falls das Zeitlimit erreicht wird, ist die Datei nach dem Aufruf der Makrofunktion nicht angemeldet; als Funktionswert wird »WAIT« geliefert, um anzuzeigen, dass noch länger gewartet werden müsste, wenn die Datei angemeldet werden soll.

Beispiel: `$$ SET datei = "testdaten"`  
`$$ SET status = OPEN (datei, READ/10, -STD-)`  
 Ergebnis: status = "OK", "WAIT" oder Fehlermeldung

CLOSE (name)

Die Makrofunktion `CLOSE` meldet eine Datei ab.

Für das Argument `name` muss ein in Anführungszeichen eingeschlossener Dateiname oder eine Variable angegeben werden, die den Dateinamen enthält.

Wird eine Scratch-Datei abgemeldet, so wird sie gelöscht. Die Daten in der Scratch-Datei werden zuvor überschrieben, wenn nicht mit dem Kommando `#WISCHEN` (siehe Seite 250) ein anderer Modus eingestellt wurde.

Falls die Datei abgemeldet bzw. gelöscht werden konnte, ist der Funktionswert »OK«; andernfalls wird als Funktionswert eine entsprechende Fehlermeldung geliefert.

Beispiel: `$$ SET datei = "testdaten"`  
`$$ SET status = CLOSE (datei)`  
 Ergebnis: status = "OK" oder Fehlermeldung

CHECK (name, art, typ)

Die Makrofunktion `CHECK` prüft eine Datei.

Für das Argument `name` muss ein in Anführungszeichen eingeschlossener Dateiname oder eine Variable angegeben werden, die den Dateinamen enthält.

Das Argument `art` bestimmt, in welcher Art und Weise die Datei überprüft wird: `WRITE` prüft, ob die Datei zum Schreiben angemeldet ist; `UPDATE` prüft bei TUSTEP-Dateien zusätzlich, ob die Datei abgeschlossen ist; `READ` prüft, ob die Datei

zum Lesen oder Schreiben angemeldet ist, und bei TUSTEP-Dateien zusätzlich, ob sie abgeschlossen ist.

Durch die Angabe `SYSTEM` bzw. `TUSTEP` für das Argument `typ` wird außerdem geprüft, ob die Datei eine Fremd-Datei bzw. eine TUSTEP-Datei ist. Diese Prüfung entfällt, wenn das Argument `typ` einschließlich des davor stehenden Kommas weggelassen wird.

Wenn nur das Argument `name` angegeben ist, wird nur geprüft, ob es einen TUSTEP-konformen Dateinamen enthält.

Falls die Bedingungen erfüllt sind, ist der Funktionswert »OK«; andernfalls wird als Funktionswert eine entsprechende Fehlermeldung geliefert.

Beispiel: `$$ SET datei = "testdaten"`  
`$$ SET status = CHECK (datei, READ, TUSTEP)`  
 Ergebnis: `status = "OK"` oder Fehlermeldung

Damit eine Datei geprüft werden kann, ist es notwendig, auf den Inhalt der Datei zuzugreifen. Dies ist jedoch nicht möglich, wenn schon ein Programm diese Datei mit schreibendem Zugriff belegt hat. In diesem Fall wird als Funktionswert eine entsprechende Fehlermeldung geliefert.

Soll in diesem Fall eine bestimmte Zeit gewartet werden, bis die Datei wieder frei ist, kann dies durch Angabe eines Zeitlimits erreicht werden. Das Zeitlimit gibt an, wieoft maximal versucht werden soll, auf die Datei zuzugreifen. Dabei wird vor jedem erneuten Versuch 1 Sekunde (bei großer Rechnerauslastung betriebssystembedingt etwas länger) gewartet. Das Zeitlimit muss nach dem Argument `art` durch einen Schrägstrich getrennt angegeben werden. Falls das Zeitlimit erreicht wird, liefert die Makrofunktion als Funktionswert »WAIT«, um anzuzeigen, dass noch länger gewartet werden müsste, wenn die Datei geprüft werden soll.

Beispiel: `$$ SET datei = "testdaten"`  
`$$ SET status = CHECK (datei, READ/10)`  
 Ergebnis: `status = "OK", "WAIT"` oder Fehlermeldung

`ERASE (name)`

Die Makrofunktion `ERASE` löscht die Daten in einer Datei.

Für das Argument `name` muss ein in Anführungszeichen eingeschlossener Dateiname oder eine Variable angegeben werden, die den Dateinamen enthält.

Die Daten in der Datei werden dabei überschrieben, wenn nicht mit dem Kommando `#WISCHEN` (siehe Seite 250) ein anderer Modus eingestellt wurde.

Falls die Daten in der Datei gelöscht werden konnten, ist der Funktionswert »OK«; andernfalls wird als Funktionswert eine entsprechende Fehlermeldung geliefert.

Beispiel: `$$ SET datei = "testdaten"`  
`$$ SET status = ERASE (datei)`

Ergebnis: status = "OK" oder Fehlermeldung

Um die Daten löschen zu können, muss auf die Datei schreibend zugegriffen werden. Dies ist jedoch nicht möglich, wenn schon ein Programm diese Datei mit lesendem oder schreibendem Zugriff belegt hat; die Bearbeitung des Makros wird in diesem Fall abgebrochen. Soll erst eine bestimmte Zeit gewartet werden, bis die Datei wieder frei ist, kann dies durch einen vorangehenden Aufruf der Makrofunktion LOCK (siehe Seite 490) erreicht werden. Dabei wird die Datei gleichzeitig für andere Zugriffe gesperrt; sie muss durch einen nachfolgenden Aufruf der Makrofunktion UNLOCK wieder freigegeben werden.

Beim Aufruf der Makrofunktion LOCK kann auch angegeben werden, dass die Daten gelöscht werden sollen; ein zusätzlicher Aufruf der Makrofunktion ERASE erübrigt sich in diesem Fall.

DELETE (name)

Diese Makrofunktion DELETE löscht eine Datei.

Für das Argument name muss ein in Anführungszeichen eingeschlossener Dateiname oder eine Variable angegeben werden, die den Namen der Datei enthält.

Die Daten in der Datei werden zuvor überschrieben, wenn nicht mit dem Kommando #WISCHEN (siehe Seite 250) ein anderer Modus eingestellt wurde.

Falls die Datei gelöscht werden konnte, ist der Funktionswert »OK«; andernfalls wird als Funktionswert eine entsprechende Fehlermeldung geliefert.

Beispiel: \$\$ SET datei = "testdaten"  
 \$\$ SET status = DELETE (datei)

Ergebnis: status = "OK" oder Fehlermeldung

Die Datei kann nicht gelöscht werden, wenn ein Programm diese Datei mit lesendem oder schreibendem Zugriff belegt hat; die Bearbeitung des Makros wird in diesem Fall abgebrochen. Soll erst eine bestimmte Zeit gewartet werden, bis die Datei wieder frei ist, kann dies durch einen vorangehenden Aufruf der Makrofunktion LOCK (siehe Seite 490) erreicht werden. Dabei wird die Datei gleichzeitig für andere Zugriffe gesperrt; ein nachfolgender Aufruf der Makrofunktion UNLOCK entfällt in diesem Fall.

DELETE (segmentname, -, dateiname)

Diese Makrofunktion DELETE löscht ein Segment in einer Segment-Datei.

Für das Argument segmentname muss ein in Anführungszeichen eingeschlossener Segmentname oder eine Variable angegeben werden, die den Namen des Segments enthält; für das Argument dateiname muss ein in Anführungszeichen eingeschlossener Dateiname oder eine Variable angegeben werden, die den Namen der Segment-Datei enthält.

Falls das Segment gelöscht werden konnte, ist der Funktionswert »OK«; andernfalls wird als Funktionswert eine entsprechende Fehlermeldung geliefert.

Beispiel: `$$ SET datei = "makro", segment = "test"`  
`$$ SET status = DELETE (segment, -, datei)`

Ergebnis: `status = "OK"` oder Fehlermeldung

Das Segment kann nicht gelöscht werden, wenn ein Programm diese Segment-Datei mit lesendem oder schreibendem Zugriff belegt hat; die Bearbeitung des Makros wird in diesem Fall abgebrochen. Soll erst eine bestimmte Zeit gewartet werden, bis die Datei wieder frei ist, kann dies durch einen vorangehenden Aufruf der Makrofunktion `LOCK` (siehe Seite 490) erreicht werden. Dabei wird die Segment-Datei gleichzeitig für andere Zugriffe gesperrt; sie muss durch einen nachfolgenden Aufruf der Makrofunktion `UNLOCK` wieder freigegeben werden.

`DELETE (name, traeger)`

Diese Makrofunktion `DELETE` löscht ein Projekt (Verzeichnis). Ein Projekt kann jedoch nur gelöscht werden, wenn es keine Dateien und keine Verzeichnisse (mehr) enthält.

Für das Argument `name` muss ein in Anführungszeichen eingeschlossener Projektname oder eine Variable angegeben werden, die den Projektnamen enthält.

Für das Argument `traeger` muss entweder der Name einer System-Variablen, die den Pfad für das zu löschende Projekt enthält, oder `-STD-` angegeben werden. Die Angabe `-STD-` ist gleichbedeutend mit der Angabe der System-Variablen `TUSTEP_DSK`, die den Standard-Träger für permanente Dateien enthält.

Unter Windows kann für das Argument `traeger` auch direkt ein Laufwerksbuchstabe angegeben werden, falls der Pfad keinen Verzeichnisnamen enthält.

Falls das Projekt gelöscht werden konnte, ist der Funktionswert »OK«; andernfalls wird als Funktionswert eine entsprechende Fehlermeldung geliefert.

Beispiel: `$$ SET projekt = "testdir"`  
`$$ SET status = DELETE (projekt, -STD-)`

Ergebnis: `status = "OK"` oder Fehlermeldung

Ist mit dem Argument `name` kein Projektname, sondern `"-"` (mit Anführungszeichen) angegeben, so wird das letzte Verzeichnis des mit dem Argument `traeger` angegebenen Pfades gelöscht.

`DELETE ("-", traeger1, traeger2)`

Diese Makrofunktion `DELETE` löscht diejenigen Verzeichnisse, die in dem mit dem Argument `traeger1` angegebenen Pfad, aber nicht in dem mit dem Argument `traeger2` angegebenen Pfad aufgeführt sind; der mit `traeger2` angegebene Pfad muss mit dem Anfang des mit `traeger1` angegebenen übereinstimmen, d. h. das Verzeichnis, das mit `traeger2` angegeben ist, muss ein übergeordnetes Verzeichnis von dem Verzeichnis sein, das mit `traeger1` angegeben ist. Ein Verzeichnis kann jedoch nur gelöscht werden, wenn es keine Dateien und keine Verzeichnisse (mehr) enthält.

Für die Argumente `traeger1` und `traeger2` muss jeweils entweder der Name

einer System-Variablen, die einen Pfad enthält, oder `-STD-` angegeben werden. Die Angabe `-STD-` ist gleichbedeutend mit der Angabe der System-Variablen `TUSTEP_DSK`, die den Standard-Träger für permanente Dateien enthält.

Unter Windows kann für das Argument `traeger2` auch direkt ein Laufwerksbuchstabe angegeben werden, falls der Pfad keinen Verzeichnisnamen enthält.

Falls die Verzeichnisse gelöscht werden konnten, ist der Funktionswert »OK«; andernfalls wird als Funktionswert eine entsprechende Fehlermeldung geliefert.

Wenn im folgenden Beispiel die System-Variablen `T1` und `T2` die Pfade »~/a/b/c« bzw. »~/a« enthalten, so werden die Verzeichnisse »~/a/b/c« und »~/a/b« gelöscht.

Beispiel: `SET status = DELETE ("~", T1, T2)`

Ergebnis: `status = "OK"` oder Fehlermeldung

`RENAME (altername, neuename)`

Diese Makrofunktion `RENAME` benennt eine Datei um.

Für die Argumente `altername` und `neuename` muss jeweils ein in Anführungszeichen eingeschlossener Dateiname oder eine Variable angegeben werden, die den alten bzw. neuen Namen der Datei enthält.

Falls mit dem Argument `altername` ein »definierter Dateiname« (siehe Seite 29) angegeben wird, so wird dieser umbenannt und nicht die Datei, die damit bezeichnet wird. Soll die damit bezeichnete Datei umbenannt werden, muss die nachfolgend beschriebene Form der Makrofunktion `RENAME` verwendet werden.

Falls die Datei umbenannt werden konnte, ist der Funktionswert »OK«; andernfalls wird als Funktionswert eine entsprechende Fehlermeldung geliefert.

Beispiel: `$$ SET alt = "testdaten", neu = "textdaten"`

`$$ SET status = RENAME (alt, neu)`

Ergebnis: `status = "OK"` oder Fehlermeldung

`RENAME (altername, neuename, traeger)`

Diese Makrofunktion `RENAME` benennt ein Projekt (Verzeichnis) oder eine Datei um.

Für die Argumente `altername` und `neuename` muss jeweils ein in Anführungszeichen eingeschlossener Projekt- bzw. Dateiname oder eine Variable angegeben werden, die den Projekt- bzw. Dateinamen enthält.

Für das Argument `traeger` muss entweder der Name einer System-Variablen, die den Pfad für das umzubenennende Projekt enthält, `-STD-` oder ein Minuszeichen angegeben werden.

Wenn für das Argument `traeger` der Name einer System-Variablen oder `-STD-` angegeben ist, werden in den Argumenten `altername` und `neuename` Projektnamen erwartet. Die System-Variable muss den Pfad für das umzubenennende Projekt enthalten. Die Angabe `-STD-` ist gleichbedeutend mit der Angabe der

System-Variablen `TUSTEP_DSK`, die den Standard-Träger für permanente Dateien enthält.

Wenn für das Argument `traeger` ein Minuszeichen angegeben ist, werden in den Argumenten `altername` und `neuename` »definierte Dateinamen« (siehe Seite 29) erwartet. Es wird in diesem Fall nicht der »definierte Dateiname«, sondern die Datei, die damit bezeichnet wird, umbenannt. Soll ein »definierter Dateiname« selbst umbenannt werden, muss die vorangehend beschriebene Form der Makrofunktion `RENAME` verwendet werden.

Unter Windows kann für das Argument `traeger` auch direkt ein Laufwerksbuchstabe angegeben werden, falls der Pfad keinen Verzeichnisnamen enthält. In diesem Fall werden in den Argumenten `altername` und `neuename` ebenfalls Projektnamen erwartet.

Falls das Projekt bzw. die Datei umbenannt werden konnte, ist der Funktionswert »OK«; andernfalls wird als Funktionswert eine entsprechende Fehlermeldung geliefert.

```
Beispiel: $$ SET alt = "testdir", neu = "textdir"
          $$ SET status = RENAME (alt, neu, -STD-)
```

```
Ergebnis: status = "OK" oder Fehlermeldung
```

```
RENAME (altername, neuename, -, dateiname)
```

Diese Makrofunktion `RENAME` benennt ein Segment in einer Segment-Datei um.

Für die Argument `altername` und `neuename` muss jeweils ein in Anführungszeichen eingeschlossener Segmentname oder eine Variable angegeben werden, die den alten bzw. neuen Namen des Segments enthält.

Falls das Segment umbenannt werden konnte, ist der Funktionswert »OK«; andernfalls wird als Funktionswert eine entsprechende Fehlermeldung geliefert.

```
Beispiel: $$ SET alt = "testdaten", neu = "textdaten"
          $$ SET segmentdatei = "testdatei"
          $$ SET status = RENAME (alt, neu, -, segmentdatei)
```

```
Ergebnis: status = "OK" oder Fehlermeldung
```

Das Segment kann nicht umbenannt werden, wenn ein Programm diese Segment-Datei mit lesendem oder schreibendem Zugriff belegt hat; die Bearbeitung des Makros wird in diesem Fall abgebrochen. Soll erst eine bestimmte Zeit gewartet werden, bis die Datei wieder frei ist, kann dies durch einen vorangehenden Aufruf der Makrofunktion `LOCK` (siehe Seite 490) erreicht werden. Dabei wird die Segment-Datei gleichzeitig für andere Zugriffe gesperrt; sie muss durch einen nachfolgenden Aufruf der Makrofunktion `UNLOCK` wieder freigegeben werden.

```
REARRANGE (name)
```

Die Makrofunktion `REARRANGE` reorganisiert die Daten einer TUSTEP-Datei, d. h. die beim Edieren entstandenen Lücken werden beseitigt und die Sätze in ihrer logischen Reihenfolge angeordnet (vgl. »Reorganisieren von Dateien« Seite 40).

Außerdem wird der nicht mehr belegte Plattenplatz freigegeben, damit die Datei nur noch so viel Platz belegt, wie für die Daten erforderlich ist.

Für das Argument `name` muss ein in Anführungszeichen eingeschlossener Dateiname oder eine Variable angegeben werden, die den Namen der Datei enthält.

Falls die Daten reorganisiert werden konnten, ist der Funktionswert »OK«; andernfalls wird als Funktionswert eine entsprechende Fehlermeldung geliefert.

```
Beispiel: $$ SET datei = "daten"  
          $$ SET status = REARRANGE (datei)
```

Ergebnis: `status = "OK"` oder Fehlermeldung

Die Daten können nicht reorganisiert werden, wenn ein Programm diese Datei mit lesendem oder schreibendem Zugriff belegt hat; die Bearbeitung des Makros wird in diesem Fall abgebrochen. Soll erst eine bestimmte Zeit gewartet werden, bis die Datei wieder frei ist, kann dies durch einen vorangehenden Aufruf der Makrofunktion `LOCK` (siehe Seite 490) erreicht werden. Dabei wird die Datei gleichzeitig für andere Zugriffe gesperrt; sie muss durch einen nachfolgenden Aufruf der Makrofunktion `UNLOCK` wieder freigegeben werden.

#### `COPY (name1, name2)`

Diese Makrofunktion `COPY` kopiert die Daten aus der mit dem Argument `name1` angegebenen Datei unverändert in die mit dem Argument `name2` angegebene Datei; enthält diese schon Daten, so werden sie überschrieben.

Für die Argumente `name1` und `name2` muss jeweils ein in Anführungszeichen eingeschlossener Dateiname oder eine Variable angegeben werden, die den Namen der Datei enthält.

Falls die Daten kopiert werden konnten, ist der Funktionswert »OK«; andernfalls wird als Funktionswert eine entsprechende Fehlermeldung geliefert.

```
Beispiel: $$ SET quelle = "daten", ziel = "kopie"  
          $$ SET status = COPY (quelle, ziel)
```

Ergebnis: `status = "OK"` oder Fehlermeldung

#### `COPY (name1, name2, seite)`

Diese alternative Makrofunktion `COPY` kopiert die Daten aus der mit dem Argument `name1` angegebenen TUSTEP-Datei in die mit dem Argument `name2` angegebene TUSTEP-Datei; enthält diese schon Daten, so bleiben sie erhalten.

Für die Argumente `name1` und `name2` muss jeweils ein in Anführungszeichen eingeschlossener Dateiname oder eine Variable angegeben werden, die den Namen der Datei enthält.

Für das Argument `seite` muss eine Zahl oder eine Variable, die eine Zahl enthält, angegeben werden.

Falls die Ziel-Datei leer ist oder die erste Satznummer der Quell-Datei größer ist als die letzte Satznummer in der Ziel-Datei, bleiben die Satznummern unverändert;



andernfalls wird auf die Seitennummer jeweils das kleinste Vielfache des Arguments `seite` aufaddiert, das erforderlich ist, um größere Satznummern zu erhalten als in der Ziel-Datei bereits vorhanden sind.

Falls die Daten kopiert werden konnten, ist der Funktionswert »OK«; andernfalls wird als Funktionswert eine entsprechende Fehlermeldung geliefert.

Beispiel: `$$ SET quelle = "daten", ziel = "kopie"`  
`$$ SET status = COPY (quelle, ziel, 1000)`

Ergebnis: `status = "OK"` oder Fehlermeldung

#### COMPARE (name1, name2)

Die Makrofunktion `COMPARE` vergleicht die Daten zweier Dateien; bei TUSTEP-Dateien werden auch die Satznummern verglichen, der Dateititel bleibt unberücksichtigt.

Für die Argumente `name1` und `name2` muss jeweils ein in Anführungszeichen eingeschlossener Dateiname oder eine Variable angegeben werden, die den Namen der Datei enthält.

Falls die Daten verglichen werden konnten, ist der Funktionswert

– »YES«, wenn die Daten und die Satznummern identisch sind,

– »DATA«, wenn nur die Daten identisch sind,

– »NO«, wenn die Daten nicht identisch sind;

andernfalls wird als Funktionswert eine entsprechende Fehlermeldung geliefert.

Beispiel: `$$ SET quelle = "daten", ziel = "kopie"`  
`$$ SET status = COMPARE (quelle, ziel)`

Ergebnis: `status = "YES", "DATA", "NO"`  
 oder Fehlermeldung

#### REPAIR (name)

Die Makrofunktion `REPAIR` repariert nicht abgeschlossene TUSTEP-Dateien (vgl. »Retten von Dateien« Seite 40).

Für das Argument `name` muss ein in Anführungszeichen eingeschlossener Dateiname oder eine Variable angegeben werden, die den Namen der Datei enthält.

Falls die Datei repariert werden konnte, ist der Funktionswert »OK«; andernfalls wird als Funktionswert eine entsprechende Fehlermeldung geliefert.

Beispiel: `$$ SET datei = "daten"`  
`$$ SET status = REPAIR (datei)`

Ergebnis: `status = "OK"` oder Fehlermeldung

Die Daten können nicht gerettet werden, wenn ein Programm diese Datei mit lesendem oder schreibendem Zugriff belegt hat; die Bearbeitung des Makros wird in diesem Fall abgebrochen. Soll erst eine bestimmte Zeit gewartet werden, bis die Datei wieder frei ist, kann dies durch einen vorangehenden Aufruf der Makrofunk-

tion `LOCK` (siehe Seite 490) erreicht werden. Dabei wird die Datei gleichzeitig für andere Zugriffe gesperrt; sie muss durch einen nachfolgenden Aufruf der Makrofunktion `UNLOCK` wieder freigegeben werden.

`PROJECT (name)`

Die Makrofunktion `PROJECT` definiert den Projektnamen, der zur Ergänzung der Dateinamen verwendet werden soll, falls dieser nicht explizit mit dem Dateinamen angegeben wird.

Für das Argument `name` muss ein in Anführungszeichen eingeschlossener Projektname oder eine Variable angegeben werden, die den Projektnamen enthält.

Wird an Stelle des Projektnamens `-STD-` angegeben, so wird der Projektname zur Ergänzung der Dateinamen definiert, der beim Initialisieren der TUSTEP-Sitzung eingestellt wurde; dieser ist durch die System-Variable `TUSTEP_PRJ` vorgegeben.

Beispiel: `$$ SET projekt = "name/xy"`  
`$$ SET status = PROJECT (projekt)`

Ergebnis: `status = "OK"` oder Fehlermeldung

## Makrofunktionen zur Dateisperre

Lesende Zugriffe auf eine Datei sind nicht möglich, wenn schon schreibend auf die Datei zugegriffen wird; schreibende sind nicht möglich, wenn schon lesend oder schreibend auf die Datei zugegriffen wird. Ist ein Zugriff auf eine Datei nicht möglich, wird die Bearbeitung des Makros abgebrochen.

Wenn nicht von mehreren TUSTEP-Sitzungen auf dieselbe Datei zugegriffen wird, ergeben sich daraus in der Regel keine Probleme. Andernfalls muss der Zugriff auf solche Dateien innerhalb von Makros koordiniert werden. Dazu kann die Datei vor dem Zugriff mit der Makrofunktion `LOCK` gesperrt werden und nach dem Zugriff mit der Makrofunktion `UNLOCK` wieder freigegeben werden.

Die Makrofunktion `LOCK` sperrt die Datei nur für fremde Zugriffe. Ist dies nicht möglich, weil auf die Datei gerade zugegriffen wird, kann gewartet werden, bis keine Zugriffe mehr vorliegen bzw. bis ein vorgegebenes Zeitlimit abgelaufen ist.

Falls beim Aufruf von der Makrofunktion `LOCK` oder `UNLOCK` das Fehlerflag zuvor gesetzt wurde (z. B. explizit durch die Makroanweisung `ERROR` oder automatisch bei Auftreten eines Syntaxfehlers) wird die Leistung der Makrofunktion nicht erbracht; der Funktionswert liefert in diesem Fall eine entsprechende Fehlermeldung.

`LOCK (name, modus, limit)`

Die Makrofunktion `LOCK` sperrt eine Datei für fremde Zugriffe, d. h. für alle Zugriffe außerhalb des Makros.

Für das Argument `name` muss ein in Anführungszeichen eingeschlossener Dateiname oder eine Variable angegeben werden, die den Dateinamen enthält.

Sollen auf die Datei anschließend nur Lesezugriffe möglich sein, muss für das

Argument `modus` die Zeichenfolge `READ` angegeben werden. Die Datei wird damit für fremde Schreibzugriffe gesperrt.

Sollen auf die Datei anschließend Lese- und Schreibzugriffe möglich sein, muss für das Argument `modus` die Zeichenfolge `WRITE` oder `ERASE` angegeben werden. Die Datei wird damit für fremde Lese- und Schreibzugriffe gesperrt. Wird für das Argument `modus` das Schlüsselwort `ERASE` angegeben, so werden zusätzlich die Daten in der Datei gelöscht; dabei werden die Daten überschrieben, wenn nicht mit dem Kommando `#WISCHEN` (siehe Seite 250) ein anderer Modus eingestellt wurde.

Mit dem Argument `limit` muss angegeben werden, wieoft maximal versucht werden soll, die Datei zu sperren. Dabei wird vor jedem erneuten Versuch 1 Sekunde (bei großer Rechnerauslastung betriebssystembedingt etwas länger) gewartet.

Falls die Datei gesperrt werden kann, ist der Funktionswert »OK«; falls das Zeitlimit erreicht wird (die Datei also nicht gesperrt werden kann, weil die Datei durch Lese- bzw. Schreibzugriffe noch belegt ist), wird als Funktionswert »WAIT« geliefert, um anzuzeigen, dass noch länger gewartet werden müsste, wenn die Datei gesperrt werden soll; andernfalls wird als Funktionswert eine entsprechende Fehlermeldung geliefert.

Beispiel: `$$ SET datei = "testdaten"`  
`$$ SET status = LOCK (datei, READ, 10)`

Ergebnis: `status = "OK", "WAIT" oder Fehlermeldung`

`UNLOCK (name)`

Die Makrofunktion `UNLOCK` hebt die mit der Makrofunktion `LOCK` gesetzte Sperre für eine Datei wieder auf.

Für das Argument `name` muss ein in Anführungszeichen eingeschlossener Dateiname oder eine Variable angegeben werden, die den Dateinamen enthält.

Falls die Sperre aufgehoben werden kann, ist der Funktionswert »OK«; andernfalls wird als Funktionswert eine entsprechende Fehlermeldung geliefert.

Beispiel: `$$ SET datei = "testdaten"`  
`$$ SET status = UNLOCK (datei)`

Ergebnis: `status = "OK" oder Fehlermeldung`

## Makrofunktionen für Dateiinhalte

Die folgenden Makrofunktionen fragen Informationen über den Inhalt von Dateien ab und liefern das Ergebnis als Funktionswert.

Als Argument `name` muss ein in Anführungszeichen eingeschlossener Dateiname oder eine Variable angegeben werden, die den Dateinamen enthält.

Die Datei muss angemeldet sein; ggf. muss sie zuvor angemeldet werden. Ist die Datei nicht angemeldet oder nicht vom verlangten Typ (Fremd- oder TUSTEP-Datei), so wird als Funktionswert 0 (Null) bzw. eine leere Zeichenfolge geliefert.

Bei diesen Makrofunktionen muss auf den Inhalt einer Datei zugegriffen werden. Dies ist jedoch nicht möglich, wenn schon ein Programm diese Datei mit schreibendem Zugriff belegt hat. Soll erst eine bestimmte Zeit gewartet werden, bis die Datei wieder frei ist, kann dies durch einen vorangehenden Aufruf der Makrofunktion LOCK (siehe Seite 490) erreicht werden. Dabei wird die Datei gleichzeitig für andere Zugriffe gesperrt; sie muss durch einen nachfolgenden Aufruf der Makrofunktion UNLOCK wieder freigegeben werden.

BYTES (name)

Die Makrofunktion BYTES liefert als Funktionswert die Größe einer TUSTEP- oder Fremd-Datei in Anzahl Bytes.

KBYTES (name)

Die Makrofunktion KBYTES liefert als Funktionswert die Größe einer TUSTEP- oder Fremd-Datei in Anzahl Kilobytes.

MBYTES (name)

Die Makrofunktion MBYTES liefert als Funktionswert die Größe einer TUSTEP- oder Fremd-Datei in Anzahl Megabytes.

USED (name)

Die Makrofunktion USED liefert als Funktionswert die Anzahl der mit Daten belegten Bytes einer TUSTEP-Datei.

TYPE (name)

Die Makrofunktion TYPE liefert als Funktionswert den Typ einer Datei (FDF für Fremd-Dateien, SEQ oder RAN für TUSTEP-Dateien, TAPE für Band-Dateien).

RECORDS (name)

Die Makrofunktion RECORDS liefert als Funktionswert die Anzahl der Sätze einer TUSTEP-Datei.

RECLLEN (name)

Die Makrofunktion RECLLEN liefert als Funktionswert die durchschnittliche Länge der Sätze einer TUSTEP-Datei.

MINLEN (name)

Die Makrofunktion MINLEN liefert als Funktionswert die Länge des kürzesten Satzes einer TUSTEP-Datei.

Einschränkung: Wird mit dem Editor der kürzeste Satz einer Datei gelöscht oder verlängert, so wird die Länge des danach kürzesten Satzes nicht neu festgestellt. Die Variable enthält in diesem Fall den alten (falschen) Wert. Ist der korrekte Wert erforderlich, muss die Datei zuvor mit dem Kommando #RETTE (siehe Seite 216) kopiert werden.

MAXLEN (name)

Die Makrofunktion MAXLEN liefert als Funktionswert die Länge des längsten Satzes einer TUSTEP-Datei.

Einschränkung: Wird mit dem Editor der längste Satz einer Datei gelöscht oder verkürzt, so wird die Länge des danach längsten Satzes nicht neu festgestellt. Die Variable enthält in diesem Fall den alten (falschen) Wert. Ist der korrekte Wert erforderlich, muss die Datei zuvor mit dem Kommando #RETTE (siehe Seite 216) kopiert werden.

PAGES (name)

Die Makrofunktion PAGES liefert als Funktionswert die Anzahl der Seiten (genauer: Seitennummer des letzten Satzes – Seitennummer des ersten Satzes + 1) einer TUSTEP-Datei.

FIRST\_PAGE (name)

Die Makrofunktion FIRST\_PAGE liefert als Funktionswert die Seitennummer des ersten Satzes einer TUSTEP-Datei.

LAST\_PAGE (name)

Die Makrofunktion LAST\_PAGE liefert als Funktionswert die Seitennummer des letzten Satzes einer TUSTEP-Datei.

MODIFIED (name)

Die Makrofunktion MODIFIED (mit einem Argument) liefert als Funktionswert Datum und Uhrzeit der letzten Änderung einer TUSTEP- oder einer Fremd-Datei.

Datum und Uhrzeit werden in der Form »yyyy-mm-dd hh:mm:ss« angegeben. Diese Form ermöglicht eine einfache Abfrage, welche von zwei Dateien zuletzt geändert wurde. Das folgende Beispiel zeigt dies:

```

$$ SET zeit1 = MODIFIED (datei1)
$$ SET zeit2 = MODIFIED (datei2)
$$ IF (zeit1.LT.zeit2) THEN
$$   + Daten in <datei2> nach denen in <datei1> geändert
$$ ELSEIF (zeit1.GT.zeit2) THEN
$$   + Daten in <datei1> nach denen in <datei2> geändert
$$ ELSE
$$   + Zeit der Änderung von <datei1> und <datei2> gleich
$$ ENDIF

```

In diesem Beispiel wird davon ausgegangen, dass zuvor mit der MODE-Anweisung (siehe ab Seite 413) der Modus VARIABLE eingestellt wurde.

MODIFIED (dateiname, segmentname)

Diese Makrofunktion MODIFIED (mit zwei Argumenten) liefert als Funktionswert Datum und Uhrzeit der letzten Änderung eines Segments einer Segment-Datei.

Datum und Uhrzeit werden wie bei der vorangehend beschriebenen Makrofunktion in der Form »yyyy-mm-dd hh:mm:ss« angegeben.

TITLE (name)

Die Makrofunktion TITLE liefert als Funktionswert den Dateititel einer TUSTEP-Datei.

TITLE (dateiname, segmentname)

Die Makrofunktion TITLE liefert als Funktionswert den Titel eines Segments in einer Segment-Datei.

EDIT\_INFO (name, datei, segment)

Die Makrofunktion EDIT\_INFO liefert Informationen zum Inhalt einer TUSTEP-Datei.

Falls die Datei Daten enthält, die im Editor mit einer Hole-Anweisung aus einer anderen Datei geholt wurden, wird der Name dieser Datei in die Variable `datei` gespeichert; andernfalls wird eine leere Zeichenfolge in die Variable `datei` gespeichert. Falls ein Segment aus einer anderen Datei geholt wurde, wird zusätzlich der Name dieses Segments in die Variable `segment` gespeichert; andernfalls wird eine leere Zeichenfolge in die Variable `segment` gespeichert.

Der Funktionswert liefert die Zeichenfolge

- »UNKNOWN«, falls die Datei nicht angemeldet oder eine Fremd-Datei ist.
- »MODIFIED«, wenn die Daten mit der Hole-Anweisung geholt, seither geändert, aber noch nicht wieder mit der Rette-Anweisung gerettet wurden.
- »OK« in allen anderen Fällen.

FILE (name, auswahl, anzahl, code)

Die Makrofunktion FILE liefert als Funktionswert Daten aus einer TUSTEP- oder Fremd-Datei.

Mit dem Argument `auswahl` können Teile der Datei ausgewählt werden; es sind folgende Angaben möglich:

- -: Inhalt der Datei (Voreinstellung).
- "segmentname" (mit Anführungszeichen): Inhalt des Segments mit dem angegebenen Namen.
- `variablenname`: Inhalt des Segments, dessen Name in der angegebenen Variablen steht.
- "?" (mit Anführungszeichen): Inhaltsverzeichnis der Segment-Datei.
- `n` (`n` = Zahl): Inhalt der Datei, jedoch von jedem Satz maximal `n` Zeichen.

Mit dem Argument `anzahl` können die mit dem Argument `auswahl` ausgewählten Daten eingeschränkt werden; es sind folgende Angaben möglich:

- 0 (Null): Keine Einschränkung der Daten (Voreinstellung).
- `+n` (`n` = Zahl): Nur die ersten `n` Sätze
- `-n` (`n` = Zahl): Nur die letzten `n` Sätze.
- `+var`: Nur die ersten `n` Sätze, wobei die Variable `var` die Anzahl `n` enthält.

– `-var`: Nur die letzten `n` Sätze, wobei die Variable `var` die Anzahl `n` enthält.

Bei Fremd-Dateien kann mit dem Argument `code` der Code angegeben werden, von dem die Daten in den TUSTEP-Code umcodiert werden sollen; vorgesehen sind `ISO` (= `ISO-8859-1`), `UTF8` und `UTF16`. Voreingestellt ist `ISO`. Wird statt eines Codes ein Minuszeichen oder `BINARY` angegeben, so werden die Daten nicht umcodiert; es werden nur die Codes für Zeilenwechsel (`CR+LF` und `LF`) ausgewertet. Wird `BINARY/NLF` angegeben, bleiben die Daten einschließlich jeglicher Steuerzeichen unverändert.

Das Argument `code` bzw. die beiden Argumente `anzahl` und `code` bzw. die drei Argumente `auswahl`, `anzahl` und `code` können jeweils einschließlich des davor stehenden Kommas weggelassen werden; in diesem Fall gelten die Voreinstellungen.

Der Funktionswert wird als Sternvariable gespeichert, wobei jede Zeile der Sternvariablen einem Satz (Record) der Datei entspricht. Falls jedoch als Argument `code` `BINARY/NLF` angegeben ist, wird der Inhalt der Datei als eine Zeichenfolge gespeichert.

`SEGMENT (dateiname, segmentname)`

Die Makrofunktion `SEGMENT` liefert als Funktionswert die Daten eines Segments einer Segment-Datei.

Der Funktionswert wird als Sternvariable gespeichert, wobei jede Zeile der Sternvariablen einem Satz (Record) des Segments entspricht.

`WRITE (pfad, daten, code, limit)`

Die Makrofunktion `WRITE` löscht den Inhalt der mit dem Argument `pfad` angegebenen Datei und schreibt den Inhalt der Variablen `daten` in diese Datei. Die Datei muss bei dieser Makrofunktion existieren, muss aber nicht angemeldet sein.

Der Funktionswert ist

- »OK«, wenn die Daten in die Datei geschrieben wurden,
- »CREATE«, wenn die Datei nicht existiert,
- »WAIT«, wenn das Zeitlimit (s. u.) abgelaufen ist.

In allen anderen Fällen wird als Funktionswert eine entsprechende Fehlermeldung geliefert.

Das Argument `pfad` muss den vollständigen Pfad samt Dateiname der Datei enthalten. Dieser Pfad kann in der Regel mit der Makrofunktion `FULL_NAME` (siehe Seite 498) abgefragt werden. Statt einer Variablen kann der Pfad auch in Anführungszeichen eingeschlossen angegeben werden.

Der Inhalt der mit dem Argument `daten` angegebenen Variablen wird als ein Satz in die angegebenen Datei geschrieben. Falls die Variable eine Sternvariable ist, wird für jede Zeile der Variablen ein Satz in die Datei geschrieben.

Bei Fremd-Dateien kann mit dem Argument `code` der Code angegeben werden, in den die Daten vom TUSTEP-Code umcodiert werden sollen; vorgesehen sind die Codes `ISO` (= `ISO-8859-1`), `UTF8` und `UTF16`. Nach den Optionen `UTF8` und

UTF16 kann noch die Option `BOM` angegeben werden um die Byte-Order-Mark am Dateianfang zu unterdrücken. Wird statt eines Codes ein Minuszeichen oder `BINARY` angegeben, werden die Daten nicht umcodiert. Wird `BINARY/NLF` angegeben, werden nur die Daten und keine Codes für Zeilenwechsel (CR+LF unter Windows, LF unter Linux und macOS) in die Datei ausgegeben.

Schreibender Zugriff auf eine Datei ist nicht möglich, wenn ein Programm diese Datei gerade mit lesendem oder schreibendem Zugriff belegt hat. Soll in diesem Fall eine bestimmte Zeit gewartet werden, bis die Datei wieder frei ist, kann mit dem Argument `limit` ein Zeitlimit angegeben werden. Das Zeitlimit gibt an, wie oft noch maximal versucht werden soll, auf die Datei zuzugreifen. Dabei wird vor jedem erneuten Versuch 1 Sekunde (bei großer Rechnerauslastung betriebssystembedingt etwas länger) gewartet.

Das Argument `limit` kann einschließlich des davor stehenden Kommas weggelassen werden; in diesem Fall wird als Zeitlimit der Wert 0 angenommen; falls außerdem die Daten nicht umcodiert werden sollen, kann auch das Argument `code` einschließlich des davor stehenden Kommas weggelassen werden.

`READ (pfad, daten, code, limit)`

Die Makrofunktion `READ` liest den gesamten Inhalt der mit dem Argument `pfad` angegebenen Datei. Die Datei muss bei dieser Makrofunktion existieren, muss aber nicht angemeldet sein.

Der Funktionswert ist

- »OK«, wenn die Daten aus der Datei gelesen wurden,
- »CREATE«, wenn die Datei nicht existiert,
- »WAIT«, wenn das Zeitlimit (s. u.) abgelaufen ist.

In allen anderen Fällen wird als Funktionswert eine entsprechende Fehlermeldung geliefert.

Das Argument `pfad` muss den vollständigen Pfad samt Dateiname der Datei enthalten. Dieser Pfad kann in der Regel mit der Makrofunktion `FULL_NAME` (siehe Seite 498) abgefragt werden. Statt einer Variablen kann der Pfad auch in Anführungszeichen eingeschlossen angegeben werden.

Die von der Datei gelesenen Daten werden als Sternvariable in der mit dem Argument `daten` angegebenen Variablen gespeichert, wobei jede Zeile der Sternvariablen einem Satz (Record) der Datei entspricht.

Bei Fremd-Dateien kann mit dem Argument `code` der Code angegeben werden, von dem die Daten in den TUSTEP-Code umcodiert werden sollen; vorgesehen sind `ISO` (= ISO-8859-1), `UTF8` und `UTF16`. Wird statt eines Codes ein Minuszeichen oder `BINARY` angegeben, werden die Daten nicht umcodiert. Wird `BINARY/NLF` angegeben, bleiben die Daten einschließlich jeglicher Steuerzeichen unverändert.

Lesender Zugriff auf eine Datei ist nicht möglich, wenn ein Programm diese Datei gerade mit schreibendem Zugriff belegt hat. Soll in diesem Fall eine bestimmte Zeit gewartet werden, bis die Datei wieder frei ist, kann mit dem Argument `limit` ein Zeitlimit angegeben werden. Das Zeitlimit gibt an, wie oft noch maximal ver-



sucht werden soll, auf die Datei zuzugreifen. Dabei wird vor jedem erneuten Versuch 1 Sekunde (bei großer Rechnerauslastung betriebssystembedingt etwas länger) gewartet.

Das Argument `limit` kann einschließlich des davor stehenden Kommas weggelassen werden; in diesem Fall wird als Zeitlimit der Wert 0 angenommen; falls außerdem die Daten nicht umcodiert werden sollen, kann auch das Argument `code` einschließlich des davor stehenden Kommas weggelassen werden.

## Makrofunktionen für Datei- und Projektnamen

Die folgenden Makrofunktionen fragen Informationen über Dateien und Projekte ab und liefern das Ergebnis als Funktionswert.

Hinweis: Mit der Makrofunktion `RENAME` (siehe Seite 486) können Datei- und Projektnamen geändert werden.

`PROJECT_NAME` (*name*)

Die Makrofunktion `PROJECT_NAME` liefert als Funktionswert den Namen des Projekts einer TUSTEP- oder Fremd-Datei.

Für das Argument *name* muss ein in Anführungszeichen eingeschlossener Dateiname oder eine Variable angegeben werden, die den Dateinamen enthält.

`FILE_NAME` (*name*)

Die Makrofunktion `FILE_NAME` liefert als Funktionswert den Namen einer TUSTEP- oder Fremd-Datei (ohne den Projektnamen).

Für das Argument *name* muss ein in Anführungszeichen eingeschlossener Dateiname oder eine Variable angegeben werden, die den Dateinamen enthält.

`COMPLETE_NAME` (*name*)

Die Makrofunktion `COMPLETE_NAME` liefert als Funktionswert den Projekt- und Dateinamen einer TUSTEP- oder Fremd-Datei.

Für das Argument *name* muss ein in Anführungszeichen eingeschlossener Dateiname oder eine Variable angegeben werden, die den Dateinamen enthält.

`FULL_NAME` (*TUSTEP*, *name*, *traeger*)

Die Makrofunktion `FULL_NAME` liefert als Funktionswert den vollständigen Namen (Pfad) einer TUSTEP- oder Fremd-Datei, unter dem die Datei vom Betriebssystem verwaltet wird.

Für das Argument *name* muss ein in Anführungszeichen eingeschlossener Dateiname oder eine Variable angegeben werden, die den Dateinamen enthält. Der Dateiname muss TUSTEP-Konventionen entsprechen.

Für das Argument *traeger* kann entweder der Name einer System-Variablen, die den Pfad für die Datei enthält, `-STD-` oder ein Minuszeichen angegeben werden.

Die Angabe `-STD-` ist gleichbedeutend mit der Angabe der System-Variablen `TUSTEP_DSK`, die den Standard-Träger für permanente Dateien enthält. Wird für das Argument `traeger` ein Minuszeichen angegeben, so muss zum Argument `name` ein definierter Dateiname (siehe »Definierte Dateinamen« Seite 29) angegeben werden; ist der angegebene Dateiname nicht definiert, liefert die Makrofunktion eine leere Zeichenfolge.

Unter Windows kann für das Argument `traeger` auch direkt ein Laufwerksbuchstabe angegeben werden, falls der Pfad keinen Verzeichnisnamen enthält.

Wenn die zum Argument `name` angegebene Datei angemeldet ist, kann das Argument `traeger` einschließlich des davor stehenden Kommas weggelassen werden; in diesem Fall wird der beim Anmelden angegebene Träger angenommen. Wenn die zum Argument `name` angegebene Datei eine Scratch-Datei ist, darf das Argument `traeger` nicht angegeben werden.

`FULL_NAME (SYSTEM, name)`

Die Makrofunktion `FULL_NAME` liefert als Funktionswert den vollständigen Namen (Pfad) einer TUSTEP- oder Fremd-Datei, unter dem die Datei vom Betriebssystem verwaltet wird.

Für das Argument `name` muss ein in Anführungszeichen eingeschlossener Dateiname oder eine Variable angegeben werden, die den Dateinamen enthält. Der Dateiname muss Betriebssystem-Konventionen entsprechen.

Die Makrofunktion liefert den Dateinamen unverändert, falls er schon einen vollständigen Pfad enthält. Falls der Dateiname mit einer Tilde beginnt, wird diese Tilde durch den Inhalt der System-Variablen `HOME` ersetzt; andernfalls wird das aus Sicht des Betriebssystems aktuelle Verzeichnis (current working directory, z. B. das Verzeichnis, aus dem heraus TUSTEP aufgerufen wurde) hinzugefügt.

Unter Linux und macOS wird die System-Variablen `HOME` vom Betriebssystem definiert und enthält z. B. `/home/userid` bzw. `/Users/userid`, wobei an Stelle von `userid` die jeweilige Benutzerkennung steht. Unter Windows wird die System-Variablen `HOME` beim Start von TUSTEP definiert und enthält den Inhalt der beiden System-Variablen `HOMEDRIVE` und `HOMEPATH`, die vom Betriebssystem definiert werden; sie enthält z. B. `C:\Users\userid`, wobei an Stelle von `userid` die jeweilige Benutzerkennung steht.

`FULL_NAME (EXECUTE, name)`

Die Makrofunktion `FULL_NAME` liefert als Funktionswert den vollständigen Namen (Pfad) eines Programms, unter dem es vom Betriebssystem verwaltet wird.

Für das Argument `name` muss ein in Anführungszeichen eingeschlossener Programmname oder eine Variable angegeben werden, die den Programmnamen enthält.

Unter Windows muss der Programmname mit »`.EXE` oder mit `.BAT` enden; tut er das nicht, wird der angegebene Name intern mit `.EXE` ergänzt.

Das Programm wird in den Verzeichnissen gesucht, die in der System-Variablen

PATH vorgegeben sind. Wird es in keinem dieser Verzeichnisse gefunden, liefert der Funktionswert eine leere Zeichenfolge.

`DIRNAME (pfad)`

Die Makrofunktion `DIRNAME` liefert als Funktionswert den Teil vom Inhalt der Variablen `pfad`, der vor dem letzten Backslash bzw. Schrägstrich steht; ein abschließender Backslash bzw. Schrägstrich bleibt dabei unberücksichtigt. Falls die Variable `pfad` (außer einem abschließenden) keinen Backslash bzw. Schrägstrich enthält, ist der Funktionswert ein Punkt (= Pfad für das aktuelle Verzeichnis).

`BASENAME (pfad)`

Die Makrofunktion `BASENAME` liefert als Funktionswert den Teil vom Inhalt der Variablen `pfad`, der nach dem letzten Backslash bzw. Schrägstrich steht; ein abschließender Backslash bzw. Schrägstrich bleibt dabei unberücksichtigt. Falls die Variable `pfad` (außer einem abschließenden) keinen Backslash bzw. Schrägstrich enthält, ist der Funktionswert der Inhalt der Variablen `pfad`.

`ADJUST_PATH (pfad, name)`

Die Makrofunktion `ADJUST_PATH` fügt den in der Variablen `pfad` enthaltene Pfad und den in der Variablen `name` enthaltenen Dateinamen zusammen und liefert das Ergebnis als Funktionswert. Falls der Pfad nicht mit Backslash bzw. Schrägstrich endet, wird unter Windows ein Backslash, unter Linux und macOS ein Schrägstrich ergänzt. Außerdem werden unter Windows jeder Schrägstrich durch einen Backslash, unter Linux und macOS jeder Backslash durch einen Schrägstrich ersetzt.

An Stelle von Variablen kann auch jeweils eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben werden.

`UNIQUE (name1, name2, name3, ...)`

Die Makrofunktion `UNIQUE` prüft, ob die angegebenen Dateinamen alle verschiedene Dateien bezeichnen. Wenn ja, ist der Funktionswert »OK«, wenn nicht, nennt der Funktionswert den Projekt- und Dateinamen der Datei, die als erste mehrfach angegeben ist.

Für die Argumente `name1`, `name2` usw. muss jeweils ein in Anführungszeichen eingeschlossener Dateiname oder eine Variable angegeben werden, die einen Dateinamen enthält. Die Dateien müssen angemeldet sein; ggf. müssen sie zuvor angemeldet werden.

`VOLUME (name)`

Die Makrofunktion `VOLUME` liefert als Funktionswert den Träger einer TUSTEP- oder Fremd-Datei.

Für das Argument `name` muss ein in Anführungszeichen eingeschlossener Dateiname oder eine Variable angegeben werden, die einen Dateinamen enthält. Die angegebene Datei muss angemeldet sein; ggf. muss sie zuvor angemeldet werden.

Der Träger ist der Name einer System-Variablen, die den Pfad zu der angegebenen Datei enthält; unter Windows kann es auch ein Laufwerksbuchstabe sein, der beim Einrichten bzw. letzten Anmelden dieser Datei zur Spezifikation TRAEGER angegeben wurde.

FILES (modus)

Die Makrofunktion FILES liefert als Funktionswert die Namen von Dateien.

Modi:

- CURRENT: Namen aller angemeldeten Dateien (auch Scratch-Dateien).
- SCRATCH: Namen aller Scratch-Dateien.
- MACRO: Namen der mit dem Kommando #DEFINIERE (siehe Seite 132) definierten Makro-Dateien.

Wird kein Modus, d. h. nur () angegeben, so wird der Modus CURRENT angenommen.

FILE\_NAMES ()

Ohne Argumente liefert die Makrofunktion FILE\_NAMES als Funktionswert die mit dem Kommando #DEFINIERE (siehe Seite 132) oder mit der Makroanweisung DEFINE (siehe Seite 423) definierten Dateinamen.

FILE\_NAMES (projekt, traeger)

Mit Argumenten liefert die Makrofunktion FILE\_NAMES als Funktionswert die Namen aller Dateien, die unter dem angegebenen Projekt auf dem angegebenen Träger eingerichtet sind und einen für TUSTEP zulässigen Namen haben (für Dateien mit unzulässigem Namen siehe letzten Abschnitt).

Für das Argument projekt muss entweder eine Variable, die den Namen des Projekts enthält, oder eine der Zeichenfolgen »-«, »+« und »-STD-« (jeweils ohne Anführungszeichen) angegeben werden. Bei der Angabe »-« liefert der Funktionswert die Namen der Dateien, die auf dem angegebenen Träger keinem Projekt zugeordnet sind, bei »+« die Namen der Dateien im aktuellen Projekt und bei »-STD-« die Namen der Dateien in dem bei der Initialisierung der TUSTEP-Sitzung voreingestellten Projekt.

Für das Argument traeger muss entweder der Name einer System-Variablen, die den Pfad für das als Argument projekt angegebene Projekt (Verzeichnis) enthält, oder -STD- angegeben werden. Die Angabe -STD- ist gleichbedeutend mit der Angabe der System-Variablen TUSTEP\_DSK, die den bei der Initialisierung der TUSTEP-Sitzung voreingestellten Pfad enthält.

Unter Windows kann für das Argument traeger auch direkt ein Laufwerksbuchstabe angegeben werden, falls der Pfad keinen Verzeichnisnamen enthält.

Wenn für das Argument projekt »-« (ohne Anführungszeichen) angegeben wird, kann bei dieser Makrofunktion für das Argument traeger auch ein Verzeichnis (directory) angegeben werden. Diese Angabe muss in Anführungszeichen eingeschlossen werden und den vollständigen Pfad zu diesem Verzeichnis sowie den

Namen des Verzeichnisses enthalten. Die Makrofunktion liefert in diesem Fall die Namen aller in diesem Verzeichnis enthaltenen Dateien, also auch derjenigen, die einen für TUSTEP unzulässigen Namen haben. Da solche Namen auch einen Apostroph enthalten können, werden sie nicht als Teilzeichenfolgen (jeweils durch Apostroph getrennt), sondern in Zeilen (zum Speichern in einer Sternvariablen) geliefert.

`PROJECT_NAMES (traeger)`

Die Makrofunktion `PROJECT_NAMES` liefert als Funktionswert die Namen aller Projekte (Verzeichnisse), die auf dem angegebenen Träger eingerichtet sind und einen für TUSTEP zulässigen Namen haben (für Projekte mit unzulässigem Namen siehe letzten Abschnitt).

Für das Argument `traeger` muss entweder der Name einer System-Variablen, die den Pfad für die abzufragenden Projekte (Verzeichnisse) enthält, oder `-STD-` angegeben werden. Die Angabe `-STD-` ist gleichbedeutend mit der Angabe der System-Variablen `TUSTEP_DSK`, die den Standard-Träger für permanente Dateien enthält.

Unter Windows kann für das Argument `traeger` auch direkt ein Laufwerksbuchstabe angegeben werden, falls der Pfad keinen Verzeichnisnamen enthält.

Bei dieser Makrofunktion kann für das Argument `traeger` auch ein Verzeichnis (directory) angegeben werden. Diese Angabe muss in Anführungszeichen eingeschlossen werden und den vollständigen Pfad zu diesem Verzeichnis sowie den Namen des Verzeichnisses enthalten. Die Makrofunktion liefert in diesem Fall die Namen aller in diesem Verzeichnis enthaltenen Verzeichnisse, also auch derjenigen, die einen für TUSTEP unzulässigen Namen haben. Da solche Namen auch einen Apostroph enthalten können, werden sie nicht als Teilzeichenfolgen (jeweils durch Apostroph getrennt), sondern in Zeilen (zum Speichern in einer Sternvariablen) geliefert.

## Makrofunktionen für Datenträger

Die folgenden Makrofunktionen fragen Informationen über Datenträger ab und liefern das Ergebnis als Funktionswert.

Für das Argument `traeger` muss entweder der Name einer System-Variablen, die einen Pfad auf den abzufragenden Datenträger enthält, oder `-STD-` angegeben werden. Die Angabe `-STD-` ist gleichbedeutend mit der Angabe der System-Variablen `TUSTEP_DSK`, die den bei der Initialisierung der TUSTEP-Sitzung voreingestellten Pfad enthält.

Unter Windows kann für das Argument `traeger` auch direkt ein Laufwerksbuchstabe angegeben werden.

`VOLUMES ()`

Die Makrofunktion `VOLUMES` liefert als Funktionswert den Namen derjenigen System-Variablen, die als Trägerangabe verwendet werden können und deren Inhalte

jeweils den vollständigen Pfad eines Verzeichnisses angeben, auf das zugegriffen werden kann.

Unter Windows liefert die Makrofunktion zusätzlich noch die Laufwerksbuchstaben der betriebsbereiten Laufwerke.

`DISK_NAME (traeger)`

Die Makrofunktion `DISK_NAME` liefert als Funktionswert den Namen des angegebenen Datenträgers (z. B. den auf dem USB-Speicher-Stick stehenden Label).

Einschränkung: Die Makrofunktion `DISK_NAME` liefert z. Z. nur unter Windows den Namen des Datenträgers; unter anderen Betriebssystemen ist der Funktionswert immer eine leere Zeichenfolge.

`FREE_DISK_SPACE (traeger)`

Die Makrofunktion `FREE_DISK_SPACE` liefert als Funktionswert den freien Speicherplatz auf dem angegebenen Datenträger (z. B. einem USB-Speicher-Stick) in Anzahl Megabytes.

Einschränkung: Die Makrofunktion `FREE_DISK_SPACE` liefert z. Z. nur unter Windows die richtige Anzahl Megabytes; unter anderen Betriebssystemen ist der Funktionswert immer 0 (Null).

`USED_DISK_SPACE (traeger)`

Die Makrofunktion `USED_DISK_SPACE` liefert als Funktionswert den belegten Speicherplatz auf dem angegebenen Datenträger (z. B. einem USB-Speicher-Stick) in Anzahl Megabytes.

Einschränkung: Die Makrofunktion `USED_DISK_SPACE` liefert z. Z. nur unter Windows die richtige Anzahl Megabytes; unter anderen Betriebssystemen ist der Funktionswert immer 0 (Null).

`DRIVES (typ)`

Die Makrofunktion `DRIVES` liefert unter Windows als Funktionswert die Laufwerksbuchstaben derjenigen Laufwerke, die betriebsbereit sind und die dem angegebenen Typ entsprechen; unter anderen Betriebssystemen ist der Funktionswert immer eine leere Zeichenfolge. Als Typ sind folgende Angaben möglich: `FIXED`, `REMOVEABLE`, `RAMDISK`, `CDROM`, `REMOTE` und `ALL`.

## Makrofunktionen für Datum und Uhrzeit

`DATE (n) oder DATE_n ()`

`n = 0`: Aktueller Wochentag  
(z. B. »Sonntag«)

`n = 1`: Aktuelles Datum in der Form `xx.xx.xx`  
(z. B. »25.01.16«)

`n = 2`: Aktuelles Datum in der Form `xx. xxx. xxxx`

(z. B. »25. Jan. 2019«)

n = 3: Aktuelles Datum in der Form xx. xxxxxx xxxx

(z. B. »25. Januar 2019«)

n = 4: Aktuelles Datum in der Form xxxx-xx-xx

(z. B. »2019-01-25«)

TIME (n) oder TIME\_n ()

n = 1: Uhrzeit in der Form hh.mm (z. B. »12.00«)

n = 2: Uhrzeit in der Form hh:mm (z. B. »12:00«)

n = 3: Uhrzeit in der Form hh:mm:ss (z. B. »12:00:00«)

n = 4: Uhrzeit in der Form hh:mm:ss.xxx (z. B. »12:00:00.000«)

TIME (0) oder TIME ()

Datum und Uhrzeit in der Form »yyyy-mm-dd hh:mm:ss«.

Diese Form der Zeitangabe ermöglicht eine einfache Abfrage, welche von zwei Zeitangaben einen früheren bzw. späteren Zeitpunkt bezeichnet. Ein Beispiel dafür ist bei der Beschreibung der Makrofunktion MODIFIED (siehe Seite 493) angegeben.

TIME\_INTERVAL (modus, von, bis)

Zeitraum zwischen den beiden Zeitangaben, die in den Variablen zu den Argumenten *von* und *bis* jeweils in der Form »yyyy-mm-dd hh:mm:ss« (ohne Anführungszeichen) enthalten sind.

Für das Argument *modus* sind die Angaben DAYS, HOURS, MINUTES und SECONDS vorgesehen; damit wird die Maßeinheit bestimmt, in der die Zeitraum berechnet wird.

Für die Argumente *von* und *bis* genügen auch verkürzte Zeitangaben; die dem Argument *modus* entsprechende Angabe muss jedoch immer vorhanden sein.

Hinweis: Mit der Makrofunktion DATE kann ein Zeitraum in Jahren, Monaten und Tagen bestimmt werden.

DATE (modus, tag, monat, jahr, nummer)

Diese Makrofunktion DATE liefert als Funktionswert eine Zahl, die den Wochentag des Datums angibt, das nach dem Aufruf der Makrofunktion in den zu den Argumenten *tag*, *monat*, *jahr* angegebenen Variablen steht. Dabei gilt folgende Zuordnung: 1=Montag, 2=Dienstag, 3=Mittwoch, 4=Donnerstag, 5=Freitag, 6=Samstag, 7=Sonntag. Bei unzulässigen Eingabewerten (z. B. 29. Februar eines Nicht-Schaltjahres) ist der Funktionswert 0 (Null).

Diese Makrofunktion liefert nicht nur einen Funktionswert; mit ihr sind auch verschiedene Be- und Umrechnungen von Kalenderdaten möglich. Gesteuert wird diese Makrofunktion über das Argument *modus*. Für die Argumente *tag*, *monat*, *jahr* und *nummer* müssen Namen von Variablen angegeben werden. Ob diese Variablen beim Aufruf der Makrofunktion DATE schon Werte enthalten müssen und ob die Werte dieser Variablen von der Makrofunktion verändert werden, ist von der Angabe zum Argument *modus* abhängig.

Um das Rechnen mit Kalenderdaten zu erleichtern, werden die Tage durchnummeriert. Damit hat jeder Tag eine eindeutige Nummer, die Tagesnummer.

Wenn mit dem Argument `modus` nichts anderes angegeben ist, wird jeweils der Gregorianische Kalender zugrunde gelegt, für ein Datum vor dem 15. Oktober 1582 jedoch der Julianische Kalender.

Modi:

– TODAY: Aktuelles Datum

Eingabe: keine

Ergebnis: Aktuelles Tagesdatum in den Variablen zu den Argumenten `tag`, `monat`, `jahr` und aktuelle Tagesnummer in der Variablen zum Argument `nummer`

– NUMBER: Tagesnummer aus dem Tagesdatum berechnen

Eingabe: Tagesdatum in den Variablen zu den Argumenten `tag`, `monat`, `jahr`

Ergebnis: Tagesnummer in der Variablen zum Argument `nummer`

Mit `NUMBER/GREGORIAN` und `NUMBER/JULIAN` kann explizit festgelegt werden, nach welchem Kalender umgerechnet werden soll. Für ein Datum vor dem 15. Oktober 1582 wird jedoch immer der Julianische Kalender zugrunde gelegt.

– DATE: Tagesdatum aus der Tagesnummer berechnen

Eingabe: Tagesnummer in der Variablen zum Argument `nummer`

Ergebnis: Tagesdatum in den Variablen zu den Argumenten `tag`, `monat`, `jahr`

Mit `DATE/GREGORIAN` und `DATE/JULIAN` kann explizit festgelegt werden, nach welchem Kalender umgerechnet werden soll. Für ein Datum vor dem 15. Oktober 1582 wird jedoch immer der Julianische Kalender zugrunde gelegt.

– EASTER: Osterdatum berechnen

Eingabe: Jahreszahl in der Variablen zum Argument `jahr`

Ergebnis: Osterdatum in den Variablen zu den Argumenten `tag`, `monat` und die entsprechende Tagesnummer in der Variablen zum Argument `nummer`

Mit `EASTER/GREGORIAN` und `EASTER/JULIAN` kann explizit festgelegt werden, nach welchem Kalender das Osterdatum berechnet werden soll. Für ein Jahr vor 1583 wird jedoch immer der Julianische Kalender zugrunde gelegt.

– INTERVAL: Zeitraum zwischen zwei Kalenderdaten berechnen

Eingabe: Tagesdatum des ersten Datums in den Variablen zu den Argumenten `tag`, `monat`, `jahr` und die Tagesnummer des zweiten (späteren) Datums in der Variablen zum Argument `nummer`

Ergebnis: Der zwischen den beiden Daten liegende Zeitraum in Jahren, Monaten und Tagen (in den Variablen zu den Argumenten `jahr`, `monat` und `tag`) sowie in Tagen (in der Variablen zum Argument `nummer`)



Mit `INTERVAL/GREGORIAN` und `INTERVAL/JULIAN` kann explizit festgelegt werden, nach welchem Kalender der Zeitraum berechnet werden soll. Sollen für das Tagesdatum und die Tagesnummer verschiedene Kalender gelten, so kann dies mit den Modi `INTERVAL/GREGORIAN/JULIAN` oder mit `INTERVAL/JULIAN/GREGORIAN` angegeben werden. Für ein Datum vor dem 15. Oktober 1582 wird jedoch immer der Julianische Kalender zugrunde gelegt.

#### Beispiel 1: Berechnen des Pfingstdatums von 2019

```
$$ SET jhr = 2019
$$ SET wtg = DATE (EASTER, tag, mon, jhr, num)
$$ SET num = num + 49
$$ SET wtg = DATE (DATE, tag, mon, jhr, num)
```

Es wird zuerst das Osterdatum für das Jahr 2019 berechnet. Dabei ist es belanglos, welche Werte die Variablen `tag`, `mon` und `num` vor dem Aufruf der Datumsfunktion enthalten. Da als Modus `EASTER` angegeben ist, wird nur das Argument `jahr` (d. h. die Jahreszahl, die in der Variablen `jhr` steht) ausgewertet. Wochentag, Tag und Monat von Ostern werden in den Variablen `wtg`, `tag` und `mon` abgespeichert, werden aber hier im Beispiel nicht weiter verwendet. Die Tagesnummer von Ostern wird in der Variablen `num` abgespeichert.

Da zwischen Ostern und Pfingsten genau 7 Wochen (= 49 Tage) liegen, ergibt sich die Tagesnummer von Pfingsten, indem zur Tagesnummer von Ostern 49 addiert wird. Diese Tagesnummer wird dann in das entsprechende Tagesdatum umgerechnet. Dabei ist es belanglos, welche Werte die Variablen für die Argumente `tag`, `monat` und `jahr` vor dem Aufruf der Datumsfunktion enthalten. Da als Modus `DATE` angegeben ist, wird nur das Argument `nummer` (d. h. die Tagesnummer, die in der Variablen `num` steht) ausgewertet. In der Variablen `wtg` wird der Wochentag (7 für Sonntag), in den Variablen `tag`, `mon` und `jhr` werden Tag, Monat und Jahr des Pfingstdatums abgespeichert.

#### Beispiel 2: Berechnen der Kalenderwoche

Um in Makros die Kalenderwoche zu berechnen, in die der heutige Tag fällt, können folgende Anweisungen verwendet werden:

```
$$ SET eins = 1, vier = 4
$$ SET wtx = DATE (TODAY, tag, mon, jahr, numx)
$$ SET wt0 = DATE (NUMBER, vier, eins, jahr, num0)
$$ SET kw = (numx - (num0 - wt0) + 6) / 7
```

Bis zu drei Tage am Anfang des Jahres gehören eventuell noch zur letzten Kalenderwoche des vorangehenden Jahres. In diesem Fall liefert obige Formel die Kalenderwoche Null. Mit den folgenden Anweisungen kann festgestellt werden, ob die letzte Woche des vorangehenden Jahres die Kalenderwoche 52 oder 53 ist:

```
$$ SET tag = 28, mon = 12, jahr = jahr - 1
$$ SET wtx = DATE (NUMBER, tag, mon, jahr, numx)
$$ SET wt0 = DATE (NUMBER, vier, eins, jahr, num0)
$$ SET kw = (numx - (num0 - wt0) + 6) / 7
```

Mit diesen Anweisungen kann auch die Kalenderwoche eines jeden anderen Tages berechnet werden.

## Makrofunktionen für sonstige Informationen

HOST ()

Name des Rechners (= Inhalt der System-Variablen TUSTEP\_HST).

HOST (ipadr)

Name des Rechners mit der zum Argument (ipadr) angegebenen IP-Adresse. Die IP-Adresse muss die Form `n.n.n.n` (z. B. `134.2.3.39`) haben, wobei `n` jeweils eine Zahl bis 255 ist.

Für `ipadr` kann eine Variable oder eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben werden.

SYSTEM ()

Kürzel für das benutzte Betriebssystem:

LINUX = Linux, MAC = macOS, WIN = Windows

VERSION (modus)

Aktuelle TUSTEP-Version

Modi:

– NAME: Versionsname (z. B. `Version 2019`)

– COMPILED: Erstellungsdatum (z. B. `2019-01-23 12:00`)

PRINTER ()

Name des vom Betriebssystem voreingestellten Druckers.

Einschränkung: Die Makrofunktion `PRINTER` liefert z. Z. nur unter Windows einen Druckernamen; unter anderen Betriebssystemen ist der Funktionswert immer eine leere Zeichenfolge.

VARIABLES (modus)

Namen der Variablen, auf die in TUSTEP zugegriffen werden kann.

Modi:

– TUSTEP: TUSTEP-Variablen

– SYSTEM: System-Variablen

PROJECT ()

Aktuell eingestelltes Projekt.

SESSION ( )

Nummer der TUSTEP-Sitzung.

USER ( )

Benutzeridentifikation (= Inhalt der System-Variablen TUSTEP\_USR).

NAME ( )

Aktuell eingestellter Name.

COLS (modus)

Modi:

- CURRENT: liefert die aktuell eingestellte Spaltenzahl.
- LIMIT: liefert unter Windows die Anzahl der Spalten, die maximal in einem TUSTEP-Fenster auf dem Bildschirm dargestellt werden können.
- PIXEL: liefert unter Windows die Bildschirmbreite in Pixel.

Wird kein Modus, d. h. nur ( ) angegeben, so wird der Modus CURRENT angenommen.

ROWS (modus)

Modi:

- CURRENT: liefert die aktuell eingestellte Zeilenzahl.
- LIMIT: liefert unter Windows die Anzahl der Zeilen, die maximal in einem TUSTEP-Fenster auf dem Bildschirm dargestellt werden können.
- PIXEL: liefert unter Windows die Bildschirmhöhe in Pixel.

Wird kein Modus, d. h. nur ( ) angegeben, so wird der Modus CURRENT angenommen.

SOURCE (modus)

Quellen-Information zum ausgeführten Makro.

Modi:

- MACRO\_FILE: liefert den Namen der Makro-Datei, die das Makro enthält.
- MAIN\_FILE: liefert im Normalfall wie der Modus MACRO\_FILE den Namen der Makro-Datei, die das Makro enthält. Falls das Makro jedoch nicht aus der Makro-Datei gelesen wird, sondern aus einer anderen Datei, in der zuletzt mit dem Editor dieses Makro bearbeitet wurde (vgl. unter »Aufruf« Seite 110), liefert der Modus MAIN\_FILE den Namen dieser Datei.

- `CURRENT_FILE`: liefert den Namen der Datei, aus der die Makroanweisungen gerade gelesen werden. Dies ist die gleiche Datei, deren Name der Modus `MAIN_FILE` liefert, falls die Anweisungen nicht auf Grund einer `INCLUDE`-Anweisung (siehe Seite 452) aus einer anderen Datei gelesen werden.
- `MACRO_SEGMENT`: liefert den Namen des Segments, das das Makro enthält.
- `MAIN_SEGMENT`: liefert den Namen des Segments, das das Makro enthält.
- `CURRENT_SEGMENT`: liefert den Namen des Segments, aus dem die Makroanweisungen gerade gelesen werden. Dies ist das gleiche Segment, dessen Name der Modus `MAIN_SEGMENT` liefert, falls die Anweisungen nicht auf Grund einer `INCLUDE`-Anweisung (siehe Seite 452) aus einem anderen Segment gelesen werden.

## Makrofunktion zum Abschicken einer E-Mail

`SEND_MAIL (mailadr, ccadr, subject, text, files)`

schickt eine E-Mail an die in den Variablen `mailadr` und `ccadr` enthaltenen E-Mail-Adressen. Die Variablen können auch mehrere Adressen durch Apostroph getrennt enthalten. An Stelle der Variablen `ccadr` kann auch ein Minuszeichen angegeben werden. Als »Betreff« (Subject) wird die in der Variablen `subject` enthaltene Zeichenfolge verwendet. Der Text der E-Mail wird in der Sternvariablen `text` erwartet. Falls an die E-Mail noch Dateien (attachments) angehängt werden sollen, muss die Variable `files` die Namen dieser Dateien durch Apostroph getrennt enthalten; andernfalls kann an Stelle der Variablen `files` ein Minuszeichen angegeben werden.

Der Funktionswert liefert die Zeichenfolge »OK« oder eine Fehlermeldung. »OK« bedeutet jedoch nur, dass das Mail-Programm die E-Mail zum Versand angenommen hat.

```
Beispiel: $$ SET an = "tustep@zdv.uni-tuebingen.de"
          $$ SET wg = "Script-Sprache"
          $$ SET text = FILE ("dateiname")
          $$ SET status = SEND_MAIL (an, -, wg, text, -)
```

```
Ergebnis: status = "OK"
```

Einschränkung: Diese Makrofunktion ist nur unter Windows möglich.

## Makrofunktion zum Informationsabruf von WWW-Servern

`REQUEST (url, query)`

ruft eine HTML-Seite von einem WWW-Server ab oder ruft ein CGI-Script auf einem WWW-Server auf. Die URL der HTML-Seite bzw. des CGI-Scripts muss mit dem Argument `url` angegeben werden. Für das Argument `url` kann eine Variable oder eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben wer-

den. Erwartet ein CGI-Script einen Querystring (METHODE=GET), kann für das Argument `query` eine Variable oder eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben werden; erwartet ein CGI-Script Daten von der Standard-Eingabe (METHODE=POST), kann für das Argument `query` eine Sternvariable angegeben werden; andernfalls kann das Argument `query` einschließlich des davor stehenden Kommas weggelassen werden.

Die Antwort des WWW-Servers wird (zum Speichern in einer Sternvariablen) als Funktionswert geliefert. In der Regel muss die Antwort noch von ISO-8859-1 oder UTF-8 nach TUSTEP umcodiert werden.

```
Beispiel: $$ SET adresse = "http://www.server.de/test.html"
          $$ SET daten = REQUEST (adresse)
          $$ SET daten = DECODE (daten, ISO)
```

Ergebnis: daten = Inhalt der Datei test.html

```
$$ SET adresse = "http://www.server.de/opac"
```

```
$$ SET liste = *
NAME=Maier, Josef
ORT=Trier
```

```
$$ SET query = ENCODE (liste, CGI)
```

Ergebnis: query = "NAME=Maier,+Josef&ORT=Trier"

```
$$ SET daten = REQUEST (adresse, query)
$$ SET daten = DECODE (daten, ISO)
```

Ergebnis: daten = Antwort auf die Anfrage

Soll die Antwort des WWW-Servers unverändert in eine Datei geschrieben werden, so kann dafür die FILE-Anweisung (siehe Seite 429) verwendet werden.

```
Beispiel: $$ SET adresse = "http://www.server.de/test.html"
          $$ FILE/ERASE/BINARY "d.html" = REQUEST (adresse)
```

Ergebnis: d.html enthält eine Kopie von test.html

## Makrofunktionen für beliebige Variablen-Inhalte

VALUE (var)

Die Makrofunktion VALUE liefert als Funktionswert den (unveränderten) Inhalt der Variablen `var`; die Variable `var` kann auch eine Sternvariable sein.

```
Beispiel: $$ SET alt = "Beispiel"
          $$ SET neu = VALUE (alt)
```

```
Ergebnis: neu = "Beispiel"
```

Wenn die angegebene Variable `var` mit »@« markiert ist, wird nicht der Inhalt dieser Variable geliefert, sondern der Inhalt der Variablen, deren Namen in der angegebenen Variablen `var` steht.

```
Beispiel: $$ SET alt = "Beispiel", name = "alt"
          $$ SET neu = VALUE (@name)
          Ergebnis: neu = "Beispiel"
```

LENGTH (`var`)

Die Makrofunktion `LENGTH` liefert als Funktionswert die Anzahl der Zeichen, die die Variable `var` enthält; die Variable `var` kann auch eine Sternvariable sein.

```
Beispiel: $$ SET text = "%/et%/e"
          $$ SET anzahl = LENGTH (text)
          Ergebnis: anzahl = 7
```

CAPS (`var`)

Die Makrofunktion `CAPS` nimmt den Inhalt der Variablen `var`, tauscht alle Kleinbuchstaben in die entsprechenden Großbuchstaben aus und liefert das Ergebnis als Funktionswert; die Variable `var` kann auch eine Sternvariable sein.

```
Beispiel: $$ SET alt = "Beispiel"
          $$ SET neu = CAPS (alt)
          Ergebnis: neu = "BEISPIEL"
```

SQUEEZE (`var`)

Die Makrofunktion `SQUEEZE` nimmt den Inhalt der Variablen `var`, entfernt führende und abschließende Leerzeichen, reduziert unmittelbar aufeinander folgende Leerzeichen auf eines und liefert das Ergebnis als Funktionswert; falls die Variable `var` eine Sternvariable ist, erfolgt das Entfernen und Reduzieren der Leerzeichen zeilenweise.

```
Beispiel: $$ SET alt = " Dies ist ein Beispiel "
          $$ SET neu = SQUEEZE (alt)
          Ergebnis: neu = "Dies ist ein Beispiel"
```

SHORTEN (`var`, `anz1`, `ersatz`, `anz2`)

Diese Makrofunktion `SHORTEN` fügt die ersten `anz1` Zeichen der Variablen `var`, die Zeichenfolge in der Variablen `ersatz` und die letzten `anz2` Zeichen der Variablen `var` zusammen und liefert das Ergebnis als Funktionswert.

Beginnt die Zeichenfolge in der Variablen `ersatz` mit einem Leerzeichen, so wird in der Variablen `var` von der Position `anz1` aus das nächststehende Leerzeichen gesucht; die Zeichen bis zu diesem Leerzeichen werden übernommen. Entspre-

chend werden die letzten Zeichen bis zu einem Leerzeichen übernommen, wenn die Zeichenfolge in der Variablen `ersatz` mit einem Leerzeichen endet.

Enthält die Variable `var` nicht mehr als `anz1` Zeichen plus die Anzahl der Zeichen in der Variablen `ersatz` plus `anz2` Zeichen, so liefert die Makrofunktion den unveränderten Inhalt der Variablen `var`.

An Stelle der Variablen `ersatz` kann auch eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben werden.

```
Beispiel: $$ SET lang = "Dies ist ein Beispiel"
          $$ SET kurz = SHORTEN (lang, 6, "...", 6)
```

```
Ergebnis: kurz = "Dies i...ispiel"
```

```
          $$ SET kurz = SHORTEN (lang, 6, "...", 6)
```

```
Ergebnis: kurz = "Dies ... Beispiel"
```

`SHORTEN (var, anz1, ersatz, anz2, trenner)`

Diese alternative Makrofunktion `SHORTEN` fügt die ersten `anz1` Wörter der Variablen `var`, die Zeichenfolge in der Variablen `ersatz` und die letzten `anz2` Wörter der Variablen `var` zusammen und liefert das Ergebnis als Funktionswert.

Enthält die Variable `var` nicht mehr als `anz1` plus `anz2` Wörter, so liefert die Makrofunktion den unveränderten Inhalt der Variablen `var`.

An Stelle der Variablen `ersatz` kann auch eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben werden.

Als Trennzeichen zwischen den Wörtern gelten alle Zeichenfolgen, die in der Such-Tabelle `trenner` als Suchzeichenfolgen enthalten sind. Für das Argument `trenner` kann der Name einer mit der `BUILD`-Anweisung definierten Such-Tabelle oder direkt eine in Anführungszeichen eingeschlossene Such-Tabelle angegeben werden.

```
Beispiel: $$ SET lang = "Dies ist ein sehr kurzes Beispiel"
          $$ SET kurz = SHORTEN (lang, 2, "...", 2, ": :")
```

```
Ergebnis: kurz = "Dies ist ... kurzes Beispiel"
```

`CENTER (var, anzahl, erg)`

Die Makrofunktion `CENTER` nimmt den Inhalt der Variablen `var`, ergänzt so oft das mit dem Argument `erg` angegebene Zeichen, dass sich insgesamt `anzahl` Zeichen ergeben und liefert das Ergebnis als Funktionswert.

Für das Argument `anzahl` muss eine Zahl oder eine Variable, die eine Zahl (ohne Vorzeichen) enthält, angegeben werden. Vor der Zahl bzw. vor der Variablen kann ein Vorzeichen angegeben werden. Der Funktionswert liefert die Zeichenfolge aus der Variablen `var` linksbündig bei negativem Vorzeichen, rechtsbündig bei positivem Vorzeichen und auf die Mitte zentriert, wenn kein Vorzeichen angegeben ist.

Falls die Zeichenfolge in der Variablen `var` schon aus `anzahl` oder mehr Zeichen besteht, werden keine Zeichen ergänzt, und der Funktionswert liefert diese Zeichenfolge unverändert.

Für das Argument `erg` kann ein in Anführungszeichen eingeschlossenes Zeichen oder eine Variable, die genau ein Zeichen enthält, angegeben werden. Falls Leerzeichen ergänzt werden sollen, kann das Argument `erg` einschließlich des davor stehenden Kommas weggelassen werden.

An Stelle der Variablen `var` kann auch eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben werden.

```
Beispiel: $$ SET alt = "Beispiel"
          $$ SET neu = CENTER (alt, 10)

Ergebnis: neu = " Beispiel "
```

```
          $$ SET alt = 123
          $$ SET neu = CENTER (alt, +6, "0")

Ergebnis: neu = "000123"
```

COMPLETE (`liste`, `abk`)

Die Makrofunktion `COMPLETE` liefert als Funktionswert die Vollform der in der Variablen `abk` angegebenen Abkürzung.

Die möglichen Vollformen müssen in der Variablen `liste` enthalten und durch Apostroph getrennt sein.

```
Beispiel: $$ SET farben = "blau'gelb'grün'rot", test="ge"
          $$ SET farbe = COMPLETE (farben, test)

Ergebnis: farbe = "gelb"
```

Beginnt keine Vollform mit der Abkürzung, so ist der Funktionswert eine leere Zeichenfolge; beginnen mehrere Vollformen mit der Abkürzung, so enthält der Funktionswert alle betroffenen Vollformen.

```
Beispiel: $$ SET farben = "blau'gelb'grün'rot", test = "g"
          $$ SET farbe = COMPLETE (farben, test)

Ergebnis: farbe = "gelb'grün"
```

Stimmt eine Vollform mit der Abkürzung überein, so enthält der Funktionswert diese Vollform, unabhängig davon, ob noch andere Vollformen mit dieser Abkürzung beginnen.

```
Beispiel: $$ SET farben = "gelbgrün'gelb", test = "gelb"
          $$ SET farbe = COMPLETE (farben, test)

Ergebnis: farbe = "gelb"
```

Bei der Prüfung auf Übereinstimmung werden Groß- und Kleinbuchstaben nicht unterschieden.



REPEAT (var, anz)

Die Makrofunktion REPEAT fügt die in der Variablen `var` enthaltene Zeichenfolge `anz` mal zusammen und liefert das Ergebnis als Funktionswert.

An Stelle der Variablen `var` kann auch eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben werden.

Für die Zeichenzahl `anz` muss eine Zahl oder eine Variable, die eine Zahl enthält, angegeben werden.

Beispiel: `$$ SET line = REPEAT ("-", 5)`

Ergebnis: `line = "-----"`

`$$ SET line = REPEAT ("-", 0)`

Ergebnis: `line = ""`

QUOTES (var)

Die Makrofunktion QUOTES ergänzt am Anfang und am Ende der in der Variablen `var` enthaltenen Zeichenfolge ein doppeltes Anführungszeichen und liefert das Ergebnis als Funktionswert.

An Stelle der Variablen `var` kann auch eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben werden.

Beispiel: `$$ SET hilf = QUOTES ("Esc")`

`$$ SET text = CONCAT ("Taste ", hilf, " drücken.")`

Ergebnis: `text = "Taste "Esc" drücken."`

APPEND (var1, erg, var2)

Diese Makrofunktion APPEND (eine weitere ist auf Seite 543 beschrieben) fügt die in den Variablen `var1`, `erg` und `var2` enthaltenen Zeichenfolgen zusammen und liefert das Ergebnis als Funktionswert.

Enthält die Variable `var1` eine leere Zeichenfolge, so enthält der Funktionswert nur den Inhalt der Variablen `var2`; enthält die Variable `var2` eine leere Zeichenfolge, so enthält der Funktionswert nur den Inhalt der Variablen `var1`.

An Stelle der Variablen `erg` kann auch eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben werden.

Beispiel: `$$ SET alle = "", erg = "blau"`

`$$ SET alle = APPEND (alle, ", ", ", ", erg)`

Ergebnis: `alle = "blau"`

`$$ SET erg = "gelb"`

`$$ SET alle = APPEND (alle, ", ", ", ", erg)`

Ergebnis: alle = "blau, gelb"

Hinweis: Mit der anderen Makrofunktion APPEND (siehe Seite 543) können auch die Inhalte von Sternvariablen zusammengefügt werden.

CONCAT (var1, var2, var3, ...)

Die Makrofunktion CONCAT fügt die in den Variablen var1, var2, var3 usw. enthaltenen Zeichenfolgen zusammen und liefert das Ergebnis als Funktionswert.

An Stelle einer Variablen kann auch jeweils eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben werden.

Beispiel: \$\$ SET teil1 = "schön", teil2 = "gut"  
 \$\$ SET text = CONCAT (teil1, " und ", teil2)

Ergebnis: text = "schön und gut"

Falls Sternvariablen angegeben sind, fügt diese Makrofunktion die Daten zeilenweise zusammen und liefert das Ergebnis (zum Speichern in einer Sternvariablen) als Funktionswert.

Beispiel: \$\$ SET nliste = \*  
 Josef Maier  
 Peter Müller  
 Paula Schmid

\$\$ SET oliste = \*  
 Trier  
 München  
 Tübingen

\$\$ SET liste = CONCAT (nliste, " wohnt in ", oliste)

Ergebnis: liste = \*  
 Josef Maier wohnt in Trier  
 Peter Müller wohnt in München  
 Paula Schmid wohnt in Tübingen

JOIN (var, erg)

Diese Makrofunktion JOIN fügt die in der Sternvariablen var enthaltenen Zeilen zu einer Zeichenfolge zusammen. Dabei wird zwischen den Daten der einzelnen Zeilen jeweils eine Zeichenfolge ergänzt.

Ist nur das Argument var angegeben, wird ein Apostroph zwischen den Daten der einzelnen Zeilen ergänzt.

Beispiel: \$\$ SET stern = \*  
 blau  
 gelb  
 rot

\$\$ SET var = JOIN (stern)

Ergebnis: var = "blau'gelb'rot"

Andernfalls wird die mit dem Argument `erg` angegebene Zeichenfolge ergänzt. Für das Argument `erg` kann eine Variable oder eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben werden.

Beispiel: \$\$ SET stern = \*  
 blau  
 gelb  
 rot

\$\$ SET var = JOIN (stern, "; ")

Ergebnis: var = "blau; gelb; rot"

JOIN (var1, erg, var2, var3, ...)

Diese alternative Makrofunktion `JOIN` fügt die Zeichenfolgen, die die Variablen `var1`, `var2`, `var3` usw. enthalten, zusammen und liefert das Ergebnis als Funktionswert.

Zwischen den Zeichenfolgen wird jeweils die mit dem Argument `erg` angegebene Zeichenfolge ergänzt. Für das Argument `erg` kann eine Variable oder eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben werden.

Beispiel: \$\$ SET daten = "eins/zwei/drei"  
 \$\$ SET anz = SPLIT (daten, ":", zf1, zf2, zf3)

Ergebnis: anz = 3  
 zf1 = "eins", zf2 = "zwei", zf3 = "drei"

\$\$ SET all = JOIN (zf1, "/", zf2, zf3)

Ergebnis: all = "eins/zwei/drei"

Falls die Variablen `var1`, `var2`, `var3` usw. alle Sternvariablen sind, werden die einzelnen Zeilen dieser Sternvariablen zusammengefügt und das Ergebnis (zum Speichern in einer Sternvariablen) als Funktionswert geliefert.

Beispiel: \$\$ SET daten = \*  
 eins/uno/one  
 zwei/duo/two  
 drei/tre

```
$$ SET anz = SPLIT (daten, ":", sp1, sp2, sp3)
```

```
Ergebnis:  sp1 = *      sp2 = *      sp3 = *
             eins      uno      one
             zwei      due      two
             drei      tre
```

```
$$ SET all = JOIN (sp1, "/", sp2, sp3)
```

```
Ergebnis:  all = *
            eins/uno/one
            zwei/duetwo
            drei/tre
```

SPLIT (var, trenner, zeilenlänge)

Diese Makrofunktion SPLIT unterteilt die in der Variablen var enthaltene Zeichenfolge in einzelne Zeilen und liefert das Ergebnis (zum Speichern in einer Sternvariablen) als Funktionswert; falls die Variable var eine Sternvariable ist, erfolgt das Unterteilen zeilenweise.

Ist nur das Argument var angegeben, wird die Zeichenfolge an jedem Apostroph unterteilt.

```
Beispiel: $$ SET var = "blau'gelb'rot"
          $$ SET stern = SPLIT (var)
```

```
Ergebnis:  stern = *
            blau
            gelb
            rot
```

Andernfalls erfolgt die Unterteilung an allen Zeichenfolgen, die in der Such-Tabelle trenner als Suchzeichenfolgen enthalten sind. Für das Argument trenner kann der Name einer mit der BUILD-Anweisung definierten Such-Tabelle oder direkt eine in Anführungszeichen eingeschlossene Such-Tabelle angegeben werden.

Soll keine weitere Unterteilung auf Grund der Zeilenlänge erfolgen, kann für das Argument zeilenlänge eine Null angegeben werden, oder das Argument zeilenlänge kann einschließlich des davor stehenden Kommas weggelassen werden.

```
Beispiel: $$ SET var = "blau; gelb; rot"
          $$ SET stern = SPLIT (var, "/; /")
```

```
Ergebnis:  stern = *
            blau
            gelb
            rot
```

Die unterteilenden Zeichenfolgen werden nicht in das Ergebnis übernommen. Sollen sie mit übernommen werden, muss entweder unmittelbar vor oder unmit-

telbar nach dem Argument `trenner` ein senkrechter Strich angegeben werden. Die unterteilende Zeichenfolge wird dann in die Zeile mit dem nachfolgenden Text bzw. in die Zeile mit dem vorangehenden Text übernommen.

```
Beispiel: $$ SET var = "@n Name @o Ort @l Land"
          $$ SET stern = SPLIT (var, |"/@/")
```

```
Ergebnis: stern = *
           @n Name
           @o Ort
           @l Land
```

Ist an Stelle der Such-Tabelle ein Minuszeichen angegeben, so ist die Angabe einer Zeilenlänge obligat und es wird nur auf Grund der angegebenen Zeilenlänge unterteilt.

Für das Argument `zeilenlänge` kann eine Zahl oder eine Variable angegeben werden; wird eine Variable angegeben, so muss diese mit einem Nummernzeichen unmittelbar vor dem Namen gekennzeichnet sein und eine Zahl enthalten. Ist die Zeilenlänge (`n`) nicht Null, so wird die in der Variablen `var` enthaltene Zeichenfolge (ggf. zusätzlich) so an Leerzeichen in einzelne Zeilen unterteilt, dass diese möglichst `n` Zeichen lang sind. Ist das `n+1`-te Zeichen kein Leerzeichen, wird am letzten Leerzeichen innerhalb der ersten `n` Zeichen unterteilt; ist in diesen `n` Zeichen kein Leerzeichen enthalten, wird am ersten danach vorkommenden Leerzeichen unterteilt.

```
Beispiel: $$ SET var = "Ein kurzer Text als Beispiel."
          $$ SET lng = 12
          $$ SET stern = SPLIT (var, -, #lng)
```

```
Ergebnis: stern = *
           Ein kurzer
           Text als
           Beispiel.
```

Soll bei der Unterteilung der Zeichenfolge auf Grund des Arguments `zeilenlaenge` nicht innerhalb von Tags (genauer zwischen `><<` und `>>>`) eine neue Zeile begonnen werden, so kann nach dem Argument, durch ein Komma getrennt, `>>>` angegeben werden.

```
SPLIT (var, trenner, var1, var2, var3, ...)
```

Diese alternative Makrofunktion `SPLIT` unterteilt den Inhalt der Variablen `var` in einzelne Zeichenfolgen und speichert die erste Zeichenfolge in die Variable `var1`, die zweite in `var2` usw. Der Funktionswert liefert die Anzahl der Zeichenfolgen.

Die Unterteilung erfolgt an allen Zeichenfolgen, die in der Such-Tabelle `trenner` als Suchzeichenfolgen enthalten sind. Für das Argument `trenner` kann der Name einer mit der `BUILD`-Anweisung definierten Such-Tabelle oder direkt eine in Anführungszeichen eingeschlossene Such-Tabelle angegeben werden.

Ergeben sich beim Unterteilen weniger Zeichenfolgen als Variablen angegeben

sind, so wird den restlichen Variablen eine leere Zeichenfolge zugewiesen; ergeben sich mehr Zeichenfolgen als Variablen angegeben sind, so werden die überzähligen Zeichenfolgen zusätzlich der letzten angegebenen Variablen zugewiesen. Die unterteilenden Zeichenfolgen vor den überzähligen Zeichenfolgen bleiben in jedem Fall erhalten, die anderen unterteilenden Zeichenfolgen werden eliminiert, falls nichts anders bestimmt wird.

```
Beispiel: $$ SET var = "blau; gelb; rot; grün"
          $$ SET anz = SPLIT (var, "/; /", v1, v2, v3)
```

```
Ergebnis:  anz = 4
             v1 = "blau"
             v2 = "gelb"
             v3 = "rot; grün"
```

Sollen alle unterteilenden Zeichenfolgen erhalten bleiben, so muss entweder unmittelbar vor oder unmittelbar nach dem Argument `trenner` ein senkrechter Strich angegeben werden. Die unterteilenden Zeichenfolgen werden dann jeweils der nachfolgenden bzw. der vorangehenden Zeichenfolge zugeordnet.

```
Beispiel: $$ SET var = "@n Name @o Ort @l Land"
          $$ SET anz = SPLIT (var, "|"/@"", v1, v2, v3, v4)
```

```
Ergebnis:  anz = 3
             v1 = "@n Name"
             v2 = "@o Ort"
             v3 = "@l Land"
             v4 = ""
```

Falls die Variable `var` eine Sternvariable ist, erfolgt das Unterteilen für jede Zeile einzeln; die sich für die einzelnen Variablen ergebenden Werte werden jeweils (zum Speichern als Sternvariable) zusammengefasst. Der Funktionswert nennt in diesem Fall die maximale Anzahl der Zeichenfolgen, die eine Zeile der Variablen `var` enthält.

```
Beispiel: $$ SET daten = *
           eins/uno/one
           zwei/duo/two
           drei/tre

           $$ SET anz = SPLIT (daten, ":", sp1, sp2, sp3)
```

```
Ergebnis:  anz = 3
             sp1 = *      sp2 = *      sp3 = *
             eins      uno      one
             zwei      duo      two
             drei      tre
```

FORMAT (var, n, erg1, erg2)

Die Makrofunktion `FORMAT` unterteilt die in der Variablen `var` enthaltene Zei-

chenfolge in einzelne Zeilen und liefert das Ergebnis (zum Speichern in einer Sternvariablen) als Funktionswert. Falls die Variable `var` eine Sternvariable ist, erfolgt das Unterteilen zeilenweise.

Für das Argument `n` muss eine Zahl oder eine Variable, die eine Zahl enthält, angegeben werden. Die in der Variablen `var` enthaltene Zeichenfolge wird so an Leerzeichen in einzelne Zeilen unterteilt, dass diese möglichst `n` Zeichen lang sind. Ist das `n+1`-te Zeichen kein Leerzeichen, wird am letzten Leerzeichen innerhalb der ersten `n` Zeichen unterteilt; ist in diesen `n` Zeichen kein Leerzeichen enthalten, wird am ersten danach vorkommenden Leerzeichen unterteilt.

Die in der Variablen `erg1` enthaltene Zeichenfolge wird am Anfang der ersten Zeile, die in der Variablen `erg2` enthaltene Zeichenfolge am Anfang aller restlichen Zeilen ergänzt. Diese Zeichenfolgen werden für die Zeilenlänge mitgezählt. An Stelle der Variablen `erg1` und `erg2` können auch in Anführungszeichen eingeschlossene Zeichenfolgen angegeben werden.

```
Beispiel: $$ SET t = "Dieser Titel dient als Beispiel."
          $$ SET stern = FORMAT (t, 15, "Titel ", "      ")
```

```
Ergebnis: stern = *
           Titel Dieser Titel
           dient als
           Beispiel.
```

`EXTRACT (var, apos, epos)`

Diese Makrofunktion `EXTRACT` wählt aus dem Inhalt der Variablen `var` die Zeichen von Position `apos` (einschließlich) bis zur Position `epos` (ausschließlich) aus und liefert sie als Funktionswert; falls die Variable `var` eine Sternvariable ist, erfolgt das Auswählen zeilenweise.

Für die Argumente `apos` und `epos` kann jeweils eine Zahl, eine Variable, die eine Zahl (ohne Vorzeichen) enthält, oder eine Such-Tabelle angegeben werden. Vor einer Zahl kann ein Vorzeichen angegeben werden; vor einer Variablen muss ein Vorzeichen angegeben werden, damit der Name der Variablen nicht als Name einer Such-Tabelle interpretiert wird. Als Such-Tabelle kann entweder der Name einer mit der `BUILD`-Anweisung definierten Such-Tabelle oder direkt eine in Anführungszeichen eingeschlossene Such-Tabelle angegeben werden.

Wird eine Zahl oder Variable angegeben, so werden die Zeichenpositionen bei positiven Werten von links nach rechts gezählt, bei negativen von rechts nach links. Soll der Inhalt der Variablen vom Anfang an bzw. bis zum Ende übernommen werden, kann `0` (Null) als Position angegeben werden. Enthält die Variable `var` weniger Zeichen, als für die Auswahl bis zur Endposition erforderlich sind, so wird so verfahren, als enthielte die Zeichenfolge bis zur Endposition Leerzeichen.

```

Beispiel: $$ SET alt = "123456789"
          $$ SET neu = EXTRACT (alt, 4, 8)

Ergebnis: neu = "4567"

          $$ SET neu = EXTRACT (alt, -4, -2)

Ergebnis: neu = "67"

          $$ SET neu = EXTRACT (alt, 8, 12)

Ergebnis: neu = "89 "

```

Wird für die Anfangsposition eine Such-Tabelle angegeben, so wird in der Variablen `var` von Anfang an nach einer Zeichenfolge gesucht, die mit einer in der Such-Tabelle angegebenen Suchzeichenfolge übereinstimmt. Falls für die Endposition eine Zahl angegeben ist, endet die Suche spätestens vor der entsprechenden Position; falls ebenfalls eine Such-Tabelle angegeben ist, wird ggf. bis zum Ende des Variablen-Inhalts gesucht. Als Anfangsposition gilt die Position des ersten Zeichens der gefundenen Zeichenfolge. Falls keine entsprechende Zeichenfolge gefunden wird, ist das Ergebnis der Makrofunktion eine leere Zeichenfolge.

```

Beispiel: $$ SET alt = "Tübingen/Neckar"
          $$ SET neu = EXTRACT (alt, "://" , 0)

Ergebnis: neu = "/Neckar"

```

Durch die Angabe eines senkrechten Striches entweder unmittelbar vor oder unmittelbar nach der Such-Tabelle `apos` kann explizit bestimmt werden, ob die auszuwählende Zeichenfolge mit dem ersten Zeichen oder erst nach dem letzten Zeichen der gefundenen Zeichenfolge beginnt.

```

Beispiel: $$ SET alt = "Tübingen/Neckar"
          $$ SET neu = EXTRACT (alt, "://"|, 0)

Ergebnis: neu = "Neckar"

```

Wird für die Endposition eine Such-Tabelle angegeben, so wird nach einer Zeichenfolge gesucht, die mit einer in der Such-Tabelle angegebenen Suchzeichenfolge übereinstimmt. Die Suche nach einer solchen Zeichenfolge beginnt mit der angegebenen Anfangsposition bzw. mit der auf das letzte Zeichen der für die Anfangsposition gefundenen Zeichenfolge folgenden Position. Als Endposition gilt die Position des ersten Zeichens der gefundenen Zeichenfolge. Falls keine entsprechende Zeichenfolge gefunden wird, wird der Inhalt der Variablen bis zum Ende übernommen.

```

Beispiel: $$ SET alt = "Tübingen/Neckar"
          $$ SET neu = EXTRACT (alt, 0, "://:")

Ergebnis: neu = "Tübingen"

```

Durch die Angabe eines senkrechten Striches entweder unmittelbar vor oder unmittelbar nach der Such-Tabelle `epos` kann explizit bestimmt werden, ob die aus-



zuwählende Zeichenfolge vor dem ersten Zeichen oder erst nach dem letzten Zeichen der gefundenen Zeichenfolge endet.

```
Beispiel: $$ SET alt = "72074 Tübingen/Neckar"
          $$ SET neu = EXTRACT (alt, " ::"|, |":/:" )
          Ergebnis: neu = "Tübingen"
```

Zur Auswahl eines eingeklammerten Textes kann die öffnende Klammer in der Such-Tabelle für die Anfangsposition und die schließende Klammer in der Such-Tabelle für die Endposition definiert werden.

```
Beispiel: $$ SET alt = "...(eins)..."
          $$ SET neu = EXTRACT (alt, " (::"|, |":):" )
          Ergebnis: neu = "eins"
```

Sind die Klammern jedoch geschachtelt, liefert die Makrofunktion den gewünschten Text vermutlich unvollständig, weil innere und äußere Klammern nicht unterschieden werden:

```
Beispiel: $$ SET alt = "...(eins(zwei)drei)..."
          $$ SET neu = EXTRACT (alt, " (::"|, |":):" )
          Ergebnis: neu = "eins(zwei)"
```

Werden die Klammern zuvor mit Markierungen versehen, können innere und äußere Klammern unterschieden werden. Diese Markierungen können anschließend wieder entfernt werden.

```
Beispiel: $$ SET alt = "...(eins(zwei)drei)..."
          $$ SET hlf = MARK (alt, "@", " (::", " :):" )
          Ergebnis: hlf = "...@1(eins@2(zwei@2)drei@1)..."

          $$ SET hlf = EXTRACT (hlf, " :@1(::"|, |":@1):" )
          $$ SET neu = EXCHANGE (hlf, " :@{\0}::" )
          Ergebnis: hlf = "eins@2(zwei@2)drei"
                  neu = "eins(zwei)drei"
```

```
EXTRACT (var, apos, epos, num, anz, erg)
```

Die erweiterte Makrofunktion `EXTRACT` bestimmt im Inhalt der Variablen `var` einzelne Bereiche, wählt einen oder mehrere davon aus und liefert sie als Funktionswert; falls die Variable `var` eine Sternvariable ist, erfolgt das Auswählen zeilenweise.

Der erste Bereich wird in gleicher Weise wie bei der einfachen Makrofunktion `EXTRACT` (siehe Seite 519) mit den Argumenten `apos` und `epos` ausgewählt. Der zweite und die folgenden Bereiche werden so ausgewählt, als würde der Inhalt der Variablen erst unmittelbar nach dem Ende des jeweils zuvor ausgewählten Bereichs beginnen. Ist für die Position `epos` eine Such-Tabelle angegeben und enden

die Bereiche jeweils vor einer gefundenen Suchzeichenfolge, so kann durch Angabe eines senkrechten Striches unmittelbar vor und nach der Such-Tabelle bestimmt werden, dass nachfolgende Bereiche so ausgewählt werden, als würde der Inhalt der Variablen erst unmittelbar nach dem Ende der gefundenen Zeichenfolge beginnen.

Mit dem Argument `num` wird festgelegt, der wievielte Bereich als erster ausgewählt wird; hat `num` den Wert 0 (Null), so wird nur der letzte Bereich ausgewählt. Mit dem Argument `anz` wird festgelegt, wieviele aufeinander folgende Bereiche insgesamt ausgewählt werden; hat das Argument `anz` den Wert 0 (Null), so werden alle nachfolgenden Bereiche ausgewählt. Für die Argumente `num` und `anz` kann eine Zahl oder eine Variable, die eine Zahl enthält, angegeben werden.

Mit dem Argument `erg` kann eine Variable angegeben werden, deren Inhalt jeweils zwischen den ausgewählten Bereichen ergänzt wird; an Stelle der Variablen kann auch eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben werden. Wenn zwischen den ausgewählten Bereichen nichts ergänzt werden soll, kann das Argument `erg` einschließlich des davor stehenden Kommas weggelassen werden. Soll außerdem nur ein einziger Bereich ausgewählt werden, so kann auch das Argument `anz` einschließlich des davor stehenden Kommas weggelassen werden.

```
Beispiel: $$ SET alt = "eins;zwei;drei;vier;fünf"
          $$ SET neu = EXTRACT (alt, 0, "/;/"/, 3)
```

```
Ergebnis: neu = "drei;"
```

```
$$ SET neu = EXTRACT (alt, 0, |"/;/"/, 3)
```

```
Ergebnis: neu = "drei"
```

```
$$ SET neu = EXTRACT (alt, 0, |"/;/"/, 3, 2, "/")
```

```
Ergebnis: neu = "drei/vier"
```

```
$$ BUILD S_TABLE anf = ":@A :@C :@D :@E :"
```

```
$$ BUILD S_TABLE end = " :@:"
```

```
$$ SET alt = "@A eins @B zwei @C drei @D vier"
```

```
$$ SET neu = EXTRACT (alt, anf|, end, 1, 0, ";")
```

```
Ergebnis: neu = "eins;drei;vier"
```

```
STRINGS (var, stab, apos, epos, snum, sanz, erg)
```

Die Makrofunktion `STRINGS` nimmt den Inhalt der Variablen `var` und extrahiert von der Position `apos` (einschließlich) bis zur Position `epos` (ausschließlich) die Zeichenfolgen, die mit einer in der Such-Tabelle `stab` angegebenen Suchzeichenfolge übereinstimmen, und liefert das Ergebnis als Funktionswert; falls die Variable `var` eine Sternvariable ist, erfolgt das Extrahieren zeilenweise.

Für das Argument `stab` kann der Name einer mit der `BUILD`-Anweisung definierten Such-Tabelle oder direkt eine in Anführungszeichen eingeschlossene Such-Tabelle angegeben werden.

Soll im gesamten Inhalt der Variablen `var` nach den in der Such-Tabelle angegebenen Zeichenfolgen gesucht werden, so kann für die Argumente `apos` und `epos` jeweils 0 (Null) angegeben werden; andernfalls kann mit diesen beiden Argumenten in gleicher Weise wie bei der Makrofunktion `EXTRACT` (siehe Seite 519) ein Bereich ausgewählt werden.

Mit dem Argument `snum` wird festgelegt, die wievielte gefundene Zeichenfolge als erste extrahiert wird; hat das Argument `snum` den Wert 0 (Null), so wird nur die letzte gefundene Zeichenfolge extrahiert. Mit dem Argument `sanz` wird festgelegt, wieviele Zeichenfolgen maximal extrahiert werden; hat das Argument `sanz` den Wert 0 (Null), so werden alle nachfolgenden Zeichenfolgen extrahiert. Für die Argumente `snum` und `sanz` kann jeweils eine Zahl oder eine Variable, die eine Zahl enthält, angegeben werden.

Zwischen den extrahierten Zeichenfolgen wird jeweils die mit dem Argument `erg` angegebene Zeichenfolge ergänzt. Für das Argument `erg` kann eine Variable oder eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben werden. Falls zwischen den Zeichenfolgen ein Apostroph ergänzt werden soll, kann das Argument `erg` einschließlich des davor stehenden Kommas weggelassen werden.

Falls das Argument `erg` weggelassen wird, kann der Aufruf der Makrofunktion ggf. noch weiter verkürzt werden: Falls alle nachfolgenden Zeichenfolgen extrahiert werden sollen, kann das Argument `sanz` einschließlich des davor stehenden Kommas weggelassen werden; falls außerdem ab der ersten Zeichenfolge extrahiert werden soll, kann auch das Argument `snum` einschließlich des davor stehenden Kommas weggelassen werden; falls zusätzlich im gesamten Inhalt der Variablen `var` nach Zeichenfolgen gesucht werden soll, können auch noch die beiden Argumente `apos` und `epos` einschließlich des jeweils davor stehenden Kommas weggelassen werden.

```
Beispiel: $$ SET text = "... Seite 123 bis 456 ..."
          $$ SET seiten = STRINGS (text, ":{#}:")
          Ergebnis: seiten = "123'456"
```

`ELIMINATE (var, apos, epos)`

Diese Makrofunktion `ELIMINATE` nimmt den Inhalt der Variablen `var`, entfernt die Zeichen von der Position `apos` (einschließlich) bis zur Position `epos` (ausschließlich) und liefert das Ergebnis als Funktionswert; falls die Variable `var` eine Sternvariable ist, erfolgt das Entfernen zeilenweise.

Für die Argumente `apos` und `epos` kann jeweils eine Zahl, eine Variable, die eine Zahl (ohne Vorzeichen) enthält, oder eine Such-Tabelle angegeben werden. Vor einer Zahl kann ein Vorzeichen angegeben werden; vor einer Variablen muss ein Vorzeichen angegeben werden, damit der Name der Variablen nicht als Name einer Such-Tabelle interpretiert wird. Als Such-Tabelle kann entweder der Name einer mit der `BUILD`-Anweisung definierten Such-Tabelle oder direkt eine in Anführungszeichen eingeschlossene Such-Tabelle angegeben werden.

Wird eine Zahl oder Variable angegeben, so werden die Zeichenpositionen bei positiven Werten von links nach rechts gezählt, bei negativen von rechts nach

links. Soll der Inhalt der Variablen vom Anfang an bzw. bis zum Ende eliminiert werden, kann 0 (Null) als Position angegeben werden.

```
Beispiel: $$ SET alt = "123456789"
          $$ SET neu = ELIMINATE (alt, 4, 8)
```

```
Ergebnis: neu = "12389"
```

```
          $$ SET neu = ELIMINATE (alt, -4, -2)
```

```
Ergebnis: neu = "1234589"
```

Wird für die Anfangsposition eine Such-Tabelle angegeben, so wird in der Variablen `var` von Anfang an nach einer Zeichenfolge gesucht, die mit einer in der Such-Tabelle angegebenen Suchzeichenfolge übereinstimmt. Falls für die Endposition eine Zahl angegeben ist, endet die Suche spätestens vor der entsprechenden Position; falls ebenfalls eine Such-Tabelle angegeben ist, wird ggf. bis zum Ende des Variablen-Inhalts gesucht. Als Anfangsposition gilt die Position des ersten Zeichens der gefundenen Zeichenfolge. Falls keine entsprechende Zeichenfolge gefunden wird, ist das Ergebnis der Makrofunktion der gesamte Inhalt der Variablen `var`.

```
Beispiel: $$ SET alt = "Tübingen/Neckar"
          $$ SET neu = ELIMINATE (alt, ":/:", 0)
```

```
Ergebnis: neu = "Tübingen"
```

Durch die Angabe eines senkrechten Striches entweder unmittelbar vor oder unmittelbar nach der Such-Tabelle `apos` kann explizit bestimmt werden, ob die zu eliminierende Zeichenfolge mit dem ersten Zeichen oder erst nach dem letzten Zeichen der gefundenen Zeichenfolge beginnt.

```
Beispiel: $$ SET alt = "Tübingen/Neckar"
          $$ SET neu = ELIMINATE (alt, ":/:"|, 0)
```

```
Ergebnis: neu = "Tübingen/"
```

Wird für die Endposition eine Such-Tabelle angegeben, so wird nach einer Zeichenfolge gesucht, die mit einer in der Such-Tabelle angegebenen Suchzeichenfolge übereinstimmt. Die Suche nach einer solchen Zeichenfolge beginnt mit der angegebenen Anfangsposition bzw. mit der auf das letzte Zeichen der für die Anfangsposition gefundenen Zeichenfolge folgenden Position. Als Endposition gilt die Position des ersten Zeichens der gefundenen Zeichenfolge. Falls keine entsprechende Zeichenfolge gefunden wird, wird der Inhalt der Variablen bis zum Ende eliminiert.

```
Beispiel: $$ SET alt = "Tübingen/Neckar"
          $$ SET neu = ELIMINATE (alt, 0, ":/:")
```

```
Ergebnis: neu = "/Neckar"
```

Durch die Angabe eines senkrechten Striches entweder unmittelbar vor oder unmittelbar nach der Such-Tabelle `epos` kann explizit bestimmt werden, ob die zu

eliminierende Zeichenfolge vor dem ersten Zeichen oder erst nach dem letzten Zeichen der gefundenen Zeichenfolge endet.

Beispiel: `$$ SET alt = "Tübingen/Neckar"`  
`$$ SET neu = ELIMINATE (alt, 0, " /: ") |`  
 Ergebnis: `neu = "Neckar"`

Zum Eliminieren eines eingeklammelter Textes kann die öffnende Klammer in der Such-Tabelle für die Anfangsposition und die schließende Klammer in der Such-Tabelle für die Endposition definiert werden.

Beispiel: `$$ SET alt = "eins(zwei)drei"`  
`$$ SET neu = ELIMINATE (alt, |": (:", ":):" |)`  
 Ergebnis: `neu = "einsdrei"`

Sind die Klammern jedoch geschachtelt, liefert die Makrofunktion den gewünschten Text vermutlich unvollständig, weil innere und äußere Klammern nicht unterschieden werden:

Beispiel: `$$ SET alt = "eins(zwei(drei)vier)fünf"`  
`$$ SET neu = ELIMINATE (alt, |": (:", ":):" |)`  
 Ergebnis: `neu = "einsvier)fünf"`

Werden die Klammern zuvor mit Markierungen versehen, können innere und äußere Klammern unterschieden werden.

Beispiel: `$$ SET alt = "eins(zwei(drei)vier)fünf"`  
`$$ SET hlf = MARK (alt, "@", ":(:", ":):")`  
 Ergebnis: `hlf = "eins@1(zwei@2(drei@2)vier@1)fünf"`  
`$$ SET neu = ELIMINATE (hlf, |":@1 (:", ":@1):" |)`  
`neu = "einsfünf"`

`ELIMINATE (var, apos, epos, num, anz, erg)`

Die erweiterte Makrofunktion `ELIMINATE` bestimmt im Inhalt der Variablen `var` einzelne Bereiche, eliminiert einen oder mehrere davon und liefert das Ergebnis als Funktionswert; falls die Variable `var` eine Sternvariable ist, werden die Bereiche zeilenweise bestimmt.

Der erste Bereich wird in gleicher Weise wie bei der einfachen Makrofunktion `ELIMINATE` (siehe Seite 523) mit den Argumenten `apos` und `epos` ausgewählt. Der zweite und die folgenden Bereiche werden so ausgewählt, als würde der Inhalt der Variablen erst unmittelbar nach dem Ende des jeweils zuvor ausgewählten Bereichs beginnen. Ist für die Position `epos` eine Such-Tabelle angegeben und enden die Bereiche jeweils vor einer gefundenen Suchzeichenfolge, so kann durch Angabe eines senkrechten Striches unmittelbar vor und nach der Such-Tabelle bestimmt werden, dass nachfolgende Bereiche so ausgewählt werden, als würde

der Inhalt der Variablen erst unmittelbar nach dem Ende der gefundenen Zeichenfolge beginnen.

Mit dem Argument `num` wird festgelegt, der wievielte Bereich als erster eliminiert wird; hat das Argument `num` den Wert 0 (Null), so wird nur der letzte Bereich eliminiert. Mit dem Argument `anz` wird festgelegt, wieviele aufeinander folgende Bereiche insgesamt eliminiert werden; hat das Argument `anz` den Wert 0 (Null), so werden alle nachfolgenden Bereiche eliminiert. Für die Argumente `num` und `anz` kann jeweils eine Zahl oder eine Variable, die eine Zahl enthält, angegeben werden.

Mit dem Argument `erg` kann eine Variable angegeben werden, deren Inhalt jeweils zwischen den nicht eliminierten Bereichen ergänzt wird; an Stelle der Variablen kann auch eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben werden. Wenn zwischen den nicht eliminierten Bereichen nichts ergänzt werden soll, kann das Argument `erg` einschließlich des davor stehenden Kommas weggelassen werden. Soll außerdem nur ein einziger Bereich eliminiert werden, so kann auch das Argument `anz` einschließlich des davor stehenden Kommas weggelassen werden.

```
Beispiel: $$ SET alt = "eins;zwei;drei;vier;fünf"
          $$ SET neu = ELIMINATE (alt, 0, "/;/"/|, 3)

Ergebnis: neu = "eins;zwei;vier;fünf"

          $$ SET neu = ELIMINATE (alt, 0, "/;/"/|, 3, 2)

Ergebnis: neu = "eins;zwei;fünf"

          $$ BUILD S_TABLE anf = ":@A :@C :@E :@"
          $$ BUILD S_TABLE end = ":@:"
          $$ SET alt = "@A eins @B zwei @C drei @D vier"
          $$ SET neu = ELIMINATE (alt, anf, end, 1, 0)

Ergebnis: neu = "@B zwei @D vier"
```

`SUBSTITUTE (var, -, apos, epos, erg)`

Diese Makrofunktion `SUBSTITUTE` nimmt den Inhalt der Variablen `var` und ersetzt die Zeichen von Position `apos` (einschließlich) bis zur Position `epos` (ausschließlich) durch den Inhalt der Variablen `erg` und liefert das Ergebnis als Funktionswert.

Mit den beiden Argumenten `apos` und `epos` kann in gleicher Weise wie bei der Makrofunktion `EXTRACT` (siehe Seite 519) der zu ersetzende Bereich ausgewählt werden.

An Stelle der Variablen `erg` kann auch eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben werden.

```
Beispiel: $$ SET alt = "123456789"
          $$ SET neu = SUBSTITUTE (alt, -, 4, 7, "abcd")
```

Ergebnis: neu = "123abcd789"

SUBSTITUTE (var, stab, apos, epos, erg, snum, sanz)

Die erweiterte Makrofunktion SUBSTITUTE nimmt den Inhalt der Variablen `var` und ersetzt von der Position `apos` (einschließlich) bis zur Position `epos` (ausschließlich) die Zeichenfolgen, die mit einer in der Such-Tabelle `stab` angegebenen Suchzeichenfolge übereinstimmen, durch die Teilzeichenfolgen, die in der Variablen `erg` enthalten sind, und liefert das Ergebnis als Funktionswert.

Für das Argument `stab` kann der Name einer mit der BUILD-Anweisung definierten Such-Tabelle oder direkt eine in Anführungszeichen eingeschlossene Such-Tabelle angegeben werden. Wird unmittelbar vor oder unmittelbar nach dem Argument `stab` ein senkrechter Strich angegeben, so werden die Zeichenfolgen, die mit einer in der Such-Tabelle angegebenen Zeichenfolgen übereinstimmen, nicht ersetzt, sondern mit ins Ergebnis übernommen und die Teilzeichenfolgen aus der Variablen `erg` werden jeweils davor bzw. danach eingefügt.

Soll der gesamte Inhalt der Variablen `var` nach den in der Such-Tabelle angegebenen Zeichenfolgen durchsucht werden, so kann für die Argumente `apos` und `epos` jeweils 0 (Null) angegeben werden; andernfalls kann mit diesen beiden Argumenten in gleicher Weise wie bei der Makrofunktion EXTRACT (siehe Seite 519) ein Bereich ausgewählt werden.

Mit dem Argument `snum` wird festgelegt, die wievielte gefundene Zeichenfolge als erste ersetzt wird; hat `snum` den Wert 0 (Null), so wird nur die letzte gefundene Zeichenfolge ersetzt. Mit dem Argument `sanz` wird festgelegt, wieviele Zeichenfolgen maximal ersetzt werden; hat das Argument `sanz` den Wert 0 (Null), so werden maximal soviele Zeichenfolgen ersetzt, wie die Variable `erg` Teilzeichenfolgen enthält. Für die Argumente `snum` und `sanz` kann eine Zahl oder eine Variable, die eine Zahl enthält, angegeben werden.

Falls maximal soviele Zeichenfolgen ersetzt werden sollen, wie die Variable `erg` Teilzeichenfolgen enthält, kann das Argument `sanz` einschließlich des davor stehenden Kommas weggelassen werden; falls außerdem ab der ersten gefundenen Zeichenfolge ersetzt werden soll, kann auch das Argument `snum` einschließlich des davor stehenden Kommas weggelassen werden.

Die erste zu ersetzende Zeichenfolge wird durch die erste Teilzeichenfolge der Variablen `erg` ersetzt, die zweite durch die zweite usw. Falls die Teilzeichenfolgen in `erg` erschöpft sind und weitere Zeichenfolgen zu ersetzen sind, werden die restlichen Zeichenfolgen jeweils durch die letzte Teilzeichenfolge ersetzt; sind mehr Teilzeichenfolgen vorhanden als Zeichenfolgen zu ersetzt sind, bleiben sie unberücksichtigt. Falls die Variable `erg` eine Sternvariable ist, wird statt einer Teilzeichenfolge jeweils der Inhalt einer Zeile eingesetzt.

```
Beispiel: $$ SET alt = "... insgesamt 123 Einträge ..."
          $$ SET wert = "4567"
          $$ SET neu = SUBSTITUTE (alt, ":{#}:", 0,0, wert)
          Ergebnis: neu = "... insgesamt 4567 Einträge ..."
```

```

$$ SET alt = "... von (?) bis (?) ..."
$$ SET werte = "Tübingen'München"
$$ SET neu = SUBSTITUTE (alt, ":(\?):", 0,0,werte)

Ergebnis: neu = "... von Tübingen bis München ..."

```

SUBSTITUTE (var, +, apos, epos, trenner, name, modus, num, anz)

Diese alternative Makrofunktion SUBSTITUTE bestimmt im Inhalt der Variablen `var` einzelne Bereiche, unterteilt jeden Bereich in »Schlüsselwörter«, tauscht diese entsprechend dem Wörterbuch `name` (jeweils gegen den ersten Textteil) aus und liefert das Ergebnis als Funktionswert; falls die Variable `var` eine Sternvariable ist, werden die Bereiche zeilenweise bestimmt.

Die Bereiche werden in gleicher Weise wie bei der erweiterten Makrofunktion EXTRACT (siehe Seite 521) mit den Argumenten `apos` und `epos` sowie `num` und `anz` ausgewählt. Falls nur im ersten Bereich ausgetauscht werden soll, können die Argumente `num` und `anz` jeweils einschließlich des davor stehenden Kommas weggelassen werden.

Die Unterteilung der Bereiche in »Schlüsselwörter« erfolgt an allen Zeichenfolgen, die in der Such-Tabelle `trenner` als Suchzeichenfolgen enthalten sind. Für das Argument `trenner` kann der Name einer mit der BUILD-Anweisung definierten Such-Tabelle oder direkt eine in Anführungszeichen eingeschlossene Such-Tabelle angegeben werden. Soll jeweils der ganze Bereich als ein »Schlüsselwort« gelten, kann für das Argument `trenner` ein Minuszeichen angegeben werden.

Für das Argument `name` muss der Name eines mit der DICTIONARY-Anweisung erstellten Wörterbuchs angegeben werden (siehe Kapitel »Wörterbücher« ab Seite 445).

Für das Argument `modus` muss entweder EXCHANGE oder INFIX angegeben werden; damit wird die Aktion festgelegt, in der das Wörterbuch abgefragt wird.

```

Beispiel: $$ DICTIONARY w CREATE/INFIX
          $$ DICT w ADD "Beispiel", num,cnt, "Bei\spiel"
          $$ DICT w ADD "schlafende", num,cnt, "schla\fende"
          $$ BUILD S_TABLE t = ":{00}{@}::%{1--2}{%}:"
          $$ SET alt = "Schlafende Hunde, schlafende Katzen"
          $$ SET neu = SUBSTITUTE (alt, +, 0,0, t, w, INFIX)

Ergebnis: "Schla\fende Hunde, schla\fende Katzen"

```

RENUMBER (var, stab, apos, epos, num, schr, anz)

Die Makrofunktion RENUMBER sucht im Inhalt der Variablen `var` von der Position `apos` (einschließlich) bis zur Position `epos` (ausschließlich) nach Zeichenfolgen, die mit einer in der Such-Tabelle `stab` angegebenen Suchzeichenfolge übereinstimmen, entfernt jeweils die unmittelbar dahinter stehenden Ziffern, setzt an diesen Stellen jeweils die neue Nummer ein und liefert das Ergebnis als Funktionswert; falls die Variable `var` eine Sternvariable ist, erfolgt das Nummerieren zeilenweise.



Für das Argument `stab` kann der Name einer mit der `BUILD`-Anweisung definierten Such-Tabelle oder direkt eine in Anführungszeichen eingeschlossene Such-Tabelle angegeben werden.

Soll der gesamte Inhalt der Variablen `var` durchsucht werden, so kann für die Argumente `apos` und `epos` jeweils 0 (Null) angegeben werden; andernfalls kann mit diesen beiden Argumenten in gleicher Weise wie bei der Makrofunktion `EXTRACT` (siehe Seite 519) ein Bereich ausgewählt werden.

Mit der Variablen `num` wird die neue Nummer vorgegeben.

Mit dem Argument `schr` wird festgelegt, um wieviel die in der Variablen `num` enthaltene Nummer nach jedem Einsetzen erhöht werden soll. Für das Argument `schr` kann eine Zahl oder eine Variable, die eine Zahl enthält, angegeben werden.

Mit dem Argument `anz` wird festgelegt, auf wieviele Stellen die neue Nummer vor dem Einsetzen mit Nullen aufgefüllt werden soll. Für das Argument `anz` kann eine Zahl oder eine Variable, die eine Zahl enthält, angegeben werden.

Falls die neuen Nummern nicht mit führenden Nullen ergänzt werden sollen, kann das Argument `anz` einschließlich des davor stehenden Kommas weggelassen werden; falls außerdem die Nummer nur einmal eingesetzt und danach nicht erhöht werden soll, kann auch das Argument `schr` einschließlich des davor stehenden Kommas weggelassen werden.

```
Beispiel: $$ SET lnr = 1
          $$ SET alt = "Beispiel[48] mit lfd. Nummern[]"
          $$ SET neu = RENUMBER (alt, ":\[:", 0, 0, lnr, 1)

          Ergebnis: neu = "Beispiel[1] mit lfd. Nummern[2]"
```

`MARK (var, erg, klauf, klzu, num)`

Die Makrofunktion `MARK` sucht im Inhalt der Variablen `var` nach öffnenden und schließenden Klammern oder nach Zeichenfolgen, markiert diese und liefert das Ergebnis als Funktionswert; falls die Variable `var` eine Sternvariable ist, erfolgt das Markieren zeilenweise.

Die Markierung wird jeweils unmittelbar vor den Klammern bzw. Zeichenfolgen eingefügt und besteht aus der in der Variablen `erg` enthaltenen Zeichenfolge und einer Nummer. An Stelle der Variablen `erg` kann auch eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben werden.

Die Nummer beginnt bei der in der Variablen `num` enthaltenen Zahl, wird vor jeder öffnenden Klammer (vor dem Einsetzen) um 1 erhöht und nach jeder schließenden Klammer (nach dem Einsetzen) um 1 erniedrigt, falls sie dadurch nicht kleiner als 0 wird. Am Ende wird der Variablen `num` der aktuelle Wert der Nummer wieder zugewiesen.

Als öffnende bzw. schließende Klammern dienen alle in der Such-Tabelle `klau` bzw. `klzu` angegebenen Suchzeichenfolgen. Für die Argumente `klau` und `klzu` kann der Name einer mit der `BUILD`-Anweisung definierten Such-Tabelle oder direkt eine in Anführungszeichen eingeschlossene Such-Tabelle angegeben werden.

Wird statt der Such-Tabelle `klzu` ein Minuszeichen angegeben, so werden die Zeichenfolgen markiert, die mit einer Suchzeichenfolge in der Such-Tabelle `klauf` übereinstimmen, wobei die Nummer jeweils (vor dem Einsetzen) um 1 erhöht wird; das Minuszeichen kann einschließlich des davor stehenden Kommas weggelassen werden, wenn auch das Argument `num` weggelassen wird.

Wird das Argument `num` einschließlich des davor stehenden Kommas weggelassen, ist 0 (Null) der Anfangswert der Nummer.

```
Beispiel: $$ SET alt = "..(..)..(..)...."
          $$ neu = MARK (text, "@", ":", ":", ":")
```

```
Ergebnis: neu = "..@1(..@1)..@1(..@2(..@2)..@1).."
```

```
$$ SET alt = "...;...;...;..."
$$ SET neu = MARK (text, "@", ":", ":", ":")
```

```
Ergebnis: neu = "...@1;...@2;...@3;..."
```

`EXCHANGE (var, xtab, apos, epos, xnum, xanz)`

Diese Makrofunktion `EXCHANGE` nimmt den Inhalt der Variablen `var`, tauscht darin die in der Austausch-Tabelle `xtab` angegebenen Suchzeichenfolgen mit den dazugehörigen Ersatzzeichenfolgen aus und liefert das Ergebnis als Funktionswert; falls die Variable `var` eine Sternvariable ist, erfolgt das Austauschen zeilenweise.

Für das Argument `xtab` kann der Name einer mit der `BUILD`-Anweisung definierten Austausch-Tabelle oder direkt eine in Anführungszeichen eingeschlossene Austausch-Tabelle angegeben werden.

Sollen die Zeichenfolgen im gesamten Inhalt der Variablen `var` ausgetauscht werden, so kann für die Argumente `apos` und `epos` jeweils 0 (Null) angegeben werden; andernfalls kann mit diesen beiden Argumenten in gleicher Weise wie bei der Makrofunktion `EXTRACT` (siehe Seite 519) ein Bereich ausgewählt werden.

Mit dem Argument `xnum` wird festgelegt, die wievielte gefundene Zeichenfolge als erste ausgetauscht wird; hat das Argument `xnum` den Wert 0 (Null), so wird nur die letzte gefundene Zeichenfolge ausgetauscht. Mit dem Argument `xanz` wird festgelegt, wieviele Zeichenfolgen maximal ausgetauscht werden; hat das Argument `xanz` den Wert 0 (Null), so werden alle nachfolgenden Zeichenfolgen ausgetauscht. Für die Argumente `xnum` und `xanz` kann eine Zahl oder eine Variable, die eine Zahl enthält, angegeben werden.

Falls alle nachfolgenden Zeichenfolgen ausgetauscht werden sollen, kann das Argument `xanz` einschließlich des davor stehenden Kommas weggelassen werden; falls außerdem ab der ersten Zeichenfolge ausgetauscht werden soll, kann auch das Argument `xnum` einschließlich des davor stehenden Kommas weggelassen werden; falls zusätzlich im gesamten Inhalt der Variablen `var` nach auszutauschenden Zeichenfolgen gesucht werden soll, können auch noch die beiden Argumente `apos` und `epos` einschließlich des jeweils davor stehenden Kommas weggelassen werden.

```
Beispiel: $$ BUILD X_TABLE tausch = ":wird:ist:"
          $$ SET alt = "Dies wird ein Beispiel"
          $$ SET neu = EXCHANGE (alt, tausch)
```

```
Ergebnis: neu = "Dies ist ein Beispiel"
```

```
$$ BUILD X_TABLE t = "::.;:"
$$ SET alt = "...(1.2.3.4)..."
$$ SET neu = EXCHANGE (alt, t, ":(:", ")::", 2, 1)
```

```
Ergebnis: neu = "...(1.2;3.4)..."
```

```
EXCHANGE (var, xtab, apos, epos, xnum, xanz, num, anz, xtabx)
```

Die erweiterte Makrofunktion `EXCHANGE` bestimmt im Inhalt der Variablen `var` einzelne Bereiche, tauscht innerhalb dieser Bereiche Zeichenfolgen entsprechend der Austausch-Tabelle `xtab`, außerhalb dieser Bereiche entsprechend der Austausch-Tabelle `xtabx` aus und liefert das Ergebnis als Funktionswert; falls die Variable `var` eine Sternvariable ist, werden die Bereiche zeilenweise bestimmt.

Die Bereiche werden in gleicher Weise wie bei der erweiterten Makrofunktion `EXTRACT` (siehe Seite 521) mit den Argumenten `apos` und `epos` sowie `num` und `anz` ausgewählt. Soll nur in einem einzigen Bereich ausgetauscht werden, so kann das Argument `anz` einschließlich des davor stehenden Kommas weggelassen werden, falls auch das Argument `xtabx` weggelassen wird.

Soll innerhalb der Bereiche nichts ausgetauscht werden, kann für das Argument `xtab` ein Minuszeichen angegeben werden; soll außerhalb der Bereiche nichts ausgetauscht werden, kann für das Argument `xtabx` ein Minuszeichen angegeben oder das Argument `xtabx` kann einschließlich des davor stehenden Kommas weggelassen werden.

```
Beispiel: $$ BUILD X_TABLE t = "::.;:"
          $$ BUILD S_TABLE a = ":(:"
          $$ BUILD S_TABLE e = ")::"
          $$ SET alt = "...(1.2.3.4)...(1.2.3.4)..."
          $$ SET neu = EXCHANGE (alt, t, a, e, 2, 1, 1, 0)
```

```
Ergebnis: neu = "...(1.2;3.4)...(1.2;3.4)..."
```

```
$$ BUILD X_TABLE t = "::.i:"
$$ BUILD S_TABLE a = ":(:"
$$ BUILD S_TABLE e = ")::"
$$ BUILD X_TABLE x = "::.a:"
$$ SET alt = "..(..)..(..).."
$$ SET neu = EXCHANGE (alt, t, a, e, 1,0, 1,0, x)
```

```
Ergebnis: neu = "aa(ii)aa(ii)aa"
```

```
$$ SET alt = "..(..)..(..).."
$$ SET neu = EXCHANGE (alt, t, a, e, 1,0, 1,0, x)
```

```

Ergebnis: neu = "aa(ii)aa(ii(ii)aa)aa"

$$ BUILD X_TABLE t = "::.i:@{\0}::"
$$ BUILD S_TABLE a = ":@1(:"
$$ BUILD S_TABLE e = ":@1):"
$$ BUILD X_TABLE x = "::.a:"
$$ SET alt = "..(..)..(..(..)).."
$$ SET hlf = MARK (alt, "@", ":(:", ")::")
$$ SET neu = EXCHANGE (hlf, t, |a, e|, 1,0,1,0, x)

Ergebnis: hlf = "..@1(..@1)..@1(..@2(..@2)..@1).."
           neu = "aa(ii)aa(ii(ii)ii)aa"

```

In Sonderfällen kann es sinnvoll sein, das Austauschen nicht nur einmal, sondern vom jeweiligen Ergebnis ausgehend wiederholt auszuführen. Hierzu kann nach dem Argument `xtabx` durch Komma getrennt noch ein Argument `nma1` angegeben werden. Für das Argument `nma1` kann eine Zahl oder eine Variable, die eine Zahl enthält, angegeben werden. Diese Zahl bestimmt wie oft das Austauschen auf Grund der Austausch-Tabelle `xtab` maximal ausgeführt werden soll; sind keine auszutauschenden Zeichenfolgen mehr vorhanden, so wird das Austauschen beendet. Wird für `nma1` der Wert 0 (Null) angegeben, so wird das Austauschen wiederholt, bis keine auszutauschenden Zeichenfolgen mehr vorhanden sind.

```

Beispiel: $$ BUILD S_GROUP s:bb = ": {&a}:"
          $$ BUILD X_TABLE t = *
          :{#}{&a}{0-0}{s:bb} {+1=}:{+1-4=}:
          $$ SET alt = "1a 3b 3c 3d 4b 4e 4f 4g 4i 67a 67b"
          $$ SET neu = EXCHANGE (alt, t, 0,0, 1,0,1,0, -, 0)

Ergebnis: neu = "1a 3b c d 4b e f g i 67a b"

```

#### Interne Zwischenergebnisse nach dem

1. Austauschen: 1a 3b c 3d 4b e 4f g 4i 67a b
2. Austauschen: 1a 3b c d 4b e f g 4i 67a b
3. Austauschen: 1a 3b c d 4b e f g i 67a b

ENCLOSE (var, -, apos, epos, anfang, ende)

Diese Makrofunktion `ENCLOSE` nimmt den Inhalt der Variablen `var` und umschließt die Zeichen von Position `apos` (einschließlich) bis zur Position `epos` (ausschließlich) mit dem Inhalt der Variablen `anfang` und mit dem Inhalt der Variablen `ende` und liefert das Ergebnis als Funktionswert.

Mit den beiden Argumenten `apos` und `epos` kann in gleicher Weise wie bei der Makrofunktion `EXTRACT` (siehe Seite 519) der zu umschließende Bereich ausgewählt werden.

An Stelle der Variablen `anfang` und `ende` kann jeweils auch eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben werden.

Beispiel: `$$ SET alt = "123456789"`  
`$$ SET neu = ENCLOSE (alt, -, 4, 7, "(", ")")`  
 Ergebnis: `neu = "123(456)789"`

`ENCLOSE (var, stab, apos, epos, anfang, ende, zwischen)`

Diese Makrofunktion `ENCLOSE` nimmt den Inhalt der Variablen `var` und sucht zwischen der Position `apos` (einschließlich) und Position `epos` (ausschließlich) die in der Such-Tabelle `stab` enthalten Zeichenfolgen. Vor jeder gefundenen Zeichenfolge wird der Inhalt der Variablen `anfang` und nach jeder gefundenen Zeichenfolge der Inhalt der Variablen `ende` eingefügt. Folgen zwei gefundene Zeichenfolgen unmittelbar aufeinander, so wird dazwischen nur der Inhalt der Variablen `zwischen` eingefügt. Das Ergebnis wird als Funktionswert geliefert.

Für das Argument `stab` kann der Name einer mit der `BUILD`-Anweisung definierten Such-Tabelle oder direkt eine in Anführungszeichen eingeschlossene Such-Tabelle angegeben werden.

Soll der gesamte Inhalt der Variablen `var` nach den in der Such-Tabelle angegebenen Zeichenfolgen durchsucht werden, so kann für die Argumente `apos` und `epos` jeweils 0 (Null) angegeben werden; andernfalls kann mit diesen beiden Argumenten in gleicher Weise wie bei der Makrofunktion `EXTRACT` (siehe Seite 519) ein Bereich ausgewählt werden.

Soll zwischen zwei gefundenen Zeichenfolgen, die unmittelbar aufeinanderfolgen, der Inhalt der Variablen `anfang` und `ende` eingefügt werden, so kann die Variable `zwischen` einschließlich des davor stehenden Kommas weggelassen werden.

An Stelle der Variablen `anfang`, `ende` und `zwischen` kann jeweils auch eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben werden.

Beispiel: `$$ BUILD S_TABLE kursiv = ":Haus:Hof:Hund:"`  
`$$ SET ka = "<i>", ke = "</i>"`  
`$$ SET alt = "Der Hofhund lag im Hof"`  
`$$ SET neu = ENCLOSE (alt, kursiv, 0,0, ka,ke, "")`  
 Ergebnis:  
`neu = "Der <i>Hofhund</i> lag im <i>Hof</i>"`

`HIGHLIGHT (var, rtab, apos, epos, anfang, ende, xtab)`

Die Makrofunktion `HIGHLIGHT` nimmt den Inhalt der Variablen `var` und sucht zwischen der Position `apos` (einschließlich) und Position `epos` (ausschließlich) die in der Recherchier-Tabelle `rtab` enthalten Zeichenfolgen. Vor jedem Wort, in dem eine Zeichenfolge gefunden wurde, wird der Inhalt der Variablen `anfang` und nach jedem Wort, in dem eine Zeichenfolge gefunden wurde, der Inhalt der Variablen `ende` eingefügt. Als Wort gilt in diesem Fall jede Zeichenfolge, die durch Leerzeichen begrenzt ist. Das Ergebnis wird als Funktionswert geliefert.

Für das Argument `rtab` kann der Name einer mit der `BUILD`-Anweisung definierten Recherchier-Tabelle oder direkt eine in Anführungszeichen eingeschlossene Recherchier-Tabelle angegeben werden.

Wird für das Argument `rtab` eine in Anführungszeichen eingeschlossene Recherchier-Tabelle angegeben, so werden automatisch die Optionen `TEXT` und `OR` angenommen. Falls andere Optionen (siehe Seite 441) erforderlich sind, muss die Tabelle mit der `BUILD`-Anweisung definiert und der Name der Tabelle angegeben werden.

Soll der gesamte Inhalt der Variablen `var` nach den in der Recherchier-Tabelle angegebenen Zeichenfolgen durchsucht werden, so kann für die Argumente `apos` und `epos` jeweils 0 (Null) angegeben werden; andernfalls kann mit diesen beiden Argumenten in gleicher Weise wie bei der Makrofunktion `EXTRACT` (siehe Seite 519) ein Bereich ausgewählt werden.

An Stelle der Variablen `anfang` und `ende` kann jeweils auch eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben werden.

Für das Argument `xstab` kann der Name einer mit der `BUILD`-Anweisung definierten Austausch-Tabelle oder direkt eine in Anführungszeichen eingeschlossene Austausch-Tabelle angegeben werden.

Mit dem Argument `xstab` kann vorgegeben werden, welche Zeichenfolgen (z. B. Akzente) eliminiert oder durch andere ersetzt werden sollen, bevor geprüft wird, ob eine mit dem Argument `rtab` angegebene Zeichenfolge vorkommt. Es dürfen jedoch keine Leerzeichen entfernt oder eingefügt werden. Falls ein Eliminieren/Ersetzen von Zeichenfolgen nicht erforderlich ist, kann das Argument `xstab` einschließlich des davor stehenden Kommas weggelassen werden.

```
Beispiel: $$ BUILD R_TABLE/OR fett1 = ":Haus:Hof:"
          $$ BUILD R_TABLE/WORD/OR fett2 = ":Haus:Hof:"
          $$ SET fa = "<b>", fe = "</b>"
          $$ SET alt = "Der Hofhund lag im Hof"
          $$ SET neu1 = HIGHLIGHT (alt, fett1, 0, 0, fa, fe)
          $$ SET neu2 = HIGHLIGHT (alt, fett2, 0, 0, fa, fe)
```

Ergebnis:

```
neu1 = "Der <b>Hofhund</b> lag im <b>Hof</b>"
neu2 = "Der Hofhund lag im <b>Hof</b>"
```

`IDENTIFY (var, stab, apos, epos, npos)`

Die Makrofunktion `IDENTIFY` vergleicht die Zeichenfolge, die in der Variablen `var` von der Position `apos` (einschließlich) bis zur Position `epos` (ausschließlich) steht, mit den in der Such-Tabelle `stab` angegebenen Suchzeichenfolgen.

Für das Argument `stab` kann der Name einer mit der `BUILD`-Anweisung definierten Such-Tabelle oder direkt eine in Anführungszeichen eingeschlossene Such-Tabelle angegeben werden.

Soll der gesamte Inhalt der Variablen `var` mit den in der Such-Tabelle angegebenen Zeichenfolgen verglichen werden, so kann für die Argumente `apos` und `epos` jeweils 0 (Null) angegeben werden; andernfalls kann mit diesen beiden Argumenten in gleicher Weise wie bei der Makrofunktion `EXTRACT` (siehe Seite 519) ein Bereich ausgewählt werden.

Falls der Anfang der Zeichenfolge (oder die ganze Zeichenfolge) in der Variablen `var` mit einer Zeichenfolge aus der Such-Tabelle übereinstimmt, gibt der Funktionswert die Nummer der entsprechenden Zeichenfolge in der Such-Tabelle an und der Variablen `npos` wird die Position des Zeichens zugewiesen, das in der Variablen `var` nach der übereinstimmenden Zeichenfolge steht; andernfalls ist der Funktionswert und der Inhalt der Variablen `npos` 0 (Null).

Falls die Position des Zeichens nach der übereinstimmenden Zeichenfolge nicht benötigt wird, kann das Argument `npos` einschließlich des davor stehenden Kommas weggelassen werden; falls außerdem der gesamte Inhalt der Variablen `var` mit den in der Such-Tabelle angegebenen Zeichenfolgen verglichen werden soll, können auch die beiden Argumente `apos` und `epos` einschließlich des jeweils davor stehenden Kommas weggelassen werden.

```
Beispiel: $$ BUILD S_TABLE kennung = " :@n :@s :@o :@l :"  
          $$ SET text = "@o Tübingen"  
          $$ SET pos = IDENTIFY (text, kennung)
```

```
Ergebnis: pos = 3
```

Falls die Variable `var` eine Sternvariable ist, erfolgt das oben beschriebene Vergleichen für jede Zeile einzeln; die sich ergebenden Werte werden durch Apostroph getrennt als Funktionswert geliefert bzw. in der Variablen `npos` gespeichert.

```
FIND_STRING (var, such, apos, epos, n, npos)
```

Die Makrofunktion `FIND_STRING` sucht im Inhalt der Variablen `var` von der Position `apos` (einschließlich) bis zur Position `epos` (ausschließlich) nach der `n`-ten bzw. letzten Zeichenfolge, die mit der in der Variablen `such` enthaltenen Zeichenfolge übereinstimmt. Groß- und Kleinbuchstaben gelten dabei als gleichwertig.

An Stelle der Variablen `such` kann auch eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben werden.

Soll der gesamte Inhalt der Variablen `var` durchsucht werden, so kann für die Argumente `apos` und `epos` jeweils 0 (Null) angegeben werden; andernfalls kann mit diesen beiden Argumenten in gleicher Weise wie bei der Makrofunktion `EXTRACT` (siehe Seite 519) ein Bereich ausgewählt werden.

Mit dem Argument `n` wird festgelegt, die wievielte Zeichenfolge gesucht wird; hat das Argument `n` den Wert 0 (Null), so wird die letzte gesucht. Für das Argument `n` kann eine Zahl oder eine Variable, die eine Zahl enthält, angegeben werden.

Der Funktionswert gibt die Position des ersten Zeichens der gefundenen Zeichenfolge an; der Variablen `npos` wird die Position des Zeichens zugewiesen, das nach der gefundenen Zeichenfolge steht; falls keine Zeichenfolge bzw. weniger als `n` Zeichenfolgen gefunden wurden, ist der Funktionswert und der Inhalt der Variablen `npos` 0 (Null).

Falls die Position des Zeichens nach der gefundenen Zeichenfolge nicht benötigt wird, kann das Argument `npos` einschließlich des davor stehenden Kommas weggelassen werden; falls außerdem die erste Zeichenfolge gesucht werden soll, kann auch das Argument `n` einschließlich des davor stehenden Kommas weggelassen

werden; falls zusätzlich der gesamte Inhalt der Variablen `var` durchsucht werden soll, können auch noch die beiden Argumente `apos` und `epos` einschließlich des jeweils davor stehenden Kommas weggelassen werden.

```
Beispiel: $$ SET text = "Dies ist ein Beispiel"
          $$ SET apos = FIND_STRING (text, "ist", 0, 0, 1, epos)
```

```
Ergebnis: apos = 6
           epos = 9
```

Falls die Variable `var` eine Sternvariable ist, erfolgt das oben beschriebene Suchen für jede Zeile einzeln; die sich ergebenden Werte werden durch Apostroph getrennt als Funktionswert geliefert bzw. in der Variablen `npos` gespeichert.

```
SEARCH (var, stab, apos, epos, n, npos)
```

Die Makrofunktion `SEARCH` sucht im Inhalt der Variablen `var` von der Position `apos` (einschließlich) bis zur Position `epos` (ausschließlich) nach der `n`-ten bzw. letzten Zeichenfolge, die mit einer in der Such-Tabelle `stab` angegebenen Suchzeichenfolge übereinstimmt.

Für das Argument `stab` kann der Name einer mit der `BUILD`-Anweisung definierten Such-Tabelle oder direkt eine in Anführungszeichen eingeschlossene Such-Tabelle angegeben werden.

Soll der gesamte Inhalt der Variablen `var` durchsucht werden, so kann für die Argumente `apos` und `epos` jeweils 0 (Null) angegeben werden; andernfalls kann mit diesen beiden Argumenten in gleicher Weise wie bei der Makrofunktion `EXTRACT` (siehe Seite 519) ein Bereich ausgewählt werden.

Mit dem Argument `n` wird festgelegt, die wievielte Zeichenfolge gesucht wird; hat das Argument `n` den Wert 0 (Null), so wird die letzte gesucht. Für das Argument `n` kann eine Zahl oder eine Variable, die eine Zahl enthält, angegeben werden.

Der Funktionswert gibt die Position des ersten Zeichens der gefundenen Zeichenfolge an und der Variablen `npos` wird die Position des Zeichens zugewiesen, das nach der gefundenen Zeichenfolge steht; falls keine Zeichenfolge bzw. weniger als `n` Zeichenfolgen gefunden wurden, ist der Funktionswert und der Inhalt der Variablen `npos` 0 (Null).

Falls die Position des Zeichens nach der gefundenen Zeichenfolge nicht benötigt wird, kann das Argument `npos` einschließlich des davor stehenden Kommas weggelassen werden; falls außerdem die erste Zeichenfolge gesucht werden soll, kann auch das Argument `n` einschließlich des davor stehenden Kommas weggelassen werden; falls zusätzlich der gesamte Inhalt der Variablen `var` durchsucht werden soll, können auch noch die beiden Argumente `apos` und `epos` einschließlich des jeweils davor stehenden Kommas weggelassen werden.

```
Beispiel: $$ BUILD S_TABLE blanks = ": :"
```

```
          $$ SET text = "Dies ist ein Beispiel"
```

```
          $$ SET pos = SEARCH (text, blanks, 0, 0, 2)
```



```
Ergebnis: pos = 9
```

Falls die Variable `var` eine Sternvariable ist, erfolgt das oben beschriebene Suchen für jede Zeile einzeln; die sich ergebenden Werte werden durch Apostroph getrennt als Funktionswert geliefert bzw. in der Variablen `npos` gespeichert.

```
SEARCH_ALL (var, stab, apos, epos, npos)
```

Die Makrofunktion `SEARCH_ALL` sucht im Inhalt der Variablen `var` von der Position `apos` (einschließlich) bis zur Position `epos` (ausschließlich) nach allen Zeichenfolgen, die mit einer in der Such-Tabelle `stab` angegebenen Suchzeichenfolge übereinstimmen.

Für das Argument `stab` kann der Name einer mit der `BUILD`-Anweisung definierten Such-Tabelle oder direkt eine in Anführungszeichen eingeschlossene Such-Tabelle angegeben werden.

Soll der gesamte Inhalt der Variablen `var` durchsucht werden, so kann für die Argumente `apos` und `epos` jeweils 0 (Null) angegeben werden; andernfalls kann mit diesen beiden Argumenten in gleicher Weise wie bei der Makrofunktion `EXTRACT` (siehe Seite 519) ein Bereich ausgewählt werden.

Der Funktionswert gibt die Positionen des jeweils ersten Zeichens der gefundenen Zeichenfolgen an und der Variablen `npos` wird jeweils die Position des Zeichens zugewiesen, das nach der gefundenen Zeichenfolge steht; falls mehrere Zeichenfolgen gefunden wurden, werden die einzelnen Positionen durch Apostroph getrennt; falls keine Zeichenfolge gefunden wurden, ist der Funktionswert und der Inhalt der Variablen `npos` eine leere Zeichenfolge.

Falls die Positionen der Zeichen nach den gefundenen Zeichenfolgen nicht benötigt werden, kann das Argument `npos` einschließlich des davor stehenden Kommas weggelassen werden; falls zusätzlich der gesamte Inhalt der Variablen `var` durchsucht werden soll, können auch noch die beiden Argumente `apos` und `epos` einschließlich des jeweils davor stehenden Kommas weggelassen werden.

```
Beispiel: $$ SET text = "Dies ist ein Beispiel"
          $$ SET pos = SEARCH_ALL (text, ": :")
```

```
Ergebnis: pos = 5'9'13
```

Falls die Variable `var` eine Sternvariable ist, erfolgt das oben beschriebene Suchen für jede Zeile einzeln; die sich ergebenden Werte werden zeilenweise zum Speichern in einer Sternvariablen als Funktionswert geliefert bzw. in der Sternvariablen `npos` gespeichert.

```
COUNT (var, stab, apos, epos)
```

Die Makrofunktion `COUNT` zählt im Inhalt der Variablen `var` von der Position `apos` (einschließlich) bis zur Position `epos` (ausschließlich) die Zeichenfolgen, die mit einer in der Such-Tabelle `stab` angegebenen Suchzeichenfolge übereinstimmen, und liefert die Anzahl als Funktionswert.

Für das Argument `stab` kann der Name einer mit der `BUILD`-Anweisung definier-

ten Such-Tabelle oder direkt eine in Anführungszeichen eingeschlossene Such-Tabelle angegeben werden.

Sollen die Zeichenfolgen im gesamten Inhalt der Variablen `var` gezählt werden, so kann für die Argumente `apos` und `epos` jeweils 0 (Null) angegeben werden, oder es können beide Argumente einschließlich des jeweils davor stehenden Kommas weggelassen werden; andernfalls kann mit diesen beiden Argumenten in gleicher Weise wie bei der Makrofunktion `EXTRACT` (siehe Seite 519) ein Bereich ausgewählt werden.

```
Beispiel: $$ BUILD S_TABLE spatien = ": :"  
          $$ SET text = "Dies ist ein Beispiel"  
          $$ SET anzahl = COUNT (text, spatien)
```

```
Ergebnis: anzahl = 3
```

Falls die Variable `var` eine Sternvariable ist, erfolgt das oben beschriebene Zählen für jede Zeile einzeln; die sich ergebenden Werte werden durch Apostroph getrennt als Funktionswert geliefert.

```
FIND_DIFF (var1, apos1, epos1, var2, apos2, epos2, char1, char2)
```

Die Makrofunktion `FIND_DIFF` vergleicht die Zeichenfolge, die in der Variablen `var1` von der Position `apos1` (einschließlich) bis zur Position `epos1` (ausschließlich) steht, mit der Zeichenfolge, die in der Variablen `var2` von der Position `apos2` (einschließlich) bis zur Position `epos2` (ausschließlich) steht. Bei der Prüfung auf Übereinstimmung werden Groß- und Kleinbuchstaben unterschieden.

Soll der gesamte Inhalt der Variablen `var1` bzw. `var2` verglichen werden, so kann für die Argumente `apos1` und `epos1` bzw. `apos2` und `epos2` jeweils 0 (Null) angegeben werden; andernfalls kann mit diesen beiden Argumenten jeweils in gleicher Weise wie bei der Makrofunktion `EXTRACT` (siehe Seite 519) ein Bereich ausgewählt werden.

Falls die beiden zu vergleichenden Zeichenfolgen gleich sind, ist der Funktionswert 0 (Null) und den Variablen `char1` und `char2` wird eine leere Zeichenfolge zugewiesen; andernfalls gibt der Funktionswert an, das wievielte verglichene Zeichen sich als erstes unterscheidet; den beiden Variablen `char1` und `char2` wird das Zeichen zugewiesen, das sich als erstes in der ersten bzw. zweiten Zeichenfolge unterscheidet. Werden diese beiden Zeichen nicht benötigt, können die beiden Variablen `char1` und `char2` jeweils einschließlich des davor stehenden Kommas weggelassen werden.

```
Beispiel: $$ SET a = "Das ist ein Beispiel"  
          $$ SET b = "Dies ist ein Beispiel"  
          $$ SET pos = FIND_DIFF (a, 0, 0, b, 0, 0, z1, z2)
```

```
Ergebnis: pos = 2  
          z1 = "a"  
          z2 = "i"
```

Falls die Variablen `var1` und `var2` Sternvariablen sind, erfolgt das Vergleichen

zeilenweise bis ein Unterschied gefunden wird. Nach dem Argument `char2` kann durch ein Komma getrennt noch zusätzlich eine Variable `nummer` angegeben werden; dieser Variablen wird die laufende Nummer der Zeile zugewiesen, in der ein Unterschied gefunden wurde, bzw. der Wert 0 (Null), falls kein Unterschied gefunden wurde.

`DISTANCE (var1, var2)`

Die Makrofunktion `DISTANCE` vergleicht die Zeichenfolge, die in der Variablen `var1` steht, mit der Zeichenfolge, die in der Variablen `var2` steht, und berechnet die Levenshtein-Distanz (auch Editierdistanz genannt). Sie entspricht der minimalen Anzahl der Einfüge-, Lösch- und Ersetzungs-Operationen, die erforderlich sind, um aus der Zeichenfolge in der Variablen `var1` die Zeichenfolge in der Variablen `var2` zu erhalten.

Bei der Prüfung auf Übereinstimmung werden Groß- und Kleinbuchstaben nicht unterschieden. Sollen sie unterschieden werden, muss die Makrofunktion `DISTANCE_EXACT` verwendet werden.

```
Beispiel: $$ SET texta = "Dies ist ein kleines Beispiel"
          $$ SET textb = "Dies ist ein kurzes Beispiel"
          $$ SET dist = DISTANCE (texta, textb)
```

```
Ergebnis: dist = 4
```

`DISTANCE_EXACT (var1, var2)`

Die Makrofunktion `DISTANCE_EXACT` unterscheidet sich von der zuvor beschriebenen Makrofunktion `DISTANCE` dadurch, dass Groß- und Kleinbuchstaben als verschiedenwertig gelten.

`VERIFY (var, stab, apos, epos)`

Die Makrofunktion `VERIFY` überprüft, ob die Zeichenfolge, die in der Variablen `var` von der Position `apos` (einschließlich) bis zur Position `epos` (ausschließlich) steht, nur aus solchen Zeichenfolgen besteht, die in der Such-Tabelle `stab` als Suchzeichenfolgen angegeben sind.

Für das Argument `stab` kann der Name einer mit der `BUILD`-Anweisung definierten Such-Tabelle oder direkt eine in Anführungszeichen eingeschlossene Such-Tabelle angegeben werden.

Soll der gesamte Inhalt der Variablen `var` überprüft werden, so kann für die Argumente `apos` und `epos` jeweils 0 (Null) angegeben werden, oder es können beide Argumente einschließlich des jeweils davor stehenden Kommas weggelassen werden; andernfalls kann mit diesen beiden Argumenten in gleicher Weise wie bei der Makrofunktion `EXTRACT` (siehe Seite 519) ein Bereich ausgewählt werden.

Falls ein Zeichen in der Variablen `var` gefunden wird, das keiner Suchzeichenfolge (d. h. nur einer Ausnahmezeichenfolge oder gar keiner Zeichenfolge) aus der Such-Tabelle zugeordnet werden kann, gibt der Funktionswert die Position dieses Zeichens an; andernfalls ist der Funktionswert 0 (Null).

```
Beispiel: $$ BUILD S_TABLE test = ":{&a}:%{%}{&a}:{&a}{} :"  
          $$ SET text = "Dies ist ein Beispiel"  
          $$ SET pos = VERIFY (text, test)
```

```
Ergebnis: pos = 10
```

Falls die Variable `var` eine Sternvariable ist, erfolgt das oben beschriebene Überprüfen für jede Zeile einzeln; die sich ergebenden Werte werden durch Apostroph getrennt als Funktionswert geliefert.

```
TURN (var, stab, apos, epos, xtab)
```

Diese Makrofunktion `TURN` nimmt den Inhalt der Variablen `var`, dreht die Zeichen um und liefert das Ergebnis als Funktionswert; falls die Variable `var` eine Sternvariable ist, erfolgt das Umdrehen zeilenweise.

Für das Argument `stab` kann ein Minuszeichen, der Name einer mit der `BUILD`-Anweisung definierten Such-Tabelle oder direkt eine in Anführungszeichen eingeschlossene Such-Tabelle angegeben werden. Wird eine Tabelle angegeben, so werden die Zeichenfolgen, die mit einer Suchzeichenfolge übereinstimmen, nicht umgedreht.

Sollen die Zeichenfolgen im gesamten Inhalt der Variablen `var` umgedreht werden, so kann für die Argumente `apos` und `epos` jeweils 0 (Null) angegeben werden; andernfalls kann mit diesen beiden Argumenten in gleicher Weise wie bei der Makrofunktion `EXTRACT` (siehe Seite 519) ein Bereich ausgewählt werden.

Für das Argument `xtab` kann ein Minuszeichen, der Name einer mit der `BUILD`-Anweisung definierten Austausch-Tabelle oder direkt eine in Anführungszeichen eingeschlossene Austausch-Tabelle angegeben werden. Wird eine Tabelle angegeben, so werden innerhalb der nicht umzudrehenden Zeichenfolgen, die mit dem Argument `stab` angegeben sind, die in der Austausch-Tabelle `xtab` enthaltenen Suchzeichenfolgen mit den dazugehörigen Ersatzzeichenfolgen ausgetauscht. Falls für das Argument `stab` keine Tabelle angegeben wird, darf auch für das Argument `xtab` keine angegeben werden.

Falls keine Austausch-Tabelle benötigt wird, kann das Argument `xtab` einschließlich des davor stehenden Kommas weggelassen werden; falls außerdem der gesamte Inhalt der Variablen `var` umgedreht werden soll, können auch die beiden Argumente `apos` und `epos` einschließlich des jeweils davor stehenden Kommas weggelassen werden; falls zusätzlich keine Such-Tabelle benötigt wird, kann auch noch das Argument `stab` einschließlich des davor stehenden Kommas weggelassen werden.

```
Beispiel: $$ alt = "0123456789"  
          $$ SET neu = TURN (alt)
```

```
Ergebnis: neu = "9876543210"
```

Hinweis: Mit der Makrofunktion `REVERSE` (siehe Seite 553) kann die Reihenfolge der einzelnen Teilzeichenfolgen (bei Sternvariablen die Reihenfolge der einzelnen Zeilen) umgedreht werden.

TURN (*var*, *stab*, *apos*, *epos*, *xtab*, *num*, *anz*)

Die erweiterte Makrofunktion TURN bestimmt im Inhalt der Variablen *var* einzelne Bereiche, dreht die Zeichen innerhalb dieser Bereiche um und liefert das Ergebnis als Funktionswert; falls die Variable *var* eine Sternvariable ist, werden die Bereiche zeilenweise bestimmt.

Die Bereiche werden in gleicher Weise wie bei der erweiterten Makrofunktion EXTRACT (siehe Seite 521) mit den Argumenten *apos* und *epos* sowie *num* und *anz* ausgewählt. Soll nur in einem einzigen Bereich ausgetauscht werden, so kann das Argument *anz* einschließlich des davor stehenden Kommas weggelassen werden.

## Makrofunktionen für Teilzeichenfolgen/Zeilen

SIZE (*var*)

Die Makrofunktion SIZE liefert als Funktionswert die Anzahl der Teilzeichenfolgen, die die Variable *var* enthält; falls *var* eine Sternvariable ist, die Anzahl der Zeilen, die diese Variable enthält. Wenn die Variable eine leere Zeichenfolge bzw. keine Zeilen enthält, ist der Funktionswert 0 (Null).

Beispiel: 

```
$$ SET farben = "blau'grün'rot"
$$ SET anzahl = SIZE (farben)
```

Ergebnis: `anzahl = 3`

ADJUST\_SIZE (*erg*, *var1*, *var2*, ...)

Die Makrofunktion ADJUST\_SIZE ergänzt in den Variablen *var1*, *var2* usw. am Ende jeweils soviele Teilzeichenfolgen (bei »normalen« Variablen) bzw. Zeilen (bei Sternvariablen), dass danach alle Variablen soviele Teilzeichenfolgen bzw. Zeilen enthalten, wie zuvor die Variable mit den meisten Teilzeichenfolgen bzw. Zeilen hatte; leere Teilzeichenfolgen bzw. leere Zeilen am Ende der Variablen bleiben dabei unberücksichtigt. Die Anzahl der Teilzeichenfolgen bzw. Zeilen je Variable wird als Funktionswert geliefert.

Was ergänzt wird, kann mit der Variablen *erg* vorgegeben werden. Falls *erg* eine »normale« Variable ist, wird jeweils ihr Inhalt als Teilzeichenfolge bzw. als Zeile ergänzt; falls *erg* eine Sternvariable ist, wird die Variable *var1* mit dem Inhalt der 1. Zeile ergänzt, die Variable *var2* mit dem Inhalt der 2. Zeile usw. Falls die Sternvariable *erg* weniger Zeilen enthält als nach ihr Variablen angegeben sind, werden bei den restlichen Variablen leere Zeichenfolgen bzw. Leerzeilen ergänzt.

An Stelle der Variablen *erg* kann auch eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben werden.

Beispiel: `$$ SET alist = *`  
`eins`

`$$ SET blist = *`  
`uno`  
`due`  
`tre`

`$$ SET clist = *`  
`one`  
`two`

`$$ SET n = ADJUST_SIZE ("", alist, blist, clist)`

Ergebnis: `n = 3`  

<code>alist = *</code>	<code>blist = *</code>	<code>clist = *</code>
<code>eins</code>	<code>uno</code>	<code>one</code>
<code>(leer)</code>	<code>due</code>	<code>two</code>
<code>(leer)</code>	<code>tre</code>	<code>(leer)</code>

`MIN_LENGTH (var)`

Die Makrofunktion `MIN_LENGTH` liefert als Funktionswert die Anzahl der Zeichen der kürzesten Teilzeichenfolge in der Variablen `var`; falls `var` eine Sternvariable ist, die Anzahl der Zeichen der kürzesten Zeile, die diese Variable enthält.

Beispiel: `$$ SET text = "rot'gelb"`  
`$$ SET anzahl = MIN_LENGTH (text)`

Ergebnis: `anzahl = 3`

`MAX_LENGTH (var)`

Die Makrofunktion `MAX_LENGTH` liefert als Funktionswert die Anzahl der Zeichen der längsten Teilzeichenfolge in der Variablen `var`; falls `var` eine Sternvariable ist, die Anzahl der Zeichen der längsten Zeile, die diese Variable enthält.

Beispiel: `$$ SET text = "rot'gelb"`  
`$$ SET anzahl = MAX_LENGTH (text)`

Ergebnis: `anzahl = 4`

`ADJUST_LENGTH (xtab, var1, var2, ...)`

Die Makrofunktion `ADJUST_LENGTH` ergänzt in den einzelnen Zeilen der Sternvariablen `var1`, `var2` usw. am Zeilenende jeweils so viele Leerzeichen, dass danach alle Zeilen in allen Variablen so viele Zeichen enthalten wie zuvor die längste Zeile in den angegebenen Variablen hatte; diese Anzahl der Zeichen wird als Funktionswert geliefert.

Bevor die Länge der einzelnen Zeilen festgestellt wird, werden intern die in der Austausch-Tabelle `xtab` angegebenen Suchzeichenfolgen durch die dazugehören-

den Ersatzzeichenfolgen ausgetauscht. Damit ist es z. B. möglich, Akzent-Codierungen bei der Längenberechnung unberücksichtigt zu lassen. Der Inhalt der angegebenen Variablen wird durch dieses Austauschen nicht verändert. Falls ein Austauschen nicht erforderlich ist, kann für das Argument `xtab` ein Minuszeichen angegeben werden.

`APPEND (var1, var2)`

Diese Makrofunktion `APPEND` (eine weitere ist auf Seite 513 beschrieben) nimmt den Inhalt der Variablen `var1`, fügt den Inhalt der Variablen `var2` als neue Teilzeichenfolge hinten an und liefert das Ergebnis als Funktionswert. Falls die Variable `var1` eine Sternvariable ist, wird der Inhalt der Variablen `var2` als neue Zeile hinten angefügt. Falls die Variable `var2` eine Sternvariable ist, wird der Inhalt der Variablen `var1` als neue Zeile vorne eingefügt. Falls beide Variablen Sternvariablen sind, werden die Zeilen dieser Variablen zusammengefügt.

An Stelle der Variablen `var1` und `var2` können auch in Anführungszeichen eingeschlossene Zeichenfolgen angegeben werden.

Beispiel: `$$ SET alle = "", erg = "blau"`  
`$$ SET alle = APPEND (alle, erg)`

Ergebnis: `alle = "blau"`

`$$ SET erg = "gelb"`  
`$$ SET alle = APPEND (alle, erg)`

Ergebnis: `alle = "blau'gelb"`

`COLLECT (var1, var2)`

Die Makrofunktion `COLLECT` prüft für jede in der Variablen `var2` enthaltene Teilzeichenfolge, ob sie schon als Teilzeichenfolge in der Variablen `var1` enthalten ist. Wenn alle schon enthalten sind, liefert die Makrofunktion den Inhalt der Variablen `var1` als Funktionswert; wenn nicht, wird der Inhalt der Variablen `var1` genommen, die noch nicht enthaltenen Teilzeichenfolgen als neue Teilzeichenfolgen hinten angefügt und das Ergebnis als Funktionswert geliefert.

Bei der Prüfung auf Übereinstimmung werden Groß- und Kleinbuchstaben nicht unterschieden. Sollen sie unterschieden werden, muss die Makrofunktion `COLLECT_EXACT` verwendet werden.

Falls die Variable `var1` eine Sternvariable ist, wird geprüft, ob der Inhalt der Variablen `var2` schon als Zeile in der Variablen `var1` enthalten ist. Wenn ja, liefert die Makrofunktion den Inhalt der Variablen `var1` als Funktionswert; wenn nicht, wird der Inhalt der Variablen `var1` genommen, der Inhalt der Variablen `var2` als neue Zeile hinten angefügt und das Ergebnis als Funktionswert geliefert.

Falls die Variable `var2` eine leere Zeichenfolge enthält, liefert die Makrofunktion in jedem Fall den unveränderten Inhalt der Variablen `var1` als Funktionswert.

An Stelle der Variablen `var2` kann auch eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben werden.

Beispiel: `$$ SET alle = "blau'rot", erg = "gelb"`  
`$$ SET alle = COLLECT (alle, erg)`

Ergebnis: `alle = "blau'rot'gelb"`

`$$ SET erg = "blau"`  
`$$ SET alle = COLLECT (alle, erg)`

Ergebnis: `alle = "blau'rot'gelb"`

Hinweis: Die Makrofunktion `COLLECT` sucht den Inhalt der Variablen `var1` sequentiell ab. Bei umfangreichen Daten sollte deshalb ein Wörterbuch (siehe Seite 445), bei dem der Inhalt (zeitsparend) binär abgesucht wird, verwendet werden.

`COLLECT_EXACT (var1, var2)`

Die Makrofunktion `COLLECT_EXACT` unterscheidet sich von der zuvor beschriebenen Makrofunktion `COLLECT` dadurch, dass bei der Prüfung auf Übereinstimmung Groß- und Kleinbuchstaben unterschieden werden.

`INDEX (liste, var)`

Die Makrofunktion `INDEX` vergleicht den Inhalt der Variablen `var` mit dem Inhalt der einzelnen Teilzeichenfolgen der Variablen `liste`. Wird eine Übereinstimmung gefunden, gibt der Funktionswert die Nummer der Teilzeichenfolge in der Variablen `liste` an, die mit dem Inhalt der Variablen `var` übereinstimmt; wird keine Übereinstimmung gefunden, so ist der Funktionswert 0 (Null).

Bei der Prüfung auf Übereinstimmung werden Groß- und Kleinbuchstaben nicht unterschieden. Sollen sie unterschieden werden, muss die Makrofunktion `INDEX_EXACT` verwendet werden.

Beispiel: `$$ SET farben = "blau'gelb'rot", farbe = "gelb"`  
`$$ SET nummer = INDEX (farben, farbe)`

Ergebnis: `nummer = 2`

Falls die Variable `liste` eine Sternvariable ist, wird der Inhalt der Variablen `var` mit den einzelnen Zeilen der Variablen `liste` verglichen. Der Funktionswert gibt dann die Nummer der Zeile an.

Hinweis: Die Makrofunktion `INDEX` sucht den Inhalt der Variablen `liste` sequentiell ab. Bei umfangreichen Daten sollte deshalb ein Wörterbuch (siehe Seite 445), bei dem der Inhalt (zeitsparend) binär abgesucht wird, verwendet werden.

`INDEX_EXACT (liste, var)`

Die Makrofunktion `INDEX_EXACT` unterscheidet sich von der zuvor beschriebenen Makrofunktion `INDEX` dadurch, dass bei der Prüfung auf Übereinstimmung Groß- und Kleinbuchstaben unterschieden werden.



```
INSERT (liste, index, var)
```

Die Makrofunktion `INSERT` nimmt den Inhalt der Variablen `liste` und fügt den Inhalt der Variablen `var` vor der Teilzeichenfolge mit der Nummer `index` ein; falls `liste` eine Sternvariable ist, wird der Inhalt der Variablen `var` als neue Zeile vor die Zeile mit der Nummer `index` eingefügt. Das Ergebnis wird als Funktionswert geliefert.

Für das Argument `index` kann eine Zahl oder eine Variable angegeben werden; wird eine Variable angegeben, so muss diese mit einem Nummernzeichen unmittelbar vor dem Namen gekennzeichnet sein und eine Zahl enthalten.

```
Beispiel: $$ SET farben1 = "blau'gelb'rot", farbe = "grün"
          $$ SET ind = 2
          $$ SET farben2 = INSERT (farben1, #ind, farbe)
```

```
Ergebnis: farben2 = "blau'grün'gelb'rot"
```

Wird für das Argument `index` 0 (Null) oder eine Variable mit dem Wert 0 angegeben, so wird der Inhalt der Variablen `var` als neue Teilzeichenfolge bzw. als neue Zeile vor die erste eingefügt, die bei alphabetischer Ordnung nach ihr folgt. Dabei wird die Standard-Sortierfolge (siehe Seite 841) zugrunde gelegt. Für Teilzeichenfolgen bzw. Zeilen, die sich auf Grund der Standard-Sortierfolge nicht unterscheiden, wird zusätzlich die Sonder-Sortierfolge (siehe Seite 841) zugrunde gelegt.

```
Beispiel: $$ SET farben1 = "blau'gelb'rot", farbe = "grün"
          $$ SET farben2 = INSERT (farben1, 0, farbe)
```

```
Ergebnis: farben2 = "blau'gelb'grün'rot"
```

Hinweis: Soll der Inhalt der Variablen `var` hinten eingefügt werden, so kann dafür auch die Makrofunktion `APPEND` (siehe Seite 543) verwendet werden.

```
INSERT_INDEX (liste, var)
```

Die Makrofunktion `INSERT_INDEX` nimmt den Inhalt der Variablen `liste` und prüft, als wievielte Teilzeichenfolge der Inhalt der Variablen `var` (mit der Makrofunktion `INSERT`) bei alphabetischer Ordnung eingefügt würde; falls `liste` eine Sternvariable ist, wird geprüft, als wievielte Zeile der Inhalt der Variablen `var` eingefügt würde. Das Ergebnis wird als Funktionswert geliefert.

Beim Prüfen wird die Standard-Sortierfolge (siehe Seite 841) zugrunde gelegt. Für Teilzeichenfolgen bzw. Zeilen, die sich auf Grund der Standard-Sortierfolge nicht unterscheiden, wird zusätzlich die Sonder-Sortierfolge (siehe Seite 841) zugrunde gelegt.

```

Beispiel: $$ SET nliste = *
          Maier, Josef
          Schmid, Paula

          $$ SET oliste = *
          Trier
          Tübingen

          $$ SET name = "Müller, Peter", ort = "München"
          $$ SET nummer = INSERT_INDEX (nliste, name)
          $$ SET nliste = INSERT (nliste, #nummer, name)
          $$ SET oliste = INSERT (oliste, #nummer, ort)

```

```

Ergebnis: nummer = 2
           nliste = *           oliste = *
           Maier, Josef       Trier
           Müller, Peter     München
           Schmid, Paula     Tübingen

```

REPLACE (liste, index, var)

Die Makrofunktion REPLACE nimmt den Inhalt der Variablen `liste` und ersetzt die Teilzeichenfolge mit der Nummer `index` durch den Inhalt der Variablen `var`; falls `liste` eine Sternvariable ist, wird die Zeile mit der laufenden Nummer `index` ersetzt. Das Ergebnis wird als Funktionswert geliefert.

Für das Argument `index` kann eine Zahl oder eine Variable angegeben werden; wird eine Variable angegeben, so muss diese mit einem Nummernzeichen unmittelbar vor dem Namen gekennzeichnet sein und eine Zahl enthalten.

```

Beispiel: $$ SET farben1 = "blau'gelb'rot", farbe = "grün"
          $$ SET ind = 2
          $$ SET farben2 = REPLACE (farben1, #ind, farbe)

```

```

Ergebnis: farben2 = "blau'grün'rot"

```

REMOVE (liste, auswahl, raus)

Die Makrofunktion REMOVE nimmt den Inhalt der Variablen `liste` und entfernt Teilzeichenfolgen; falls `liste` eine Sternvariable ist, werden Zeilen entfernt. Das Ergebnis wird als Funktionswert geliefert.

Welche Teilzeichenfolgen bzw. Zeilen entfernt werden, wird durch das Argument `auswahl` bestimmt. Für das Argument `auswahl` kann eine Zahl, eine Variable oder eine Such-Tabelle angegeben werden; wird eine Variable angegeben, so muss diese mit einem Nummernzeichen unmittelbar vor dem Namen gekennzeichnet sein und eine Zahl, einen Zahlenbereich (= zwei durch »-« verbundene Zahlen) oder mehrere durch Apostroph getrennte Zahlen und/oder Zahlenbereiche enthalten; als Such-Tabelle kann der Name einer mit der BUILD-Anweisung definierten Such-Tabelle oder direkt eine in Anführungszeichen eingeschlossene Such-Tabelle angegeben werden.

Die entfernten Teilzeichenfolgen bzw. Zeilen werden der Variablen `raus` zugewiesen. Werden die entfernten Teilzeichenfolgen bzw. Zeilen nicht benötigt, kann die Variable `raus` einschließlich des davor stehenden Kommas weggelassen werden.

Wird für das Argument `auswahl` eine Zahl oder eine Variable, die Zahlen enthält, angegeben, so werden die Teilzeichenfolgen bzw. die Zeilen mit den entsprechenden laufenden Nummern entfernt.

```
Beispiel: $$ SET farben1 = "blau'gelb'grün'rot"
          $$ SET ind = "2-3"
          $$ SET farben2 = REMOVE (farben1, #ind)

          Ergebnis: farben2 = "blau'rot"
```

Wird für das Argument `auswahl` eine Such-Tabelle angegeben, so werden alle Teilzeichenfolgen bzw. Zeilen entfernt, die mindestens eine der in der Such-Tabelle angegebenen Zeichenfolgen enthalten.

```
Beispiel: $$ BUILD S_TABLE tab = ":b:"
          $$ SET farben1 = "blau'gelb'grün'rot"
          $$ SET farben2 = REMOVE (farben1, tab, farben3)

          Ergebnis: farben2 = "grün'rot"
                   farben3 = "blau'gelb"
```

Hinweis: Mit der Makrofunktion `REDUCE` (siehe Seite 550) können mehrfach vorkommende Teilzeichenfolgen/Zeilen entfernt werden, mit der Makrofunktion `SEPARATE` (siehe Seite 548) können Teilzeichenfolgen/Zeilen, die mit einer Teilzeichenfolge/Zeile in einer anderen Variablen übereinstimmen, entfernt werden.

```
SELECT (liste, auswahl, rest)
```

Die Makrofunktion `SELECT` nimmt den Inhalt der Variablen `liste` und wählt aus ihm Teilzeichenfolgen aus; falls `liste` eine Sternvariable ist, werden Zeilen ausgewählt. Das Ergebnis wird als Funktionswert geliefert.

Welche Teilzeichenfolgen bzw. Zeilen ausgewählt werden, wird durch das Argument `auswahl` bestimmt. Für das Argument `auswahl` kann eine Zahl, eine Variable oder eine Such-Tabelle angegeben werden; wird eine Variable angegeben, so muss diese mit einem Nummernzeichen unmittelbar vor dem Namen gekennzeichnet sein und eine Zahl, einen Zahlenbereich (= zwei durch »-« verbundene Zahlen) oder mehrere durch Apostroph getrennte Zahlen und/oder Zahlenbereiche enthalten; als Such-Tabelle kann der Name einer mit der `BUILD`-Anweisung definierten Such-Tabelle oder direkt eine in Anführungszeichen eingeschlossene Such-Tabelle angegeben werden.

Die nicht ausgewählten Teilzeichenfolgen bzw. Zeilen werden der Variablen `rest` zugewiesen. Werden die nicht ausgewählten Teilzeichenfolgen bzw. Zeilen nicht benötigt, kann die Variable `rest` einschließlich des davor stehenden Kommas weggelassen werden.

Wird für das Argument `auswahl` eine Zahl oder eine Variable, die Zahlen enthält, angegeben, so werden die Teilzeichenfolgen bzw. die Zeilen mit den entsprechen-

den laufenden Nummern ausgewählt. Die Reihenfolge der Nummern in der Variablen ist bedeutungslos; im Ergebnis stehen die ausgewählten Teilzeichenfolgen bzw. Zeilen immer in der Reihenfolge, in der sie in der Variablen `liste` stehen. Werden Nummern mehrfach angegeben, so werden die entsprechende Teilzeichenfolgen bzw. Zeilen trotzdem jeweils nur einmal ins Ergebnis übernommen.

```
Beispiel: $$ SET farben1 = "blau'gelb'grün'rot"
          $$ SET ind = "2-3"
          $$ SET farben2 = SELECT (farben1, #ind)

          Ergebnis: farben2 = "gelb'grün"
```

Wird für das Argument `auswahl` eine Such-Tabelle angegeben, so werden alle Teilzeichenfolgen bzw. Zeilen ausgewählt, die mindestens eine der in der Such-Tabelle angegebenen Zeichenfolgen enthalten.

```
Beispiel: $$ BUILD S_TABLE tab = ":b:"
          $$ SET farben1 = "blau'gelb'grün'rot"
          $$ SET farben2 = SELECT (farben1, tab)

          Ergebnis: farben2 = "blau'gelb"
```

Hinweis: Mit der Makrofunktion `SEPARATE` (siehe Seite 548) können Teilzeichenfolgen/Zeilen, die mit einer Teilzeichenfolge/Zeile in einer anderen Variablen übereinstimmen, ausgewählt werden.

```
SEPARATE (liste, auswahl, raus)
```

Die Makrofunktion `SEPARATE` nimmt den Inhalt der Variablen `liste`, entfernt die Teilzeichenfolgen, die mit einer Teilzeichenfolge in der Variablen `auswahl` übereinstimmen, und liefert das Ergebnis als Funktionswert.

Bei der Prüfung auf Übereinstimmung werden Groß- und Kleinbuchstaben nicht unterschieden. Sollen sie unterschieden werden, muss die Makrofunktion `SEPARATE_EXACT` verwendet werden.

An Stelle der Variablen `auswahl` kann auch eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben werden.

Falls die Variablen `liste` und/oder `auswahl` Sternvariablen sind, werden die Zeilen in gleicher Weise wie andernfalls die Teilzeichenfolgen behandelt.

Die entfernten Teilzeichenfolgen bzw. Zeilen werden der Variablen `raus` zugewiesen. Werden die entfernten Teilzeichenfolgen bzw. Zeilen nicht benötigt, kann die Variable `raus` einschließlich des davor stehenden Kommas weggelassen werden.

```
Beispiel: $$ SET alt = "eins'und'zwei"
          $$ SET hlf = "und'oder"
          $$ SET neu = SEPARATE (alt, hlf, raus)

          Ergebnis: neu = "eins'zwei", raus = "und"
```

SEPARATE\_EXACT (liste, var)

Die Makrofunktion `SEPARATE_EXACT` unterscheidet sich von der zuvor beschriebenen Makrofunktion `SEPARATE` dadurch, dass bei der Prüfung auf Übereinstimmung Groß- und Kleinbuchstaben unterschieden werden.

FILTER (liste, positiv, negativ, raus)

Die Makrofunktion `FILTER` nimmt den Inhalt der Variablen `liste` und wählt aus ihm Teilzeichenfolgen aus; falls `liste` eine Sternvariable ist, werden Zeilen ausgewählt. Das Ergebnis wird als Funktionswert geliefert.

Welche Teilzeichenfolgen bzw. Zeilen ausgewählt werden, wird durch die Argumente `positiv` und `negativ` bestimmt. Für die Argumente `positiv` und `negativ` muss jeweils der Name einer mit der `BUILD`-Anweisung definierten Recherchier-Tabelle, eine in Anführungszeichen eingeschlossene Recherchier-Tabelle oder ein Minuszeichen angegeben werden.

Wird eine in Anführungszeichen eingeschlossene Recherchier-Tabelle angegeben, so werden automatisch die Optionen `TEXT` und `OR` angenommen. Falls andere Optionen (siehe Seite 441) erforderlich sind, muss die Tabelle mit der `BUILD`-Anweisung definiert und der Name der Tabelle angegeben werden.

Wird für das Argument `positiv` eine Recherchier-Tabelle angegeben, so werden nur solche Teilzeichenfolgen bzw. Zeilen ausgewählt, die mindestens eine (Tabelle mit Option `OR`) bzw. alle (Tabelle mit Option `AND`) in der Tabelle angegebenen Zeichenfolgen enthalten.

Wird für das Argument `negativ` eine Recherchier-Tabelle angegeben, so werden nur solche Teilzeichenfolgen bzw. Zeilen ausgewählt, die keine (Tabelle mit Option `OR`) bzw. nicht alle (Tabelle mit Option `AND`) in der Tabelle angegebenen Zeichenfolgen enthalten.

Hinweis: Die Option muss beim Definieren der Recherchier-Tabelle mit der `BUILD`-Anweisung angegeben werden.

Die nicht ausgewählten Teilzeichenfolgen bzw. Zeilen werden der Variablen `raus` zugewiesen. Werden diese Teilzeichenfolgen bzw. Zeilen nicht benötigt, kann die Variable `raus` einschließlich des davor stehenden Kommas weggelassen werden.

```
Beispiel: $$ BUILD R_TABLE/OR tab1 = ":b:"
          $$ BUILD R_TABLE/OR tab2 = ":g:"
          $$ SET farben1 = "blau'gelb'grün'rot"
          $$ SET farben2 = FILTER (farben1, tab1, -)
          $$ SET farben3 = FILTER (farben1, -, tab2)
          $$ SET farben4 = FILTER (farben1, tab1, tab2)
```

```
Ergebnis: farben2 = "blau'gelb"
           farben3 = "blau'rot"
           farben4 = "blau"
```

`FILTER_INDEX (liste, positiv, negativ, raus)`

Die Makrofunktion `FILTER_INDEX` nimmt in gleicher Weise wie die Makrofunktion `FILTER` den Inhalt der Variablen `liste` und wählt aus ihm Teilzeichenfolgen aus; falls `liste` eine Sternvariable ist, werden Zeilen ausgewählt. Als Funktionswert werden jedoch nicht die ausgewählten Teilzeichenfolgen bzw. Zeilen geliefert, sondern nur die Positionen, an denen sie in der Variablen `liste` stehen; diese Positionsangaben werden durch Apostroph getrennt.

Die Positionen der nicht ausgewählten Teilzeichenfolgen bzw. Zeilen werden der Variablen `raus` zugewiesen. Werden diese Positionen nicht benötigt, kann die Variable `raus` einschließlich des davor stehenden Kommas weggelassen werden.

```
Beispiel: $$ SET alt = "dies'und'das'oder'sonstwas"
          $$ SET index = FILTER_INDEX (alt, -, ":und:oder:")
```

```
Ergebnis: index = "1'3'5"
```

```
          $$ SET neu = INDEX_SORT (alt, index)
```

```
Ergebnis: neu = "dies'das'sonstwas"
```

`REDUCE (liste, raus)`

Die Makrofunktion `REDUCE` nimmt den Inhalt der Variablen `liste` und entfernt die Teilzeichenfolgen, die mit der jeweils vorangehenden übereinstimmen; falls `liste` eine Sternvariable ist, werden die Zeilen entfernt, die mit der jeweils vorangehenden übereinstimmen. Das Ergebnis wird als Funktionswert geliefert.

Bei der Prüfung auf Übereinstimmung werden Groß- und Kleinbuchstaben nicht unterschieden.

Die entfernten Teilzeichenfolgen bzw. Zeilen werden der Variablen `raus` zugewiesen. Werden die entfernten Teilzeichenfolgen bzw. Zeilen nicht benötigt, kann die Variable `raus` einschließlich des davor stehenden Kommas weggelassen werden.

```
Beispiel: $$ SET alt = "test'eins'EINS'ZWEI'zwei'test"
          $$ SET neu = REDUCE (alt, tst)
```

```
Ergebnis: neu = "test'eins'ZWEI'test"
          tst = "EINS'zwei"
```

```
          $$ SET hlf = ALPHA_SORT (alt)
```

```
          $$ SET neu = REDUCE (hlf)
```

```
Ergebnis: neu = "eins'test'ZWEI"
```

`REDUCE_INDEX (liste, raus)`

Die Makrofunktion `REDUCE_INDEX` nimmt den Inhalt der Variablen `liste` und prüft, welche Teilzeichenfolgen jeweils mit der vorangehenden übereinstimmen; falls `liste` eine Sternvariable ist, werden die Zeilen geprüft, ob sie mit der jeweils vorangehenden übereinstimmen. Als Funktionswert werden die Positionen der Teilzeichenfolgen bzw. der Zeilen geliefert, die jeweils mit der vorangehenden nicht übereinstimmen.

Bei der Prüfung auf Übereinstimmung werden Groß- und Kleinbuchstaben nicht unterschieden.

Die Positionen der Teilzeichenfolgen bzw. Zeilen, die mit der jeweils vorangehenden übereinstimmen, werden der Variablen `raus` zugewiesen. Werden diese Positionen nicht benötigt, kann die Variable `raus` einschließlich des davor stehenden Kommas weggelassen werden.

```
Beispiel: $$ SET alt = "test'eins'EINS'ZWEI'zwei'test"
          $$ SET index = REDUCE_INDEX (alt, mehrf)
```

```
Ergebnis: index = "1'2'4'6"
           mehrf = "3'5"
```

```
          $$ SET neu = INDEX_SORT (alt, mehrf)
```

```
Ergebnis: neu = "EINS'zwei"
```

`ACCUMULATE (liste, anzahl)`

Die Makrofunktion `ACCUMULATE` nimmt den Inhalt der Variablen `liste` und entfernt die Teilzeichenfolgen, die mit der jeweils vorangehenden übereinstimmen; falls `liste` eine Sternvariable ist, werden die Zeilen entfernt, die mit der jeweils vorangehenden übereinstimmen. Das Ergebnis wird als Funktionswert geliefert.

Bei der Prüfung auf Übereinstimmung werden Groß- und Kleinbuchstaben nicht unterschieden.

Für jede Zeichenfolge bzw. Zeile im Ergebnis wird in der Variablen `anzahl` angegeben, wie oft die jeweilige Zeichenfolge bzw. Zeile hintereinander vorgekommen ist.

```
Beispiel: $$ SET alt = "test'eins'EINS'TEST'ZWEI'zwei'test"
          $$ SET neu = ACCUMULATE (alt, anz)
```

```
Ergebnis: neu = "test'eins'TEST'ZWEI'test"
           anz = "1'2'1'2'1"
```

```
          $$ SET hlf = ALPHA_SORT (alt)
```

```
          $$ SET neu = ACCUMULATE (hlf, anz)
```

```
Ergebnis: neu = "eins'test'ZWEI"
           anz = "2'3'2"
```

`SUM (werte)`

Die Makrofunktion `SUM` addiert die Zahlenwerte, die in der Variablen `werte` als Teilzeichenfolgen (jeweils durch Apostroph getrennt) stehen.

```
Beispiel: $$ SET werte = "2'8'16'4"
          $$ SET summe = SUM (werte)
```

```
Ergebnis: anz = "30"
```

COMBINE (werte, erg2)

Die Makrofunktion COMBINE fasst die in der Variablen `werte` enthaltenen Teilzeichenfolgen zusammen. Enthalten die Teilzeichenfolgen nur eine Zahl, so werden gleiche und unmittelbar aufeinanderfolgende Zahlenwerte zusammengefasst (aus `1'2'2'3` wird z. B. `1-3`). Enthalten die Teilzeichenfolgen außer einer Zahl noch andere Zeichen, so werden nur solche zusammengefasst, die entweder gleich sind, oder die bis auf die am Ende der Teilzeichenfolgen stehenden Zahlen gleich sind und deren Zahlenwerte ebenfalls gleich sind oder unmittelbar aufeinanderfolgen (aus `a1'a2'a3'b4'b5` wird z. B. `a1-a3'b4-b5`).

In welcher Weise die Zahlenwerte zusammengefasst werden, wenn nur zwei aufeinanderfolgende Zahlenwerte vorhanden sind, kann mit der Variablen `erg2` vorgegeben werden. Enthält sie eine leere Zeichenfolge, so werden die beiden Werte nicht zusammengefasst; andernfalls wird an den ersten Zahlenwert der Inhalt der Variablen `erg2` angehängt und der zweite Zahlenwert entfällt (aus `1'2` wird z. B. `1f`). Wird die Variable `erg2` einschließlich des davor stehenden Kommas weggelassen, so werden zwei aufeinanderfolgende Zahlenwerte wie drei und mehr aufeinanderfolgende zusammengefasst (aus `1'2` wird z. B. `1-2`).

Wird nach der Variablen `erg2` durch Komma getrennt eine weitere Variable `erg3`, die eine nicht-leere Zeichenfolge enthält, angegeben, so wird bei genau drei aufeinanderfolgenden Zahlenwerten diese Zeichenfolge an den ersten Zahlenwert angehängt und der zweite und dritte Zahlenwert entfällt (aus `1'2'3` wird z. B. `1ff`).

Wird nach der Variablen `erg3` durch Komma getrennt noch eine weitere Variable `erg4`, die eine nicht-leere Zeichenfolge enthält, angegeben, so wird bei vier oder mehr aufeinanderfolgenden Zahlenwerten diese Zeichenfolge an den ersten Zahlenwert angehängt und der zweite und alle weiteren aufeinanderfolgenden Zahlenwerte entfallen (aus `1'2'3'4'9` oder `1'2'3'4'5'6'9` wird z. B. `1fff'9`).

An Stelle der Variablen `erg2`, `erg3` und `erg4` kann jeweils auch eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben werden.

```
Beispiel: $$ SET alt = "1'2'3'5'6'8'9'10'11'12'14'15'16'17"
          $$ SET neu = COMBINE (alt)
```

```
Ergebnis: neu = "1-3'5-6'8-12'14-17"
```

```
$$ SET neu = COMBINE (alt, "")
```

```
Ergebnis: neu = "1-3'5'6'8-12'14-17"
```

```
$$ SET neu = COMBINE (alt, "f")
```

```
Ergebnis: neu = "1-3'5f'8-12'14-17"
```

```
$$ SET neu = COMBINE (alt, "f", "ff")
```

```
Ergebnis: neu = "1ff'5f'8-12'14-17"
```

```
$$ SET neu = COMBINE (alt, "f", "ff", "fff")
```



Ergebnis: neu = "1ff'5f'8fff'14fff"

REVERSE (var)

Die Makrofunktion REVERSE nimmt den Inhalt der Variablen var und ordnet die Teilzeichenfolgen in umgekehrter Reihenfolge an; falls var eine Sternvariable ist, wird die Reihenfolge der Zeilen umgedreht. Das Ergebnis wird als Funktionswert geliefert.

```

Beispiel: $$ SET alt = "Anfang'Mitte'Schluss"
          $$ SET neu = REVERSE (alt)

```

Ergebnis: neu = "Schluss'Mitte'Anfang"

Hinweis: Mit der Makrofunktion TURN (siehe Seite 540) kann die Reihenfolge der einzelnen Zeichen umgedreht werden.

## Makrofunktionen für Wertelisten

Werteliste wird der Inhalt einer Sternvariablen genannt, wenn alle Zeilen einen Eintrag enthalten, der entweder aus einem Namen und einem Wert, die durch ein Gleichheitszeichen getrennt sind, oder aus einer Kennung und einem Wert, die durch ein Leerzeichen getrennt sind, besteht. Der Wert kann eine beliebige Zeichenfolge sein.

Beispiel einer Werteliste mit Namen:

```

Nummer=123
Autor=Peter Müller
Titel=Die Dolomiten

```

Beispiel einer Werteliste mit Kennungen:

```

*n 123
*a Peter Müller
*t Die Dolomiten

```

GET\_VALUE (liste, name, wert)

Die Makrofunktion GET\_VALUE extrahiert aus der in der Variablen liste enthaltenen Werteliste den Wert, dessen Name bzw. Kennung in der Variablen name angegeben ist, und liefert ihn als Funktionswert.

Enthält die Werteliste keinen entsprechenden Eintrag, liefert der Funktionswert den Inhalt der Variablen wert; soll in diesem Fall als Funktionswert eine leere Zeichenfolge geliefert werden, kann die Variable wert einschließlich des davor stehenden Kommas weggelassen werden.

An Stelle der Variablen name kann auch der Name bzw. die Kennung in Anführungszeichen eingeschlossen angegeben werden; an Stelle der Variablen wert kann auch eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben werden.

Werden die Argumente `name` und `wert` einschließlich des jeweils davor stehenden Kommas weggelassen, so werden die Namen bzw. die Kennungen der in der Variablen `liste` enthaltenen Werteliste durch Apostroph getrennt als Funktionswert geliefert.

```
Beispiel: $$ SET liste = *
          Autor=Peter Müller
          Titel=Die Dolomiten

          $$ SET wert = GET_VALUE (liste, "Titel", "(k.A.)")
Ergebnis: wert = "Die Dolomiten"

          $$ SET wert = GET_VALUE (liste, "Ort", "(k.A.)")
Ergebnis: wert = "(k.A.)"

          $$ SET wert = GET_VALUE (liste)
Ergebnis: wert = "Autor'Titel"
```

`SET_VALUE (liste, name, wert)`

Die Makrofunktion `SET_VALUE` nimmt die Werteliste in der Variablen `liste`, setzt den Wert, dessen Name bzw. Kennung in der Variablen `name` angegeben ist, auf den in der Variablen `wert` enthaltenen Wert und liefert das Ergebnis als Funktionswert.

Enthält die Werteliste noch keinen entsprechenden Eintrag, wird ein neuer Eintrag am Ende der Werteliste hinzugefügt.

An Stelle der Variablen `name` und `wert` kann jeweils auch eine entsprechende Zeichenfolge in Anführungszeichen eingeschlossen angegeben werden.

```
Beispiel: $$ SET liste = *
          Autor=Peter Müller
          Titel=Die Dolomiten

          $$ SET txt = "Der Rosengarten"
          $$ SET liste = SET_VALUE (liste, "Titel", txt)

Ergebnis: liste = *
          Autor=Peter Müller
          Titel=Der Rosengarten
```

`INSERT_VALUE (liste, name, wert, vor)`

Die Makrofunktion `INSERT_VALUE` nimmt die Werteliste in der Variablen `liste`, fügt einen Eintrag hinzu und liefert das Ergebnis als Funktionswert.

Name bzw. Kennung und Wert des Eintrags werden mit den Variablen `name` und `wert` bestimmt.

An welcher Stelle der Eintrag eingefügt wird, kann mit der Variablen `vor` bestimmt

werden; sie muss einen Namen bzw. eine Kennung oder mehrere durch Apostroph getrennte Namen bzw. Kennungen enthalten. Der Eintrag wird vor dem ersten Eintrag in der Werteliste eingefügt, dessen Name bzw. Kennung mit einem Namen bzw. einer Kennung in der Variablen `vor` übereinstimmt; falls kein Name bzw. keine Kennung übereinstimmt, wird der Eintrag am Ende der Werteliste hinzugefügt.

An Stelle der Variablen `name`, `wert` und `vor` kann jeweils auch eine entsprechende Zeichenfolge in Anführungszeichen eingeschlossen angegeben werden.

```
Beispiel: $$ SET liste = *
          Autor=Peter Müller
          Jahr=2019

          $$ SET name = "Titel", txt = "Der Rosengarten"
          $$ SET wo = "Jahr'Verlag"
          $$ SET liste = INSERT_VALUE (liste, name, txt, wo)

Ergebnis: liste = *
           Autor=Peter Müller
           Titel=Der Rosengarten
           Jahr=2019
```

Hinweis: Soll ein schon vorhandener Eintrag ersetzt werden oder der Eintrag als letzter eingefügt werden, kann die Makrofunktion `SET_VALUE` (siehe Seite 554) verwendet werden.

```
REMOVE_VALUE (liste, name, raus)
```

Die Makrofunktion `REMOVE_VALUE` nimmt die Werteliste in der Variablen `liste`, eliminiert den Eintrag mit dem Namen bzw. der Kennung, die in der Variablen `name` angegeben ist, und liefert das Ergebnis als Funktionswert.

Der Wert des entfernten Eintrags wird der Variablen `raus` zugewiesen. Wird dieser nicht benötigt, kann die Variable `raus` einschließlich des davor stehenden Kommas weggelassen werden.

An Stelle der Variablen `name` kann auch ein Name bzw. eine Kennung in Anführungszeichen eingeschlossen angegeben werden.

```
Beispiel: $$ SET liste = *
          Autor=Peter Müller
          Titel=Der Rosengarten
          Jahr=2019

          $$ SET liste = REMOVE_VALUE (liste, "Jahr", raus)

Ergebnis: raus = "2019"
           liste = *
           Autor=Peter Müller
           Titel=Der Rosengarten
```

ADD\_VALUES (liste1, liste2)

Die Makrofunktion ADD\_VALUES nimmt die Werteliste in der Variablen liste1, fügt die Einträge der Werteliste in der Variablen liste2 am Ende hinzu und liefert das Ergebnis als Funktionswert.

Ist bereits ein Eintrag mit dem gleichen Namen bzw. der gleichen Kennung vorhanden, so wird der Eintrag nicht hinzugefügt, sondern nur der Wert dieses Eintrags entsprechend geändert.

```
Beispiel: $$ SET liste = *
          Datum=23.10.2019
          Autor=Peter Müller
          Titel=Der Rosengarten

          $$ SET update = *
          Datum=11.12.2019
          Jahr=2019

          $$ SET liste = ADD_VALUES (liste, update)
```

```
Ergebnis: liste = *
           Datum=11.12.2019
           Autor=Peter Müller
           Titel=Der Rosengarten
           Jahr=2019
```

SORT\_VALUES (liste, namen)

Die Makrofunktion SORT\_VALUES nimmt die Werteliste in der Variablen liste, ordnet die Einträge in der Reihenfolge an, in der die Namen bzw. Kennungen in der Variablen namen angegeben sind, und liefert das Ergebnis als Funktionswert.

Die Namen bzw. Kennungen in der Variablen namen müssen jeweils durch Apostroph getrennt sein. Einträge, deren Namen bzw. Kennungen nicht angegeben sind, werden in alphabetischer Reihenfolge hinter den anderen angeordnet. Sollen alle Einträge alphabetisch angeordnet werden, kann die Variable namen einschließlich des davor stehenden Kommas weggelassen werden.

An Stelle der Variablen namen können auch die Namen bzw. Kennungen in Anführungszeichen eingeschlossen angegeben werden.

```
Beispiel: $$ SET liste = *
          Autor=Peter Müller
          Titel=Der Rosengarten
          Jahr=2019

          $$ SET liste = SORT_VALUES (liste)

Ergebnis: liste = *
           Autor=Peter Müller
           Jahr=2019
```

```
Titel=Der Rosengarten
```

```
$$ SET reihenfolge = "Jahr'Autor'Titel"
```

```
$$ SET liste = SORT_VALUES (liste, reihenfolge)
```

```
Ergebnis: liste = *
           Jahr=2019
           Autor=Peter Müller
           Titel=Der Rosengarten
```

```
INDEX_VALUE (liste, name)
```

Die Makrofunktion `INDEX_VALUE` sucht in der in der Variablen `liste` enthaltenen Werteliste den Eintrag, dessen Name bzw. Kennung in der Variablen `name` enthalten ist, und liefert die Nummer dieses Eintrags innerhalb der Werteliste als Funktionswert; enthält die Werteliste keinen entsprechenden Eintrag, ist der Funktionswert 0 (Null).

An Stelle der Variablen `name` kann auch der Name bzw. die Kennung in Anführungszeichen eingeschlossen angegeben werden.

```
Beispiel: $$ SET liste = *
           Autor=Peter Müller
           Titel=Die Dolomiten
           Jahr=2019
```

```
$$ SET num = INDEX_VALUE (liste, "Titel")
```

```
Ergebnis: num = 2
```

```
ASSIGN (liste, abk, trenner, var1, var2, var3, ...)
```

Mit der Makrofunktion `ASSIGN` können Werte aus einer Werteliste abgefragt werden. Wertelisten mit Kennungen werden von `ASSIGN` nicht unterstützt; für Wertelisten mit Namen gelten gegenüber den vorangehenden Makrofunktionen (aus historischen Gründen) andere Regeln.

Die Makrofunktion `ASSIGN` erwartet in der Variablen `liste` eine Werteliste mit folgendem Format:

```
name: wert1 ' wert2 ' wert3 ' ...
```

Der Name muss mit einem Doppelpunkt oder Gleichheitszeichen abgeschlossen sein. Die einzelnen Werte müssen durch Zeichenfolgen getrennt sein, die in der Such-Tabelle `trenner` als Suchzeichenfolge enthalten sind. Für das Argument `trenner` kann der Name einer mit der `BUILD`-Anweisung definierten Such-Tabelle oder direkt eine in Anführungszeichen eingeschlossene Such-Tabelle angegeben werden.

Zeilen, die keinen Doppelpunkt und kein Gleichheitszeichen enthalten, werden als Kommentarzeilen interpretiert.

```

Beispiel: $$ SET liste = *
           Vorw.   Telef.   Fax
Anton: 07071 ' 12345 ' 12345
Paul:  07071 ' 34561 ' 34562
Paula: 07071 ' 56781 ' 56782
Peter: 089   ' 135790 ' -

```

Die Makrofunktion `ASSIGN` sucht in der Sternvariablen `liste` diejenigen Zeilen, deren Name mit der Abkürzung beginnt (oder übereinstimmt), die mit dem Argument `abk` angegeben ist. Für das Argument `abk` kann eine in Anführungszeichen eingeschlossene Zeichenfolge oder eine Variable, die die Abkürzung enthält, angegeben werden.

Wird keine solche Zeile gefunden, so ist der Funktionswert eine leere Zeichenfolge; den Variablen `var1`, `var2`, `var3`, ... wird eine leere Zeichenfolge zugewiesen.

```

Beispiel: Variable liste wie im obigen Beispiel
          $$ SET name = "x"
          $$ SET test = ASSIGN (liste,name,":'":,vw,tnr,fnr)

Ergebnis: test = ""
           vw = "", tnr = "", fnr = ""

```

Wird genau eine solche Zeile gefunden, wird als Funktionswert die Vollform (das ist der Name) der Abkürzung geliefert. Den Variablen `var1`, `var2`, `var3`, ... werden die in der gefundenen Zeile stehenden Werte `wert1`, `wert2`, `wert3`, ... zugewiesen.

```

Beispiel: Variable liste wie im obigen Beispiel
          $$ SET name = "pe"
          $$ SET test = ASSIGN (liste,name,":'":,vw,tnr,fnr)

Ergebnis: test = "Peter"
           vw = "089", tnr = "135790", fnr = "-"

```

Wird eine Zeile gefunden, bei der die in der Variablen `abk` stehende Abkürzung mit der Vollform (das ist der Name) übereinstimmt, wird als Funktionswert diese Vollform geliefert, unabhängig davon, ob noch andere Vollformen mit dieser Abkürzung beginnen. Den Variablen `var1`, `var2`, `var3`, ... werden die in der gefundenen Zeile stehenden Werte `wert1`, `wert2`, `wert3`, ... zugewiesen.

```

Beispiel: Variable liste wie im obigen Beispiel
          $$ SET name = "paul"
          $$ SET test = ASSIGN (liste,name,":'":,vw,tnr,fnr)

Ergebnis: test = "Paul"
           vw = "07071", tnr = "34561", fnr = "34562"

```

Werden mehrere Zeilen gefunden, deren Name mit der in der Variablen `abk` enthaltenen Abkürzung beginnen (aber nicht übereinstimmen), so enthält der Funktionswert alle in Frage kommenden Namen durch Apostroph getrennt; den Variablen `var1`, `var2`, `var3`, ... wird eine leere Zeichenfolge zugewiesen.

Beispiel: Variable liste wie im obigen Beispiel

```
$$ SET name = "p"
```

```
$$ SET test = ASSIGN (liste,name,":'":",vw,tnr,fnr)
```

Ergebnis: test = "Paul'Paula'Peter"

```
vw = "", tnr = "", fnr = ""
```

Bei der Prüfung auf Übereinstimmung werden Groß- und Kleinbuchstaben nicht unterschieden.

Die unterteilenden Zeichenfolgen werden nicht in die Variablen übernommen. Sollen sie mit übernommen werden, muss entweder unmittelbar vor bzw. nach dem Argument `trenner` ein senkrechter Strich angegeben werden. Die unterteilende Zeichenfolge wird dann in die Variable mit dem nachfolgenden Text bzw. in die Variable mit dem vorangehenden Text übernommen.

## Makrofunktionen für Tags

Ein Tag ist eine in spitzen Klammern eingeschlossene Zeichenfolge. Es gibt Anfangs-Tags, Ende-Tags und leere Tags:

Anfangs-Tag:

```
<name>
```

```
<name attribut1=wert1, attribut2=wert2, ... >
```

Ende-Tag:

```
</name>
```

Leeres Tag:

```
<name/>
```

```
<name attribut1=wert1, attribut2=wert2, ... />
```

Damit die nachfolgend beschriebenen Makrofunktionen die angegebene Wirkung haben, müssen folgende Bedingungen erfüllt sein:

- Die spitzen Klammern dürfen nur als Anfangs- bzw. Endekennung von Tags, in Akzent-Codierungen (z. B. »%<«) oder in den Codes für doppelte Anführungszeichen (»#. <« und »#. >«) vorkommen; andere spitze Klammern müssen z. B. mit ^< bzw. ^> codiert sein.
- Tag- und Attribut-Namen dürfen aus Buchstaben, Ziffern, Minuszeichen, Punkt, Doppelpunkt und »\_« bestehen; das erste Zeichen darf kein Minuszeichen und kein Punkt sein.

START\_TAG (tag)

Die Makrofunktion `START_TAG` bildet aus dem in der Variablen `tag` enthaltenen Tag oder Namen ein Anfangs-Tag. Enthält das Tag Attribute, werden diese übernommen.

```
Beispiel: $$ SET tag = "</title>"
          $$ SET tag = START_TAG (tag)
```

```
Ergebnis: tag = "<title>"
```

```
          $$ SET name = "abc"
          $$ SET tag = START_TAG (name)
```

```
Ergebnis: tag = "<abc>"
```

END\_TAG (tag)

Die Makrofunktion END\_TAG bildet aus dem in der Variablen tag enthaltenen Tag oder Namen ein Ende-Tag. Enthält das Tag Attribute, werden diese eliminiert.

```
Beispiel: $$ SET tag = "<title align='center'>"
          $$ SET tag = END_TAG (tag)
```

```
Ergebnis: tag = "</title>"
```

```
          $$ SET name = "abc"
          $$ SET tag = END_TAG (name)
```

```
Ergebnis: tag = "</abc>"
```

EMPTY\_ELEMENT\_TAG (tag)

Die Makrofunktion EMPTY\_ELEMENT\_TAG bildet aus dem in der Variablen tag enthaltenen Tag oder Namen ein leeres Tag. Enthält das Tag Attribute, werden diese übernommen.

```
Beispiel: $$ SET tag = "<title align='center'>"
          $$ SET tag = EMPTY_ELEMENT_TAG (tag)
```

```
Ergebnis: tag = "<title align='center' />"
```

```
          $$ SET name = "abc"
          $$ SET tag = EMPTY_ELEMENT_TAG (name)
```

```
Ergebnis: tag = "<abc />"
```

GET\_TAG\_NAME (tag)

Die Makrofunktion GET\_TAG\_NAME liefert als Funktionswert den Namen des in der Variablen tag enthaltenen Tags.

```
Beispiel: $$ SET tag = "<title type='std'>"
          $$ SET name = GET_TAG_NAME (tag)
```

```
Ergebnis: name = "title"
```

SET\_TAG\_NAME (tag, name)

Die Makrofunktion SET\_TAG\_NAME nimmt das Tag in der Variablen tag, ersetzt



den Namen des Tags durch den in der Variablen `name` enthaltenen Namen und liefert das Ergebnis als Funktionswert.

An Stelle der Variablen `name` kann auch der neue Name des Tags in Anführungszeichen eingeschlossen angegeben werden. Enthält die Variable `name` ein Tag, so wird der Name dieses Tags als neuer Tagname übernommen.

```
Beispiel: $$ SET tag = "<title type='std'>"
          $$ SET tag = SET_TAG_NAME (tag, "Titel")

Ergebnis: tag = "<Titel type='std'>"
```

`GET_ATTRIBUTE (tag, name, wert)`

Die Makrofunktion `GET_ATTRIBUTE` extrahiert aus dem in der Variablen `tag` enthaltenen Tag den Wert des mit der Variablen `name` angegebenen Attributs und liefert ihn als Funktionswert.

Enthält das Tag kein entsprechendes Attribut, liefert der Funktionswert den Inhalt der Variablen `wert`; soll in diesem Fall als Funktionswert eine leere Zeichenfolge geliefert werden, kann die Variable `wert` einschließlich des davor stehenden Kommas weggelassen werden.

An Stelle der Variablen `name` kann auch der Name des Attributs in Anführungszeichen eingeschlossen angegeben werden; an Stelle der Variablen `wert` kann auch eine in Anführungszeichen eingeschlossene Zeichenfolge angegeben werden.

```
Beispiel: $$ SET tag = "<title align='center'>"
          $$ SET wert = GET_ATTRIBUTE (tag, "align", "none")

Ergebnis: wert = "center"
```

Werden die Argumente `name` und `wert` einschließlich des jeweils davor stehenden Kommas weggelassen, so werden die Namen der Attribute des in der Variablen `tag` enthaltenen Tags durch Apostroph getrennt als Funktionswert geliefert.

```
Beispiel: $$ SET tag = "<title align='center' type='std'>"
          $$ SET wert = GET_ATTRIBUTE (tag)

Ergebnis: wert = "align'type"
```

`SET_ATTRIBUTE (tag, name, wert)`

Die Makrofunktion `SET_ATTRIBUTE` nimmt das Tag in der Variablen `tag`, setzt den Wert des mit der Variablen `name` angegebenen Attributs auf den in der Variablen `wert` enthaltenen Wert und liefert das Ergebnis als Funktionswert.

Enthält das Tag noch kein Attribut mit dem angegebenen Namen, wird es als letztes Attribut hinzugefügt. Enthält die Variable `tag` nur einen Namen, wird mit ihm ein Anfangs-Tag mit dem entsprechenden Attribut gebildet.

An Stelle der Variablen `tag`, `name` und `wert` kann jeweils auch eine entsprechende Zeichenfolge in Anführungszeichen eingeschlossen angegeben werden.

Beispiel: `$$ SET tag = SET_ATTRIBUTE ("title", "type", "std")`

Ergebnis: `tag = "<title type='std'>"`

`$$ SET tag = SET_ATTRIBUTE (tag, "type", "normal")`

Ergebnis: `tag = "<title type='normal'>"`

Attribute können in einem Tag auch dadurch gesetzt werden, dass sie von einem anderen Tag übertragen werden. Dafür gibt es die Makrofunktionen `ADD_ATTRIBUTES` (siehe Seite 562) und `APPEND_ATTRIBUTES` (siehe Seite 563).

`INSERT_ATTRIBUTE (tag, name, wert, vor)`

Die Makrofunktion `INSERT_ATTRIBUTE` nimmt das Tag in der Variablen `tag`, fügt ein Attribut ein und liefert das Ergebnis als Funktionswert.

Name und Wert des Attributs werden mit den Variablen `name` und `wert` bestimmt.

An welcher Stelle das neue Attribut eingefügt wird, kann mit der Variablen `vor` bestimmt werden; sie muss den Namen eines Attributs oder mehrere durch Apostroph getrennte Namen von Attributen enthalten. Das neue Attribut wird vor dem ersten Attribut im Tag eingefügt, dessen Name mit einem Namen in der Variablen `vor` übereinstimmt; falls kein Name übereinstimmt, wird es als letztes Attribut hinzugefügt.

An Stelle der Variablen `name`, `wert` und `vor` kann jeweils auch eine entsprechende Zeichenfolge in Anführungszeichen eingeschlossen angegeben werden.

Beispiel: `$$ SET t = "<t type='std' align='left'>", l = 0`  
`$$ SET t = INSERT_ATTRIBUTE (t, "lvl", l, "align")`

Ergebnis: `t = <t type='std' lvl='0' align='left'>`

Hinweis: Soll ein schon vorhandenes Attribut ersetzt werden oder das Attribut als letztes eingefügt werden, kann die Makrofunktion `SET_ATTRIBUTE` (siehe Seite 561) verwendet werden.

`ADD_ATTRIBUTES (tag1, tag2)`

Die Makrofunktion `ADD_ATTRIBUTES` nimmt das Tag in der Variablen `tag1`, fügt die Attribute des Tags in der Variablen `tag2` am Ende hinzu und liefert das Ergebnis als Funktionswert.

Ist bereits ein gleichnamiges Attribut vorhanden, so wird das Attribut nicht hinzugefügt, sondern der Wert des Attributs entsprechend geändert (im Gegensatz zur nachfolgend beschriebenen Makrofunktion `APPEND_ATTRIBUTES`).

An Stelle der Variablen `tag1` kann auch eine entsprechende Zeichenfolge in Anführungszeichen eingeschlossen angegeben werden.

Beispiel: `$$ SET tag1 = "<title align='center'>"`  
`$$ SET tag2 = "<egal type='std' align='left'>"`

`$$ SET tag = ADD_ATTRIBUTES (tag1, tag2)`

Ergebnis: tag = "<title align='left' type='std'>"

```
$$ SET tag = ADD_ATTRIBUTES ("<abc>", tag2)
```

Ergebnis: tag = "<abc type='std' align='left'>"

APPEND\_ATTRIBUTES (tag1, tag2)

Die Makrofunktion APPEND\_ATTRIBUTES nimmt das Tag in der Variablen tag1, fügt die Attribute des Tags in der Variablen tag2 am Ende hinzu und liefert das Ergebnis als Funktionswert.

Ist bereits ein gleichnamiges Attribut vorhanden, so wird das Attribut nicht hinzugefügt und der Wert des Attributs bleibt unverändert (im Gegensatz zur zuvor beschriebenen Makrofunktion ADD\_ATTRIBUTES).

An Stelle der Variablen tag1 kann auch eine entsprechende Zeichenfolge in Anführungszeichen eingeschlossen angegeben werden.

Beispiel: \$\$ SET tag1 = "<title align='center'>"

```
$$ SET tag2 = "<egal type='std' align='left'>"
```

```
$$ SET tag = APPEND_ATTRIBUTES (tag1, tag2)
```

Ergebnis: tag = "<title align='center' type='std'>"

```
$$ SET tag = APPEND_ATTRIBUTES ("<abc>", tag2)
```

Ergebnis: tag = "<abc type='std' align='left'>"

RENAME\_ATTRIBUTE (tag, altername, neuename)

Die Makrofunktion RENAME\_ATTRIBUTE nimmt das Tag in der Variablen tag, benennt den Namen eines Attributs um und liefert das Ergebnis als Funktionswert.

An Stelle der Variablen altername und neuename kann jeweils auch der alte bzw. neue Name des Attributs in Anführungszeichen eingeschlossen angegeben werden.

Beispiel: \$\$ SET t = "<document titel='Test'>"

```
$$ SET t = RENAME_ATTRIBUTE (t, "titel", "title")
```

Ergebnis: t = "<document title='Test'>"

REMOVE\_ATTRIBUTE (tag, name, wert)

Die Makrofunktion REMOVE\_ATTRIBUTE nimmt das Tag in der Variablen tag, eliminiert das mit der Variablen name angegebene Attribut und liefert das Ergebnis als Funktionswert.

Der Wert des entfernten Attributs wird der Variablen wert zugewiesen. Wird dieser nicht benötigt, kann die Variable wert einschließlich des davor stehenden Kommas weggelassen werden.

An Stelle der Variablen `name` kann auch der Name des Attributs in Anführungszeichen eingeschlossen angegeben werden.

Beispiel: `$$ SET tag = "<title align='center' type='std'>"`

```
$$ SET tag = REMOVE_ATTRIBUTE (tag, "type")
```

Ergebnis: `tag = "<title align='center'>"`

`REMOVE_ATTRIBUTES (tag)`

Die Makrofunktion `REMOVE_ATTRIBUTES` nimmt das Tag in der Variablen `tag`, eliminiert alle Attribute und liefert das Ergebnis als Funktionswert.

Beispiel: `$$ SET tag = "<title align='center' type='std'>"`

```
$$ SET tag = REMOVE_ATTRIBUTES (tag)
```

Ergebnis: `tag = "<title>"`

`SORT_ATTRIBUTES (tag, namen)`

Die Makrofunktion `SORT_ATTRIBUTES` nimmt das Tag in der Variablen `tag`, ordnet die Attribute des Tags in der Reihenfolge an, in der die Namen der Attribute in der Variablen `namen` angegeben sind, und liefert das Ergebnis als Funktionswert.

Die Namen in der Variablen `namen` müssen durch Apostroph getrennt sein. Attribute, deren Namen nicht angegeben sind, werden in alphabetischer Reihenfolge hinter den anderen angeordnet. Sollen alle Attribute alphabetisch angeordnet werden, kann die Variable `namen` einschließlich des davor stehenden Kommas weggelassen werden.

An Stelle der Variablen `namen` können auch die Namen der Attribute in Anführungszeichen eingeschlossen angegeben werden.

Beispiel: `$$ SET tag = "<title align='left' type='std'>"`

```
$$ SET tag = SORT_ATTRIBUTES (tag, "type'align")
```

Ergebnis: `tag = "<title type='std' align='left'>"`

```
$$ SET tag = SORT_ATTRIBUTES (tag)
```

Ergebnis: `tag = "<title align='left' type='std'>"`

```
$$ SET tag = SORT_ATTRIBUTES (tag, "type")
```

Ergebnis: `tag = "<title type='std' align='left'>"`

`SPLIT_TAG (tag, name)`

Die Makrofunktion `SPLIT_TAG` nimmt das Tag in der Variablen `tag`, speichert den Namen des Tags in die Variable `name`, bildet aus den Attributen eine Werteliste, wobei jedes Attribut eine Zeile belegt, und liefert das Ergebnis (zum Speichern in einer Sternvariablen) als Funktionswert.

Beispiel: `$$ SET tag = "<title type='std' align='left'>"`

```
$$ SET liste = SPLIT_TAG (tag, name)
```

Ergebnis: `name = "title"`  
`liste = *`  
`type=std`  
`align=left`

Hinweis: Zum Bearbeiten von Wertelisten gibt es spezielle Makrofunktionen (siehe Seite 553).

`JOIN_TAG (name, liste)`

Die Makrofunktion `JOIN_TAG` bildet aus dem Namen in der Variablen `name` und der Werteliste in der Variablen `liste` ein Anfangs-Tag und liefert das Ergebnis als Funktionswert.

An Stelle der Variablen `name` kann auch der Name des Tags in Anführungszeichen eingeschlossen angegeben werden.

Beispiel: `$$ SET liste = *`  
`type=std`  
`align=left`

```
$$ SET tag = JOIN_TAG ("title", liste)
```

Ergebnis: `tag = "<title type='std' align='left'>"`

Hinweis: Zum Erstellen und Bearbeiten von Wertelisten gibt es spezielle Makrofunktionen (siehe ab Seite 553).

`REMOVE_TAGS (var, name)`

Die Makrofunktion `REMOVE_TAGS` nimmt den Inhalt der Variablen `var`, eliminiert alle darin enthaltenen Tags mit dem in der Variablen `name` angegebenen Namen und liefert das Ergebnis als Funktionswert.

Die Variable `name` kann auch mehrere Namen durch Apostroph getrennt enthalten. An Stelle von Tag-Namen können auch Tags angegeben werden; falls die Tags auch Attribute enthalten, sind diese bedeutungslos.

Sollen alle Tags entfernt werden, kann die Variable `name` einschließlich des davor stehenden Kommas weggelassen werden.

An Stelle der Variablen `name` können die Namen der Tags auch in Anführungszeichen eingeschlossen angegeben werden.

Beispiel: `$$ SET alt = "<text>anfang <i>und</i> ende</text>"`

```
$$ SET neu = REMOVE_TAGS (alt, "b'i")
```

Ergebnis: `neu = "<text>anfang und ende</text>"`

```
$$ SET neu = REMOVE_TAGS (alt)
```

```
Ergebnis: neu = "anfang und ende"
```

```
INDENT_TAGS (var, ign, erg1, erg2)
```

Die Makrofunktion `INDENT_TAGS` unterteilt die in der Variablen `var` enthaltene Zeichenfolge in einzelne Zeilen, fügt am Zeilenanfang ggf. zusätzlich Zeichenfolgen ein und liefert das Ergebnis (zum Speichern in einer Sternvariablen) als Funktionswert.

Eine neue Zeile wird jeweils vor einem Anfangs-Tag, vor und nach einem leeren Tag, sowie nach einem Ende-Tag begonnen. Falls nach dieser Aufteilung das zu einem Anfangs-Tag gehörende Ende-Tag nicht in der gleichen Zeile steht, wird nach einem solchen Anfangs-Tag und vor dem zugehörigen Ende-Tag zusätzlich eine neue Zeile begonnen.

Sollen bestimmte Tags wie Text behandelt werden (d. h. dass sie keinen Zeilenwechsel verursachen), kann zum Argument `ign` der Name einer mit der `BUILD`-Anweisung definierten Recherchier-Tabelle angegeben werden; andernfalls muss für das Argument `ign` ein Minuszeichen angegeben werden. Ein Tag wird wie Text behandelt, wenn es den Erfordernissen der angegebenen Tabelle genügt. Dies wird am Inhalt einer internen Variablen überprüft, die alle Namen der noch offenen Tags und den Namen des zu überprüfenden Tags enthält. Die Namen sind jeweils in spitze Klammern eingeschlossen und stehen ohne weitere Trennzeichen hintereinander.

Am Anfang jeder Zeile werden die mit den Variablen `erg1` und `erg2` angegebenen Zeichenfolgen ergänzt. Dazwischen wird eine Zahl eingefügt. Sie gibt an, wieviele Tags am Anfang der Zeile offen sind; falls in einer Zeile jedoch nur ein Ende-Tag steht, gibt die Zahl an, wieviele Tags am Ende der Zeile noch offen sind. Wird die Variable `erg2` einschließlich des davor stehenden Kommas weggelassen, so wird diese Zahl nicht eingefügt, sondern es wird die mit der Variablen `erg1` angegebene Zeichenfolge entsprechend oft ergänzt.

Beispiel: \$\$ = {}

```
$$ SET t = "<a>aa<b>bb<c>cc</c>bb<c/>bb</b>aa</a>"
```

```
$$ SET text = INDENT_TAGS (t, -, "<i lvl='", "'>")
```

```
Ergebnis: text = *
<i lvl='0'><a>
<i lvl='1'>aa
<i lvl='1'><b>
<i lvl='2'>bb
<i lvl='2'><c>cc</c>
<i lvl='2'>bb
<i lvl='2'><c/>
<i lvl='2'>bb
<i lvl='1'></b>
<i lvl='1'>aa
<i lvl='0'></a>
```

```
$$ SET text = INDENT_TAGS (t, -, "  ")
```

```
Ergebnis: text = *
<a>
  aa
  <b>
    bb
    <c>cc</c>
    bb
    <c/>
    bb
  </b>
  aa
</a>
```

```
$$ BUILD R_TABLE/TEXT b = ":*<b>:"
```

```
$$ SET text = INDENT_TAGS (t, b, "  ")
```

```
Ergebnis: text = *
<a>
  aa<b>bb
  <c>cc</c>
  bb
  <c/>
  bb</b>aa
</a>
```

```

$$ BUILD R_TABLE/TEXT c = " :*<c>:"
$$ SET text = INDENT_TAGS (t, c, "  ")

```

```

Ergebnis: text = *
           <a>
             aa
             <b>bb<c>cc</c>bb<c/>bb</b>
             aa
           </a>

```

```
CHECK_TAGS (var, leer, apos1, epos1, apos2, epos2)
```

Die Makrofunktion `CHECK_TAGS` überprüft die in der Variablen `var` enthaltenen Tags auf Paarigkeit. Wird eine spitze Klammer oder ein Tag ohne entsprechenden Partner gefunden, so wird das Überprüfen vorzeitig beendet. Leere Tags werden beim Prüfen auf Paarigkeit übergangen.

Enthält die Variable `var` Anfangs-Tags, zu denen grundsätzlich keine Ende-Tags vorhanden sind und die deshalb beim Überprüfen übergangen werden sollen, kann zum Argument `leer` der Name einer mit der `BUILD`-Anweisung definierten Recherchier-Tabelle angegeben werden; andernfalls muss für das Argument `leer` ein Minuszeichen angegeben werden. Ein Tag wird übergangen, wenn es den Erfordernissen der angegebenen Tabelle genügt. Dies wird am Inhalt einer internen Variablen überprüft, die alle Namen der noch offenen Tags und den Namen des zu überprüfenden Tags enthält. Die Namen sind jeweils in spitze Klammern eingeschlossen und stehen ohne weitere Trennzeichen hintereinander.

Der Funktionswert ist

- 0, falls keine Fehler gefunden wurden. Die Variablen `apos1`, `epos1`, `apos2` und `epos2` erhalten ebenfalls den Wert 0 (Null).
- -1, falls »<<« ohne entsprechende »>>« gefunden wurde. Die Variable `apos1` nennt die Position von »<<«, `epos1` die Position dahinter, `apos2` und `epos2` nennen die Position, bis zu der »>>« spätestens stehen müsste.
- -2, falls »>>« ohne entsprechende »<<« gefunden wurde. Die Variablen `apos1` und `epos1` nennen die Position, ab der »<<« frühestens stehen sollte, `apos2` nennt die Position von »>>«, `epos2` die Position dahinter.
- 1, falls ein Anfangs-Tag ohne entsprechendes Ende-Tag gefunden wurde. Die Variable `apos1` nennt die Position des Anfangs-Tags, `epos1` die Position hinter dem Anfangs-Tag, `apos2` und `epos2` nennen die Position, bis zu der das Ende-Tag spätestens stehen müsste.
- 2, falls ein Ende-Tag ohne entsprechendes Anfangs-Tag gefunden wurde. Die Variablen `apos1` und `epos1` nennen die Position, ab der das Anfangs-Tag frühestens stehen sollte, `apos2` nennt die Position des Ende-Tags, `epos2` die Position hinter dem Ende-Tag.
- 3, falls ein Anfangs-Tag und ein Ende-Tag gefunden wurden, deren Namen nicht übereinstimmen. Die Variable `apos1` nennt die Position des Anfangs-Tags, `epos1` die Position hinter dem Anfangs-Tag, `apos2` nennt die Position des Ende-Tags und `epos2` die Position hinter dem Ende-Tag.



```

Beispiel: $$ SET txt = "<a>...<...</a>"
          $$ SET i = CHECK_TAGS (txt, -, a1, e1, a2, e2)
Ergebnis: i = -1, a1 = 7, e1 = 8, a2 = 11, e2 = 11

          $$ SET txt = "<a>...>...</a>"
          $$ SET i = CHECK_TAGS (txt, -, a1, e1, a2, e2)
Ergebnis: i = -2, a1 = 4, e1 = 4, a2 = 7, e2 = 8

          $$ SET txt = "<a>...<b>...</a>"
          $$ SET i = CHECK_TAGS (txt, -, a1, e1, a2, e2)
Ergebnis: i = 1, a1 = 7, e1 = 10, a2 = 13, e2 = 13

          $$ SET txt = "<a>...</b>...</a>"
          $$ SET i = CHECK_TAGS (txt, -, a1, e1, a2, e2)
Ergebnis: i = 2, a1 = 4, e1 = 4, a2 = 7, e2 = 11

          $$ SET txt = "<a>...<b>...</x>...</a>"
          $$ SET i = CHECK_TAGS (txt, -, a1, e1, a2, e2)
Ergebnis: i = 3, a1 = 7, e1 = 10, a2 = 13, e2 = 17

          $$ BUILD R_TABLE p = " :<p>:"
          $$ SET txt = "<a>...<p>...<p>...</a>"
          $$ SET i = CHECK_TAGS (txt, p, a1, e1, a2, e2)
Ergebnis: i = 0, a1 = 0, e1 = 0, a2 = 0, e2 = 0

```

## Makrofunktion zum Prüfen von Klammern

CHECK\_BRACKETS (var, klammern, apos1, epos1, apos2, epos2)

Die Makrofunktion CHECK\_BRACKETS überprüft die in der Variablen var enthaltenen Klammern auf Paarigkeit und auf zulässige Verschachtelung. Außerdem kann geprüft werden, ob bestimmte Zeichenfolgen nur außerhalb bzw. innerhalb von bestimmten Klammern vorkommen.

Klammern dürfen nur paarweise vorkommen. Steht eine öffnende Klammer innerhalb eines anderen Klammerpaares, so muss die dazugehörige schließende Klammer ebenfalls innerhalb des selben Klammerpaares stehen; es ist also bei entsprechenden Angaben z. B. »(...[...])...« erlaubt, keinesfalls aber »(...[...])...[...].«.

Für das Argument klammern muss der Name einer mit der BUILD-Anweisung definierten Austausch-Tabelle angegeben werden. Mit der Tabelle kann bestimmt werden, welche Zeichen bzw. Zeichenfolgen als öffnende und schließende Klammern gelten, und welche Zeichen bzw. Zeichenfolgen innerhalb welcher Klammern vorkommen dürfen. Die Syntax der Angaben in der Tabelle ist unter »Prüfen von Klammern« (siehe Seite 594) beschrieben.

Der Funktionswert liefert die Zeichenfolge

- »OK«, falls keine Fehler gefunden wurden. Die Variablen `apos1`, `epos1`, `apos2` und `epos2` erhalten den Wert 0 (Null).
- »ERROR«, falls eine unerlaubte Klammer/Zeichenfolge gefunden wurden. Die Variable `apos1` nennt die Position der letzten noch offenen Klammer, `epos1` die Position hinter dieser Klammer, `apos2` nennt die Position der unerlaubten Klammer/Zeichenfolge und `epos2` die Position hinter dieser Klammer/Zeichenfolge; falls keine Klammer (mehr) offen ist, erhalten die Variablen `apos1` und `epos1` den Wert 1.

Beispiel: `$$ BUILD X_TABLE k1 = *`

```
:\[: (1:\]:)1:#/+: (2:#/-:)2:
$$ SET t = "...[...].#/+..."
$$ SET s = CHECK_BRACKETS (t, k1, a1,e1, a2,e2)
```

Ergebnis: `s = "ERROR"`  
`a1 = 12, e1 = 15, a2 = 18, e2 = 19`

## Makrofunktionen zum Sortieren

`ALPHA_SORT (var)`

Die Makrofunktion `ALPHA_SORT` nimmt den Inhalt der Variablen `var` und sortiert die einzelnen Teilzeichenfolgen alphabetisch; falls `var` eine Sternvariable ist, werden die einzelnen Zeilen sortiert. Dabei werden zum Sortieren kürzere Teilzeichenfolgen bzw. Zeilen rechts mit Leerzeichen aufgefüllt, die nachher wieder entfernt werden. Das Ergebnis wird als Funktionswert geliefert.

Zum Sortieren wird die Standard-Sortierfolge (siehe Seite 841) zugrunde gelegt. Für Teilzeichenfolgen bzw. Zeilen, die sich auf Grund der Standard-Sortierfolge nicht unterscheiden, wird zusätzlich die Sonder-Sortierfolge (siehe Seite 841) zugrunde gelegt.

Beispiel: `$$ SET alt = "eins'zwei'drei'vier'fünf'sechs"`  
`$$ SET neu = ALPHA_SORT (alt)`  
 Ergebnis: `neu = "drei'eins'fünf'sechs'vier'zwei"`

`ALPHA_INDEX (var)`

Die Makrofunktion `ALPHA_INDEX` nimmt den Inhalt der Variablen `var` und sortiert die einzelnen Teilzeichenfolgen bzw. Zeilen in gleicher Weise wie die Makrofunktion `ALPHA_SORT`; dabei wird festgehalten, an wievielter Position die Teilzeichenfolgen bzw. Zeilen ursprünglich standen. Diese Positionsangaben werden durch Apostroph getrennt als Funktionswert geliefert.

Beispiel: `$$ SET alt = "eins'zwei'drei'vier'fünf'sechs"`  
`$$ SET index = ALPHA_INDEX (alt)`

Ergebnis: `index = "3'1'5'6'4'2"`

`$$ SET neu = INDEX_SORT (alt, index)`

Ergebnis: `neu = "drei'eins'fünf'sechs'vier'zwei"`

#### MIXED\_SORT (var)

Die Makrofunktion `MIXED_SORT` nimmt den Inhalt der Variablen `var` und sortiert die einzelnen Teilzeichenfolgen alphabetisch; falls `var` eine Sternvariable ist, werden die einzelnen Zeilen sortiert. Dabei werden zum Sortieren kürzere Teilzeichenfolgen bzw. Zeilen rechts mit Leerzeichen aufgefüllt, die nachher wieder entfernt werden. Das Ergebnis wird als Funktionswert geliefert.

Zum Sortieren wird die Standard-Sortierfolge (siehe Seite 841) zugrunde gelegt. Für Teilzeichenfolgen bzw. Zeilen, die sich auf Grund der Standard-Sortierfolge nicht unterscheiden, wird zusätzlich die Sonder-Sortierfolge (siehe Seite 841) zugrunde gelegt.

Falls beim Bestimmen der Reihenfolge zweier Teilzeichenfolgen bzw. Zeilen zwei Ziffern verglichen werden müssten, werden statt der beiden Ziffern die Werte der Zahlen verglichen, die mit der jeweiligen Ziffer beginnen.

Beispiel: `$$ SET alt = "dat2'dat1'dat10'dat9"`  
`$$ SET neu = MIXED_SORT (alt)`

Ergebnis: `neu = "dat1'dat2'dat9'dat10"`

#### MIXED\_INDEX (var)

Die Makrofunktion `MIXED_INDEX` nimmt den Inhalt der Variablen `var` und sortiert die einzelnen Teilzeichenfolgen bzw. Zeilen in gleicher Weise wie die Makrofunktion `MIXED_SORT`; dabei wird festgehalten, an wievielter Position die Teilzeichenfolgen bzw. Zeilen ursprünglich standen. Diese Positionsangaben werden durch Apostroph getrennt als Funktionswert geliefert.

Beispiel: `$$ SET alt = "dat2'dat1'dat10'dat9"`  
`$$ SET index = MIXED_INDEX (alt)`

Ergebnis: `index = "2'1'4'3"`

`$$ SET neu = INDEX_SORT (alt, index)`

Ergebnis: `neu = "dat1'dat2'dat9'dat10"`

#### DIGIT\_SORT (var)

Die Makrofunktion `DIGIT_SORT` nimmt den Inhalt der Variablen `var` und sortiert die einzelnen Teilzeichenfolgen alphabetisch; falls `var` eine Sternvariable ist, werden die einzelnen Zeilen sortiert. Dabei werden zum Sortieren kürzere Teilzeichenfolgen bzw. Zeilen links mit Leerzeichen aufgefüllt, die nachher wieder entfernt werden. Das Ergebnis wird als Funktionswert geliefert.

Zum Sortieren wird die Standard-Sortierfolge (siehe Seite 841) zugrunde gelegt. Für Teilzeichenfolgen bzw. Zeilen, die sich auf Grund der Standard-Sortierfolge nicht unterscheiden, wird zusätzlich die Sonder-Sortierfolge (siehe Seite 841) zugrunde gelegt.

```
Beispiel: $$ SET alt = "1'234'56'7"
          $$ SET neu = DIGIT_SORT (alt)

          Ergebnis: neu = "1'7'56'234"
```

`DIGIT_INDEX (var)`

Die Makrofunktion `DIGIT_INDEX` nimmt den Inhalt der Variablen `var` und sortiert die einzelnen Teilzeichenfolgen bzw. Zeilen in gleicher Weise wie die Makrofunktion `DIGIT_SORT`; dabei wird festgehalten, an wievielter Position die Teilzeichenfolgen bzw. Zeilen ursprünglich standen. Diese Positionsangaben werden durch Apostroph getrennt als Funktionswert geliefert.

```
Beispiel: $$ SET alt = "1'234'56'7"
          $$ SET index = DIGIT_INDEX (alt)

          Ergebnis: index = "1'4'3'2"

          $$ SET neu = INDEX_SORT (alt, index)

          Ergebnis: neu = "1'7'56'234"
```

`INDEX_SORT (var, index)`

Die Makrofunktion `INDEX_SORT` nimmt den Inhalt der Variablen `var` und ordnet die Teilzeichenfolgen entsprechend den Positionsangaben an, die in der Variablen `index` enthalten sind; falls das Argument `var` eine Sternvariable ist, werden die einzelnen Zeilen entsprechend angeordnet. Das Ergebnis wird als Funktionswert geliefert.

Die Positionsangaben in der Variablen `index` müssen jeweils durch Apostroph getrennt sein. Eine Positionsangabe für eine Teilzeichenfolge bzw. Zeile entspricht der laufenden Nummer der jeweiligen Teilzeichenfolge bzw. Zeile innerhalb der Variablen `var`.

```
Beispiel: $$ SET alt = "eins'zwei'drei'vier'fünf'sechs"
          $$ SET index = "3'1'5'6'4'2"
          $$ SET neu = INDEX_SORT (alt, index)

          Ergebnis: neu = "drei'eins'fünf'sechs'vier'zwei"
```

Im folgenden Beispiel enthält die Variable `nliste` Personennamen und die Variable `oliste` parallel dazu Ortsnamen. Die Personennamen sollen alphabetisch sortiert werden. Die Makrofunktion `ALPHA_SORT` reicht dafür nicht aus, weil die Ortsnamen genau so wie die Personennamen umgeordnet werden müssen, damit Personennamen und Ortsnamen wieder parallel in den Variablen stehen.

```

Beispiel: $$ SET nliste = *
Müller, Peter
Schmid, Paula
Maier, Josef
Bauer, Petra

$$ SET oliste = *
München
Tübingen
Trier
Tübingen

$$ SET index = ALPHA_INDEX (nliste)
$$ SET nliste = INDEX_SORT (nliste, index)
$$ SET oliste = INDEX_SORT (oliste, index)

Ergebnis: nliste = *           oliste = *
           Bauer, Petra       Tübingen
           Maier, Josef       Trier
           Müller, Peter      München
           Schmid, Paula     Tübingen

```

Im folgenden Beispiel enthält die Variable `nliste` Personennamen und die Variable `oliste` parallel dazu Ortsnamen. Die Daten sollen nach Ortsnamen und innerhalb gleicher Ortsnamen nach Personennamen sortiert werden.

```

Beispiel: $$ SET oliste = *
Tübingen
Trier
München
Tübingen

$$ SET nliste = *
Schmid, Paula
Maier, Josef
Müller, Peter
Bauer, Petra

```

```

$$ SET index = ALPHA_INDEX (nliste)
$$ SET nliste = INDEX_SORT (nliste, index)
$$ SET oliste = INDEX_SORT (oliste, index)

$$ SET index = ALPHA_INDEX (oliste)
$$ SET oliste = INDEX_SORT (oliste, index)
$$ SET nliste = INDEX_SORT (nliste, index)

```

```

Ergebnis:  oliste = *      nliste = *
             München      Müller, Peter
             Trier        Maier, Josef
             Tübingen     Bauer, Petra
             Tübingen     Schmid, Paula

```

Im folgenden Beispiel enthält die Variable `wliste` französische Wörter, die alphabetisch sortiert werden sollen. Die Makrofunktion `ALPHA_SORT` reicht dafür nicht aus, weil sie die Codierungen für Akzente nicht richtig berücksichtigt. Es muss ein Sortierschlüssel erstellt werden, der diese Codierungen nicht enthält, und nach dem die Namen dann sortiert werden können. Damit aber Wörter, die sich nur durch ihre Akzente unterscheiden, zusammengeordnet werden, müssen die Wörter zuerst nach den darin enthaltenen Akzenten sortiert werden; da im Französischen die weiter rechts stehenden Akzente vorrangig sind, muss der Sortierschlüssel mit den Akzenten umgedreht werden.

```

Beispiel: $$ BUILD X_TABLE xs = *
          |?|0|%/|1|%\|2|%<|3|%:|4|

          $$ SET wliste = *
          %/e1%/eve (élève)
          mod%\ele (modèle)
          %/e1%\eve (élève)
          model%/e (modelé)
          %/e1%/eve (élève)

          $$ SET sort = EXCHANGE (wliste, xs)
          $$ SET sort = TURN (sort)
          $$ SET index = ALPHA_INDEX (sort)
          $$ SET wliste = INDEX_SORT (wliste, index)
          $$ SET sort = EXCHANGE (wliste, ":%{%}::")
          $$ SET index = ALPHA_INDEX (sort)
          $$ SET wliste = INDEX_SORT (wliste, index)

```

```

Ergebnis:  wliste = *
            %/e1%/eve (élève)
            %/e1%/eve (élève)
            %/e1%\eve (élève)
            mod%\ele (modèle)
            model%/e (modelé)

```

## Makrofunktion für nicht-numerische/mehrstufige Zähler

COUNTER (var, level)

Die Makrofunktion COUNTER nimmt den Zähler, der in der Variablen var steht, erhöht den Zähler auf den nächsten Wert und liefert das Ergebnis als Funktionswert. Der Zähler kann alphabetisch (a, b, ..., z, aa, ab, ..., az, ba, ...) oder ein- oder mehrstufig numerisch (z. B.: 1, 1.1, 1.2, ..., 2, 2.1, ...) sein.

Für alphabetische Zähler entfällt das Argument level einschließlich des davor stehenden Kommas. Enthält die Variable var eine leere Zeichenfolge oder außer Kleinbuchstaben von a bis z noch andere Zeichen, ist der Funktionswert »a«.

```
Beispiel: $$ SET zähler = ""
          $$ SET zähler = COUNTER (zähler)

          Ergebnis: zähler = "a"

          $$ SET zähler = COUNTER (zähler)

          Ergebnis: zähler = "b"
```

Für numerische Zähler muss mit dem Argument level eine Zahl oder eine Variable, die eine Zahl enthält, angegeben werden. Diese Zahl bestimmt die Stufe, auf der der Zähler erhöht wird. Hat der Zähler weniger Stufen, werden die fehlenden mit dem Wert 0 zuvor ergänzt; hat der Zähler höhere Stufen, werden diese entfernt. Steht auf der zu erhöhenden Stufe etwas anderes als eine arabische Zahl, wird dafür der Wert 0 angenommen.

```
Beispiel: $$ SET nummer = "2.4.6"
          $$ SET nummer = COUNTER (nummer, 3)

          Ergebnis: nummer = "2.4.7"

          $$ SET nummer = COUNTER (nummer, 2)

          Ergebnis: nummer = "2.5"

          $$ SET nummer = COUNTER (nummer, 3)

          Ergebnis: nummer = "2.5.1"
```

## Makrofunktion für Zufallszahlen

RANDOM\_NUMBERS (minval, maxval, anz)

Diese Makrofunktion liefert als Funktionswert anz Zufallszahlen mit einem Wert von minval bis maxval (je einschließlich), wobei keine Zahl mehrfach vorkommt. Die einzelnen Zahlen werden durch Apostroph getrennt.

Beispiel: `$$ SET lotto = RANDOM_NUMBERS (1, 49, 6)`  
`$$ SET lotto = DIGIT_SORT (lotto)`

Ergebnis: `lotto = "3'7'12'25'33'49"` (o.ä.)

## Makrofunktion zum Decodieren

Mit der folgenden Makrofunktion können Inhalte von Variablen nach bestimmten Regeln interpretiert und decodiert werden:

`DECODE (variable, modus)`

Diese Makrofunktion liefert als Funktionswert den decodierten Inhalt der als Argument angegebenen Variablen. Mit dem Argument `modus` werden die Regeln bestimmt, nach denen der Inhalt dieser Variablen interpretiert und decodiert wird.

Modi:

– `ISO8859, UTF8, UTF16, ANSI`

Mit diesen Modi können Daten vom angegebenen Code in den TUSTEP-Code umgewandelt werden.

– `UTF16/UTF8`

Mit diesem Modus können Daten von UTF16 nach UTF8 umgewandelt werden.

– `ENTITIES`

Mit diesem Modus können die in den Daten enthaltenen »Entities« in den TUSTEP-Code (z. B. »&auml;« in »ä«) umgewandelt werden.

– `ISO8859/ENTITIES, ANSI/ENTITIES`

Mit diesen Modi können Daten und die darin enthaltenen »Entities« (z. B. »&auml;«) vom angegebenen Code in den TUSTEP-Code umgewandelt werden.

– `CGI, CGI/ISO8859, CGI/UTF8`

Mit diesen Modi können in einem CGI-Makro die Daten des Anfragetextes (von der System-Variablen `QUERY_STRING`) bzw. der Standard-Eingabe (vgl. »Dateneingabe im Batch-Modus« Seite 427), die nach den CGI-Konventionen codiert sind, in Daten (Wertelisten) umgewandelt werden, die den TUSTEP-Konventionen entsprechen.

Dabei werden die Daten auch vom angegebenen Code in den TUSTEP-Code umgewandelt. Ist kein Code angegeben, wird UTF8 angenommen, falls TUSTEP durch einen WWW-Server aufgerufen wurde und mit der System-Variablen `TUSTEP_CGI` (siehe Seite 75) der Code UTF-8 angegeben ist; in allen anderen Fällen wird der Code ISO8859 angenommen.



**Beispiel:** Anfragetext: NAME=Peter+M%FCller&ORT=M%FCnchen

```

$$ FETCH query = QUERY_STRING

$$ SET liste = DECODE (query, CGI)

$$ SET such_name = GET_VALUE (liste, "NAME")
$$ SET such_ort = GET_VALUE (liste, "ORT")

```

```

Ergebnis: liste = *
           NAME=Peter Müller
           ORT=München

           such_name = "Peter Müller"
           such_ort = "München"

```

– DECIMAL0, DECIMAL1, DECIMAL2, ..., DECIMAL5

Mit diesen Modi kann eine Dezimalzahl (mit Komma oder Punkt) in eine Zahl ohne Komma bzw. Punkt umgewandelt werden. Dabei werden nach dem Komma 0, 1, 2, 3, 4 bzw. 5 Ziffern berücksichtigt.

**Beispiel:** \$\$ SET alt = "123.45"  
 \$\$ SET neu = DECODE (alt, DECIMAL3)

```
Ergebnis: neu = "123450"
```

```
$$ SET neu = DECODE (alt, DECIMAL1)
```

```
Ergebnis: neu = "1234"
```

```
$$ SET neu = DECODE (neu, DECIMAL2)
```

```
Ergebnis: neu = "123400"
```

– HEX2, HEX4, HEX6

Mit diesen Modi kann eine Zahl, die mit Hexadezimalzeichen dargestellt ist, in dieselbe Zahl umgewandelt werden, die mit arabischen Ziffern dargestellt ist. Bei HEX2 wird eine zweistellige, bei HEX4 eine vierstellige und bei HEX6 eine sechsstellige Hexadezimalzahl erwartet.

**Beispiel:** \$\$ SET alt = "007B"  
 \$\$ SET neu = DECODE (alt, HEX4)

```
Ergebnis: neu = 123
```

– HEX

Mit diesem Modus wird jeder Hexadezimal-Code (jeweils 2 Bytes) in das entsprechende Zeichen (ein Byte) umgewandelt.

Beispiel: `$$ SET alt = "313233616263"`  
`$$ SET neu = DECODE (alt, HEX)`  
 Ergebnis: `neu = "123abc"`

– BYTE

Mit diesem Modus kann ein Zeichen (ein Byte) in eine Zahl umgewandelt werden, die dem internen Zahlenwert des Zeichens entspricht.

Beispiel: `$$ SET alt = "{"`  
`$$ SET neu = DECODE (alt, BYTE)`  
 Ergebnis: `neu = 123`

– ROMAN, HEBREW

Mit diesen Modi kann eine Zahl, die mit römischen bzw. hebräischen Zahlzeichen dargestellt ist, in eine Zahl umgewandelt werden, die mit arabischen Ziffern dargestellt ist.

Beispiel: `$$ SET alt = "CXXIII"`  
`$$ SET neu = DECODE (alt, ROMAN)`  
 Ergebnis: `neu = 123`

– BASE64

Mit diesem Modus können beliebige Daten mit dem Verfahren BASE64 codiert werden. Das Ergebnis besteht nur noch aus den 64 ASCII-Zeichen A bis Z, a bis z, 0 bis 9, Pluszeichen und Schrägstrich. Eventuell werden am Ende noch Gleichheitszeichen als Füllzeichen ergänzt. E-Mail-Anhänge werden meist automatisch mit diesem Verfahren codiert/decodiert.

## Makrofunktion zum Codieren

Mit der folgenden Makrofunktion können Inhalte von Variablen nach bestimmten Regeln interpretiert und codiert werden:

`ENCODE (variable, modus)`

Diese Makrofunktion liefert als Funktionswert den codierten Inhalt der als Argument angegebenen Variablen. Mit dem Argument `modus` werden die Regeln bestimmt, nach denen der Inhalt dieser Variablen interpretiert und codiert wird.

Modi:

– ISO8859, UTF8, UTF16, ANSI

Mit diesen Modi können Daten vom TUSTEP-Code in den angegebenen Code umgewandelt werden.

– UTF8/UTF16

Mit diesem Modus können Daten von UTF8 nach UTF16 umgewandelt werden.

– ISO8859/ENTITIES

Mit diesem Modus können Daten vom TUSTEP-Code in den angegebenen Code umgewandelt werden. Dabei werden Zeichen, die im angegebenen Code nicht darstellbar sind, in Character-Entities Entities der Form `&#xXXXX;` umgewandelt (z. B. `#g+b#g-` in `&#x03B2;`).

– CGI, CGI/ISO8859, CGI/UTF8

Mit diesen Modi können in einem CGI-Makro die Daten des Anfragetextes (Werteliste) den CGI-Konventionen entsprechend codiert werden.

Dabei werden die Daten auch vom TUSTEP-Code in den angegebenen Code umgewandelt. Ist kein Code angegeben, wird UTF8 angenommen, falls TUSTEP durch einen WWW-Server aufgerufen wurde und mit der System-Variablen TUSTEP\_CGI (siehe Seite 75) der Code UTF-8 angegeben ist; in allen anderen Fällen wird der Code ISO8859 angenommen.

Beispiel: `$$ SET liste = *`  
`NAME=P. Müller`  
`ORT=München`

`$$ SET query = ENCODE (liste, CGI)`

Ergebnis: `query = "NAME=P.+M%FC1ler&ORT=M%FCnchen"`

Hinweis: Zum Erstellen und Bearbeiten von Wertelisten gibt es spezielle Makrofunktionen (siehe ab Seite 553).

– DECIMAL0, DECIMAL1, DECIMAL2, DECIMAL3, DECIMAL4, DECIMAL5

Mit diesen Modi kann eine Zahl in eine Dezimalzahl umgewandelt werden. Dabei werden 0, 1, 2, 3, 4 bzw. 5 Ziffern nach dem Dezimalpunkt eingesetzt.

Beispiel: `$$ SET alt = "12345"`  
`$$ SET neu = ENCODE (alt, DECIMAL3)`

Ergebnis: `neu = "12.345"`

`$$ SET alt = "12"`  
`$$ SET neu = ENCODE (alt, DECIMAL3)`

Ergebnis: `neu = "0.012"`

– HEX2, HEX4, HEX6

Mit diesen Modi kann eine Zahl, die mit arabischen Ziffern dargestellt ist, in dieselbe Zahl umgewandelt werden, die mit Hexadezimalzeichen dargestellt ist. Der Zahlenwert darf bei HEX2 von 0 bis 255, bei HEX4 von 0 bis 65535 und bei HEX6 von 0 bis 16777215 sein. Das Ergebnis ist bei HEX2 immer zweistellig, bei HEX4 immer vierstellig und bei HEX6 immer sechsstellig.

```
Beispiel: $$ SET alt = 123
          $$ SET neu = ENCODE (alt, HEX2)
```

```
Ergebnis: neu = "7B"
```

```
          $$ SET neu = ENCODE (alt, HEX4)
```

```
Ergebnis: neu = "007B"
```

– HEX

Mit diesem Modus wird jedes Zeichen (jedes Byte) in den entsprechenden Hexadezimal-Code umgewandelt.

```
Beispiel: $$ SET alt = "123abc"
          $$ SET neu = DECODE (alt, HEX)
```

```
Ergebnis: neu = "313233616263"
```

– BYTE

Mit diesem Modus kann eine Zahl in das entsprechende Zeichen (Byte) umgewandelt werden. Die Zahl darf einen Wert von 0 bis 255 haben.

```
Beispiel: $$ SET alt = 123
          $$ SET neu = ENCODE (alt, BYTE)
```

```
Ergebnis: neu = "{"
```

– ROMAN, HEBREW, HEBREW\_GADOL

Mit diesen Modi kann eine Zahl, die mit arabischen Ziffern dargestellt ist, in eine Zahl umgewandelt werden, die mit römischen bzw. hebräischen Zahlzeichen dargestellt ist. Die Zahl darf einen Wert von 1 bis 3999 bzw. von 1 bis 9999 haben. Bei Modus HEBREW werden nur Ziffern bis Taw (= 400), bei Modus HEBREW\_GADOL auch Ziffern größer als Taw verwendet.

```
Beispiel: $$ SET alt = 123
          $$ SET neu = ENCODE (alt, ROMAN)
```

```
Ergebnis: neu = "CXXIII"
```

– BASE64

Mit diesem Modus können Daten, die mit dem Verfahren BASE64 codiert sind, wieder in ihre ursprüngliche Form decodiert werden. E-Mail-Anhänge werden meist automatisch mit diesem Verfahren codiert/decodiert.

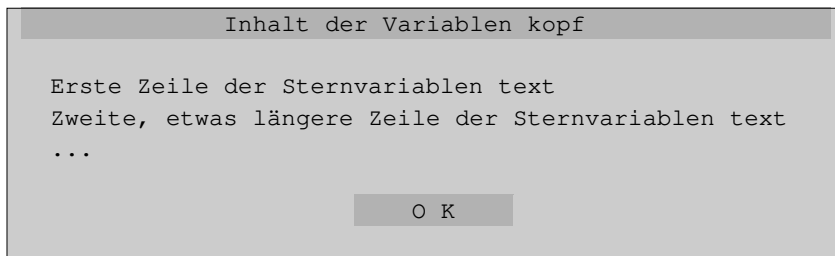
## Makrofunktion zur Anzeige einer Meldung/Anfrage

Mit den Makrofunktionen `MESSAGE` und `QUESTION` können unter Windows PopUp-Fenster angezeigt werden. Die Größe der Fenster richtet sich automatisch nach dem Umfang des Textes, der angezeigt wird.

Werden diese Makrofunktionen unter Linux und macOS aufgerufen, so wird kein PopUp-Fenster angezeigt und der Funktionswert ist immer »ERROR«.

`MESSAGE (kopf, text)`

Die Makrofunktion `MESSAGE` zeigt ein Popup-Fenster mit einer Meldung an und wartet, bis die Schaltfläche »OK« aktiviert (angeklickt), oder eine der Tasten `Return`, `Enter` oder `Escape` gedrückt wird.



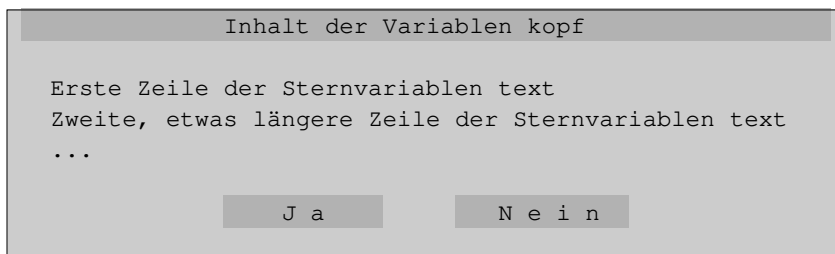
Mit der Variablen `kopf` wird der Inhalt der Kopfzeile vorgegeben.

Mit der Variablen `text` wird der Text der Meldung vorgegeben. Sie kann auch eine Sternvariable sein.

Der Funktionswert ist immer »OK«.

`QUESTION (kopf, text, default)`

Die Makrofunktion `QUESTION` zeigt ein Popup-Fenster mit einer Meldung an und wartet, bis die Schaltfläche »Ja« oder »Nein« aktiviert (angeklickt), oder eine der Tasten `Return` oder `Enter` gedrückt wird.



Mit der Variablen `kopf` wird der Inhalt der Kopfzeile vorgegeben.

Mit der Variablen `text` wird der Text der Meldung vorgegeben. Sie kann auch eine

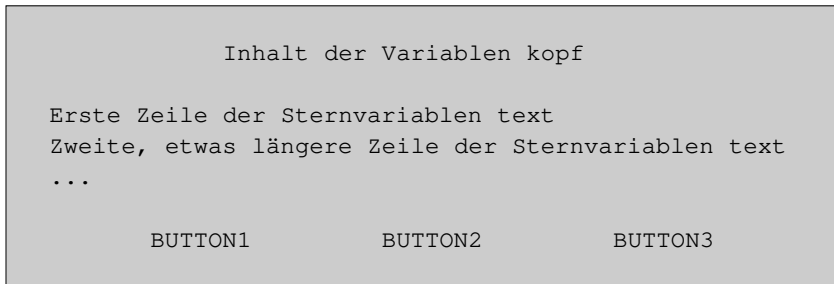
Sternvariable sein.

Für das Argument `default` muss entweder »YES« oder »NO« (ohne Anführungszeichen) angegeben werden. Damit wird der Funktionswert festgelegt, wenn eine der Tasten `Return` oder `Enter` gedrückt wird, ohne zuvor mit `TAB` auf die jeweils andere Schaltfläche zu springen. Wird die Schaltfläche »Ja« aktiviert, ist der Funktionswert »YES«, wird die Schaltflächen »Nein« aktiviert, ist der Funktionswert »NO«.

```
DISPLAY (zeile:spalte, farben, kopf, text,
        button1, button2, button3, ...)
```

Die Makrofunktion `DISPLAY` zeigt ein Fenster mit einer Meldung an und wartet, bis eine der Schaltflächen aktiviert (angeklickt), eine Funktionstaste oder eine der Tasten `Return`, `Enter` oder `Escape` gedrückt wird.

Die Meldung wird in einem Feld vom Typ `OUTPUT/SCROLL` (siehe Seite 656) angezeigt; die Schaltflächen sind Felder vom Typ `BUTTON` (siehe Seite 654).



Sowohl für die Zeilen- als auch für die Spaltenangabe gilt die gleiche Regelung wie bei der `DISPLAY`-Anweisung (siehe Seite 641).

Als Farbangabe `farben` werden 4 durch ein Minuszeichen getrennte Hexadezimal-Codes erwartet. Damit werden die Farben für das Fenster, die Kopfzeile, die Meldung und die Schaltflächen festgelegt. Die möglichen Codes können der Tabelle entnommen werden, die im Editor durch die Tastenkombination `Ctrl+F` bzw. `Strg+F` angezeigt wird.

Mit der Variablen `kopf` wird der Inhalt der Kopfzeile vorgegeben. Enthält die dafür angegebene Variable eine leere Zeichenfolge oder wird statt einer Variablen ein Minuszeichen angegeben, so wird keine Kopfzeile angezeigt.

Mit der Variablen `text` wird der Text der Meldung vorgegeben. Je Zeile wird eine Teilzeichenfolge vom Inhalt der Variablen `text` angezeigt. Die Größe des Fensters wird entsprechend angepasst. Enthält die Variable `text` mehr Teilzeichenfolgen als Zeilen möglich sind, können die angezeigten Zeilen durch entsprechende Cursor-Positionierungen nach oben bzw. unten verschoben werden. Falls die Variable `text` eine Sternvariable ist, werden die Zeilen der Sternvariablen wie die Teilzeichenfolgen einer »normalen« Variablen behandelt.

Mit den Variablen `button1`, `button2`, ... können 1 bis 9 Schaltflächen vorge-

geben werden. Die Namen der Variablen sind zugleich die Namen der Schaltflächen. Der Inhalt der Variablen wird zur Beschriftung der jeweiligen Schaltfläche verwendet.

Wird eine Schaltfläche aktiviert, wird der Name dieser Schaltfläche als Funktionswert geliefert. Eine Schaltfläche kann durch Anklicken mit der Maus aktiviert werden oder durch Drücken der Return-Taste (nicht Enter-Taste), nachdem der Cursor mit der Tabulatortaste auf die Schaltfläche positioniert worden ist.

Soll der Cursor nach dem Anzeigen des Fensters automatisch auf eine bestimmte Schaltfläche positioniert werden, muss diese Schaltfläche durch einen Stern vor dem entsprechende Variablennamen markiert werden.

Wird keine Schaltfläche aktiviert, sondern eine Funktionstaste gedrückt, wird der Name dieser Funktionstaste als Funktionswert geliefert; wird die Return-Taste gedrückt (ohne den Cursor zuvor auf eine Schaltfläche zu positionieren), ist der Funktionswert »CR«; wird die Enter-Taste gedrückt, ist der Funktionswert »ENTER«; wird die Escape-Taste oder die Tastenkombination `Ctrl+D` bzw. `Strg+D` gedrückt, ist der Funktionswert »CANCEL«.

```
Beispiel: $$ SET nix = ""
          $$ SET mld = *
          Datei xy enthält schon Daten
          Dürfen sie gelöscht werden ?
          $$ SET ok = " JA ", no = "NEIN"
          $$ SET e = DISPLAY (0:0,A0-A0-9F-E0,nix,mld,ok,no)
          Ergebnis: e = "no", falls nein angeklickt wurde
```

## Makrofunktionen zur Anzeige eines Eingabefeldes

ASK (zeile:spalte, farben, kopf, text, feld, länge, stab, button1, button2, button3, ...)

Die Makrofunktion ASK zeigt ein Fenster mit einer Meldung und einem einzeiligen Eingabefeld an und wartet, bis die Eingabe für das Eingabefeld abgeschlossen, einer der Buttons aktiviert (angeklickt), eine Funktionstaste oder eine der Tasten Return, Enter oder Escape gedrückt wird.

Die Meldung wird in einem Feld vom Typ OUTPUT/SCROLL (siehe Seite 656) angezeigt; das Eingabefeld ist ein Feld vom Typ INPUT/SHIFT (siehe Seite 648); die Schaltflächen sind Felder vom Typ BUTTON (siehe Seite 654).

```

                                Inhalt der Variablen kopf

Erste Zeile der Sternvariablen text
Zweite, etwas längere Zeile der Sternvariablen text
...

        [ Eingabefeld = Inhalt der Variablen field ]

        BUTTON1           BUTTON2           BUTTON3

```

Sowohl für die Zeilen- als auch für die Spaltenangabe gilt die gleiche Regelung wie bei der `DISPLAY`-Anweisung (siehe Seite 641).

Als Farbangabe `farben` werden 5 durch ein Minuszeichen getrennte Hexadezimal-Codes erwartet. Damit werden die Farben für das Fenster, die Kopfzeile, die Meldung, das Eingabefeld und die Schaltflächen festgelegt. Die möglichen Codes können der Tabelle entnommen werden, die im Editor durch die Tastenkombination `Ctrl+F` bzw. `Strg+F` angezeigt wird.

Mit der Variablen `kopf` wird der Inhalt der Kopfzeile vorgegeben. Enthält die dafür angegebene Variable eine leere Zeichenfolge oder wird statt einer Variablen ein Minuszeichen angegeben, so wird keine Kopfzeile angezeigt.

Mit der Variablen `text` wird der Text der Meldung vorgegeben. Je Zeile wird eine Teilzeichenfolge vom Inhalt der Variablen `text` angezeigt. Die Größe des Fensters wird entsprechend angepasst. Enthält die Variable `text` mehr Teilzeichenfolgen als Zeilen möglich sind, können die angezeigten Zeilen durch entsprechende Cursor-Positionierungen nach oben bzw. unten verschoben werden. Fall die Variable `text` eine Sternvariable ist, werden die Zeilen der Sternvariablen wie die Teilzeichenfolgen einer »normalen« Variablen behandelt.

Mit der Variablen `field` wird der Text (häufig eine leere Zeichenfolge) vorgegeben, der im Eingabefeld angezeigt wird, und zugleich die Variable bestimmt, die den eingegebenen bzw. den geänderten Text vom Eingabefeld aufnehmen soll.

Für das Argument `länge` wird eine Zahl erwartet, die die Zeichenzahl für die Länge des Eingabefeldes festlegt. Wird 0 (Null) angegeben, wird das Eingabefeld der Fensterbreite angepasst; es können beliebig viele Zeichen eingegeben werden. Können nicht alle Zeichen im Feld angezeigt werden, so werden die Zeichen im Feld verschoben.

Mit der Such-Tabelle `stab` können die Zeichen bzw. Zeichenfolgen vorgegeben werden, die im Eingabefeld erlaubt sind. Falls im Eingabefeld ein Zeichen vorkommt, das keiner Suchzeichenfolge (d. h. nur einer Ausnahmezeichenfolge oder gar keiner Zeichenfolge) aus der Such-Tabelle zugeordnet werden kann, wird die Eingabe nicht akzeptiert. Wird die Escape-Taste oder die Tastenkombination `Ctrl+D` bzw. `Strg+D` gedrückt, so unterbleibt die Überprüfung auf erlaubte Zeichen bzw. Zeichenfolgen. Wird an Stelle einer Such-Tabelle ein Minuszeichen angegeben, so werden die eingegebenen Zeichen nicht überprüft.



Mit den Variablen `button1`, `button2`, ... können 0 bis 9 Schaltflächen vorgegeben werden. Die Namen der Variablen sind zugleich die Namen der Schaltflächen. Der Inhalt der Variablen wird zur Beschriftung der jeweiligen Schaltfläche verwendet.

Wird eine Schaltfläche aktiviert, wird der Name dieser Schaltfläche als Funktionswert geliefert. Eine Schaltfläche kann durch Anklicken mit der Maus aktiviert werden oder durch Drücken der Return-Taste (nicht Enter-Taste), nachdem der Cursor mit der Tabulatortaste auf die Schaltfläche positioniert worden ist.

Soll der Cursor nach dem Anzeigen des Fensters automatisch auf eine bestimmte Schaltfläche positioniert werden, muss diese Schaltfläche durch einen Stern vor dem entsprechende Variablennamen markiert werden.

Wird keine Schaltfläche aktiviert, sondern eine Funktionstaste gedrückt, wird der Name dieser Funktionstaste als Funktionswert geliefert; wird die Return-Taste gedrückt (ohne dass der Cursor zuvor auf eine Schaltfläche positioniert wurde), ist der Funktionswert »CR«; wird die Enter-Taste gedrückt, ist der Funktionswert »ENTER«; wird die Escape-Taste oder die Tastenkombination `Ctrl+D` bzw. `Strg+D` gedrückt, ist der Funktionswert »CANCEL«.

```
Beispiel: $$ SET nix = ""
          $$ SET mld = *
          Mit welcher Nummer soll begonnen werden ?
          $$ BUILD S_TABLE zi = ":{\0}:"
          $$ SET e = ASK (0:0,A0-A0-9F-70-E0,nix,mld,n,6,zi)

          Ergebnis: n = "123", falls 123 eingegeben wurde
                   e = "CR", falls Eingabe mit CR beendet
```

```
EDIT (zeile:spalte, farben, kopf, feld, breite, höhe,
      button1, button2, button3, ...)
```

Die Makrofunktion `EDIT` zeigt ein Fenster mit einem mehrzeiligen Eingabefeld (= `textbox`) an und wartet, bis die Eingabe für das Eingabefeld abgeschlossen, eine der Schaltflächen aktiviert (angeklickt), eine Funktionstaste gedrückt oder eine der Tasten `Return`, `Enter` oder `Escape` gedrückt wird.

Das Eingabefeld ist ein Feld vom Typ `INPUT/SHIFT` (siehe Seite 648), falls es einzeilig ist, und vom Typ `INPUT/EDIT` (siehe Seite 649), falls es mehrzeilig ist; die Schaltflächen sind Felder vom Typ `BUTTON` (siehe Seite 654).

```

      Inhalt der Variablen kopf

[ Erste Zeile des Eingabefeldes           ]
[ Zweite Zeile ...                       ]
...

      BUTTON1           BUTTON2           BUTTON3

```

Sowohl für die Zeilen- als auch für die Spaltenangabe gilt die gleiche Regelung wie bei der `DISPLAY`-Anweisung (siehe Seite 641).

Als Farbangabe `farben` werden 4 durch ein Minuszeichen getrennte Hexadezimal-Codes erwartet. Damit werden die Farben für das Fenster, die Kopfzeile, das Eingabefeld und die Schaltflächen festgelegt. Die möglichen Codes können der Tabelle entnommen werden, die im Editor durch die Tastenkombination `Ctrl+F` bzw. `Strg+F` angezeigt wird.

Mit der Variablen `kopf` wird der Inhalt der Kopfzeile vorgegeben. Enthält die dafür angegebene Variable eine leere Zeichenfolge oder wird statt einer Variablen ein Minuszeichen angegeben, so wird keine Kopfzeile angezeigt.

Mit der Variablen `feld` wird der Text vorgegeben, der im Eingabefeld angezeigt wird, und zugleich die Variable bestimmt, die den eingegebenen bzw. den geänderten Text vom Eingabefeld aufnehmen soll.

Für das Argument `breite` wird eine Zahl erwartet, die die Zeichenzahl für die Breite des Eingabefeldes festlegt. Wird 0 (Null) angegeben, wird das Eingabefeld der Fensterbreite des TUSTEP-Fensters angepasst.

Für das Argument `hoehe` wird eine Zahl erwartet, die die Zeilenzahl für die Höhe des Eingabefeldes festlegt. Wird 0 (Null) angegeben, wird das Eingabefeld der Fensterhöhe des TUSTEP-Fensters angepasst.

Mit den Variablen `button1`, `button2`, ... können 0 bis 9 Schaltflächen vorgegeben werden. Die Namen der Variablen sind zugleich die Namen der Schaltflächen. Der Inhalt der Variablen wird zur Beschriftung der jeweiligen Schaltfläche verwendet.

Wird eine Schaltfläche aktiviert, wird der Name dieser Schaltfläche als Funktionswert geliefert. Eine Schaltfläche kann durch Anklicken mit der Maus aktiviert werden oder durch Drücken der Return-Taste (nicht Enter-Taste), nachdem der Cursor mit der Tabulatortaste auf die Schaltfläche positioniert worden ist.

Soll der Cursor nach dem Anzeigen des Fensters automatisch auf eine bestimmte Schaltfläche positioniert werden, muss diese Schaltfläche durch einen Stern vor dem entsprechenden Variablennamen markiert werden.

Wird keine Schaltfläche aktiviert, sondern eine Funktionstaste gedrückt, wird der Name dieser Funktionstaste als Funktionswert geliefert; wird die Return-Taste ge-

drückt (ohne dass der Cursor zuvor auf eine Schaltfläche positioniert wurde), ist der Funktionswert »CR«; wird die Enter-Taste gedrückt, ist der Funktionswert »ENTER«; wird die Escape-Taste oder die Tastenkombination `Ctrl+D` bzw. `Strg+D` gedrückt, ist der Funktionswert »CANCEL«.

Beispiel: `$$ SET mld = "Anmerkungen:"`  
`$$ SET txt = *`  
`$$ SET e = EDIT (0:0,A0-A0-70-E0,mld,txt,0,0)`

Ergebnis: `txt = Text, der eingegeben wurde`  
`e = "CR", falls Eingabe mit CR beendet`

## Makrofunktionen zur Anzeige eines Auswahlfeldes

`CLICK (zeile:spalte, farben, liste, breite, höhe, start)`

Die Makrofunktion `CLICK` zeigt ein Fenster mit einer Auswahlliste (= listbox, Listenfeld) an und wartet, bis eine der Zeilen aktiviert (angeklickt), eine Funktionstaste oder eine der Tasten `Return`, `Enter` oder `Escape` gedrückt wird.

Die Auswahlliste ist ein Feld vom Typ `SELECT/SINGLE` (siehe Seite 649).

```
Erste Zeile der Liste
Zweite, etwas längere Zeile der Liste
...
```

Sowohl für die Zeilen- als auch für die Spaltenangabe gilt die gleiche Regelung wie bei der `DISPLAY`-Anweisung (siehe Seite 641).

Als Farbangabe `farben` werden 2 durch ein Minuszeichen getrennte Hexadezimal-Codes erwartet. Damit werden die Farben für das Fenster und die Auswahlliste festgelegt. Die möglichen Codes können der Tabelle entnommen werden, die im Editor durch die Tastenkombination `Ctrl+F` bzw. `Strg+F` angezeigt wird.

Für das Argument `breite` wird eine Zahl erwartet; mit ihr wird die Breite des Feldes (in Zeichen) für die Auswahlliste festgelegt. Wird 0 (Null) angegeben, wird die Breite dem Inhalt der Variablen `liste` angepasst.

Für das Argument `höhe` wird eine Zahl erwartet; mit ihr wird die Höhe des Feldes (in Zeilen) für die Auswahlliste festgelegt. Wird 0 (Null) angegeben, wird die Höhe dem Inhalt der Variablen `liste` angepasst.

Für das optionale Argument `start` wird eine Variable erwartet. Ihr Inhalt gibt an, die wievielte Teilzeichenfolge bzw. die wievielte Zeile der Variablen `liste` als erste oben in der Auswahlliste angezeigt wird. Diese Variable kann einschließlich des davor stehenden Kommas weggelassen werden.

Wird eine Zeile der Auswahlliste aktiviert, wird die laufende Nummer dieser Zeile in der Auswahlliste als Funktionswert geliefert. Eine Zeile kann durch Anklicken mit der Maus aktiviert werden oder durch Drücken der `Return`-Taste (nicht `Enter`-

Taste), nachdem der Cursor mit den Cursor-Tasten in die entsprechende Zeile positioniert worden ist.

Wird stattdessen eine Funktionstaste gedrückt, wird der Name dieser Funktionstaste als Funktionswert geliefert; wird die Enter-Taste gedrückt, ist der Funktionswert »ENTER«; wird die Escape-Taste (oder die Tastenkombination `Ctrl+D` bzw. `Strg+D`) gedrückt, ist der Funktionswert »CANCEL«.

Beispiel: `$$ SET liste = "blau'gelb'rot'grün"`  
`$$ SET wahl = CLICK (0:0, A0-9F, liste, 0, 0)`  
 Ergebnis: `wahl = 2`, falls gelb angeklickt wurde

`CHOOSE` (zeile:spalte, farben, kopf, text, liste, breite, höhe, start, auswahl, button1, button2, button3, ...)

Die Makrofunktion `CHOOSE` zeigt ein Fenster mit einer Meldung und einer Auswahlliste (= listbox, Listenfeld) an und wartet, bis eine der Schaltflächen aktiviert (angeklickt), eine Funktionstaste oder eine der Tasten `Enter`, `Return` oder `Escape` gedrückt wird.

Die Meldung wird in einem Feld vom Typ `OUTPUT/SCROLL` (siehe Seite 656) angezeigt; die Auswahlliste ist ein Feld vom Typ `SELECT/MULTIPLE` (siehe Seite 651); die Schaltflächen sind Felder vom Typ `BUTTON` (siehe Seite 654).

In der Auswahlliste können eine oder mehrere Zeilen ausgewählt werden.

```

      Inhalt der Variablen kopf

Erste Zeile der Sternvariablen text
Zweite, etwas längere Zeile der Sternvariablen text
...

      [ Erste Zeile der Liste           ]
      [ Zweite, etwas längere Zeile der Liste ]
      [ ...                             ]

      BUTTON1           BUTTON2           BUTTON3
```

Sowohl für die Zeilen- als auch für die Spaltenangabe gilt die gleiche Regelung wie bei der `DISPLAY`-Anweisung (siehe Seite 641).

Als Farbangabe `farben` werden 5 durch ein Minuszeichen getrennte Hexadezimal-Codes erwartet. Damit werden die Farben für das Fenster, die Kopfzeile, die Meldung, die Auswahlliste und die Schaltflächen festgelegt. Die möglichen Codes können der Tabelle entnommen werden, die im Editor durch die Tastenkombination `Ctrl+F` bzw. `Strg+F` angezeigt wird.

Mit der Variablen `kopf` wird der Inhalt der Kopfzeile vorgegeben. Enthält die dafür

angegebene Variable eine leere Zeichenfolge oder wird statt einer Variablen ein Minuszeichen angegeben, so wird keine Kopfzeile angezeigt.

Mit der Variablen `text` wird der Text der Meldung vorgegeben. Je Zeile wird eine Teilzeichenfolge vom Inhalt der Variablen `text` angezeigt. Die Größe des Fensters wird entsprechend angepasst. Enthält die Variable `text` mehr Teilzeichenfolgen als Zeilen möglich sind, können die angezeigten Zeilen durch entsprechende Cursor-Positionierungen nach oben bzw. unten verschoben werden. Falls die Variable `text` eine Sternvariable ist, werden die Zeilen der Sternvariablen wie die Teilzeichenfolgen einer »normalen« Variablen behandelt.

Mit der Variablen `liste` wird die Auswahlliste vorgegeben. Je Zeile wird eine Teilzeichenfolge vom Inhalt der Variablen `list` angezeigt. Enthält die Variable `liste` mehr Teilzeichenfolgen als mit dem Argument `höhe` Zeilen angegeben sind, können die angezeigten Zeilen durch entsprechende Cursor-Positionierungen nach oben bzw. unten verschoben werden. Falls die Variable `list` eine Sternvariable ist, werden die Zeilen der Sternvariablen wie die Teilzeichenfolgen einer »normalen« Variablen behandelt.

Für das Argument `breite` wird eine Zahl erwartet; mit ihr wird die Breite des Feldes (in Zeichen) für die Auswahlliste festgelegt. Wird 0 (Null) angegeben, wird die Breite dem Inhalt der Variablen `liste` angepasst.

Für das Argument `höhe` wird eine Zahl erwartet; mit ihr wird die Höhe des Feldes (in Zeilen) für die Auswahlliste festgelegt. Wird 0 (Null) angegeben, wird die Höhe dem Inhalt der Variablen `liste` angepasst.

Für das Argument `start` wird eine Variable erwartet. Ihr Inhalt gibt an, die wievielte Teilzeichenfolge bzw. die wievielte Zeile der Variablen `liste` als erste oben in der Auswahlliste angezeigt wird.

Die Variable `auswahl` muss entweder eine leere Zeichenfolge oder die laufenden Nummern der Zeilen in der Auswahlliste enthalten, die beim Öffnen des Fensters automatisch ausgewählt werden sollen. Beim Schließen des Fensters werden in die Variable `auswahl` die laufenden Nummern der in der Auswahlliste ausgewählten Zeilen gespeichert; wurden keine Zeilen ausgewählt, so wird eine leere Zeichenfolge in diese Variable gespeichert. Eine Zeile kann durch Anklicken mit der Maus ausgewählt werden oder durch Drücken der Return-Taste (nicht Enter-Taste), nachdem der Cursor mit den Cursor-Tasten in die entsprechende Zeile positioniert worden ist; war die Zeile schon ausgewählt, wird dadurch die Auswahl wieder aufgehoben.

Mit den Variablen `button1`, `button2`, ... können 1 bis 9 Schaltflächen vorgegeben werden. Die Namen der Variablen sind zugleich die Namen der Schaltflächen. Der Inhalt der Variablen wird zur Beschriftung der jeweiligen Schaltfläche verwendet.

Wird eine Schaltfläche aktiviert, wird der Name dieser Schaltfläche als Funktionswert geliefert. Eine Schaltfläche kann durch Anklicken mit der Maus aktiviert werden oder durch Drücken der Return-Taste (nicht Enter-Taste), nachdem der Cursor mit der Tabulatortaste auf die Schaltfläche positioniert worden ist.

Soll der Cursor nach dem Anzeigen des Fensters automatisch auf eine bestimmte

Schaltfläche positioniert werden, muss diese Schaltfläche durch einen Stern vor dem entsprechende Variablennamen markiert werden.

Wird keine Schaltfläche aktiviert, sondern eine Funktionstaste gedrückt, wird der Name dieser Funktionstaste als Funktionswert geliefert; wird die Enter-Taste gedrückt, ist der Funktionswert »ENTER«; wird die Escape-Taste oder die Tastenkombination `Ctrl+D` bzw. `Strg+D` gedrückt, ist der Funktionswert »CANCEL«.

```
Beispiel: $$ SET kopf = ""
          $$ SET mld = *
           Welche Dateien
           sollen bearbeitet werden?
          $$ SET list = FILES (), start = 1, n = ""
          $$ SET goon = " O K "
          $$ SET e = CHOOSE (0:0, A0-A0-9F-9E-E0, kopf, mld,
                             list, 17, 0, start, n, goon)
          $$ SET namen = SELECT (list, #n)
```

```
Ergebnis: n = "2'4", falls 2.+4. Datei ausgewählt
           e = "goon", falls "OK" angeklickt wurde
           namen = Namen der ausgewählten Dateien
```

```
CHOOSE_ONE (zeile:spalte, farben, kopf, text,
            liste, breite, höhe, start, auswahl,
            button1, button2, ...)
```

Die Makrofunktion `CHOOSE_ONE` entspricht der zuvor beschriebenen Makrofunktion `CHOOSE` mit folgender Ausnahme: In der Auswahlliste kann immer nur eine Zeile ausgewählt sein. Wird eine andere Zeile ausgewählt, so wird die Auswahl der schon ausgewählten automatisch aufgehoben.

## Makrofunktion zur Anzeige einer Farbauswahl

```
PALETTE (zeile:spalte, farbe)
```

Die Makrofunktion `PALETTE` zeigt ein Fenster mit einer Farbmatrix an und wartet, bis eine Farbe ausgewählt wird, eine Funktionstaste oder eine der Tasten `Enter` oder `Escape` gedrückt wird. Eine Farbe kann durch Anklicken mit der Maus ausgewählt werden oder durch Drücken der Return-Taste (nicht Enter-Taste), nachdem der Cursor mit den Pfeiltasten auf die Farbe positioniert worden ist.

Die Farbmatrix entspricht derjenigen, die im Editor durch die Tastenkombination `Ctrl+F` bzw. `Strg+F` angezeigt wird. In jedem Farbfeld wird der entsprechende Hexadezimal-Code angegeben.

Sowohl für die Zeilen- als auch für die Spaltenangabe gilt die gleiche Regelung wie bei der `DISPLAY`-Anweisung (siehe Seite 641).

Als Farbangabe `farbe` wird ein Hexadezimal-Code erwartet. Damit wird die Farbe für den Fensterrahmen festgelegt. Die möglichen Codes können der Tabelle ent-

nommen werden, die im Editor durch die Tastenkombination `Ctrl+F` bzw. `Strg+F` angezeigt wird.

Wird eine Farbe ausgewählt, so wird der entsprechende Hexadezimal-Code als Funktionswert geliefert.

Wird stattdessen eine Funktionstaste gedrückt, wird der Name dieser Funktionstaste als Funktionswert geliefert; wird die Enter-Taste gedrückt, ist der Funktionswert »ENTER«; wird die Escape-Taste oder die Tastenkombination `Ctrl+D` bzw. `Strg+D` gedrückt, ist der Funktionswert »CANCEL«.

Beispiel: `$$ SET wahl = PALETTE (0:0, 70)`

Ergebnis: `wahl = "A0"`  
wenn schwarze Schrift auf grünem  
Hintergrund angeklickt wurde

## Makrofunktionen für Makrofenster

Zur Dateneingabe in Makrofenstern (siehe ab Seite 641) können Tastaturkürzel für Zeichenfolgen mit der `KEY`-Anweisung (siehe Seite 648) definiert werden, die mit der folgenden Makrofunktion abgefragt werden können.

### KEYS

Die Makrofunktion `KEYS` liefert alle definierten Tastaturkürzel; jedes Kürzel ergibt eine eigene Zeile in der Form

`x = "Zeichenfolge"`

zum Speichern in einer Sternvariablen.

Bei der Anzeige einer Zeichenfolge in einem Makrofenster stimmt die Länge der Zeichenfolge nicht immer mit der Anzahl der angezeigten Zeichen überein (weil z. B. die drei Zeichen »%/e« mit »é« dargestellt werden).

### EXTEND (var)

Die Makrofunktion `EXTEND` liefert als Funktionswert die Anzahl der Zeichen, die der Inhalt der Variablen `var` bei der Anzeige in einem Makrofenster belegt; falls `var` eine Sternvariable ist, die entsprechende Anzahl der Zeichen der längsten Zeile, die diese Variable enthält.

Beispiel: `$$ SET text = "%/et%/e"`  
`$$ SET anzahl = EXTEND (text)`

Ergebnis: `anzahl = 3`, falls `e` mit Akut darstellbar  
`anzahl = 7`, sonst

CURSOR\_POSITION (var, zeile, zeichen)

Die Makrofunktion CURSOR\_POSITION ermittelt für den in der Sternvariable `var` stehenden Text aus der Zeilen- und Zeichennummer die entsprechende Cursor-Position und liefert diese als Funktionswert. Die Variable `zeichen` kann einschließlich des davor stehenden Kommas weggelassen werden, wenn die Cursor-Position des ersten Zeichens der angegebenen Zeile ermittelt werden soll.

CURSOR\_ROW (var, pos)

Die Makrofunktion CURSOR\_ROW ermittelt für den in der Sternvariable `var` stehenden Text aus der mit der Variablen `pos` vorgegebenen Cursor-Position die entsprechende Zeilennummer und liefert diese als Funktionswert.

CURSOR\_COLUMN (var, pos)

Die Makrofunktion CURSOR\_COLUMN ermittelt für den in der Sternvariable `var` stehenden Text aus der mit der Variablen `pos` vorgegebenen Cursor-Position die entsprechende Zeichennummer (bezogen auf die entsprechende Zeile) und liefert diese als Funktionswert.

Die beiden folgenden Makrofunktionen dürfen nur innerhalb der für die einzelnen Felder eines Makrofensters angegebenen Makroanweisungen verwendet werden. Außerdem dürfen nur Namen von Feldern des eigenen Makrofensters angegeben werden.

WIDTH (feldname)

Die Makrofunktion WIDTH liefert als Funktionswert die Breite des angegebenen Feldes in Anzahl Zeichen.

Beispiel: ... [ Status ] ...

```
$$ SET zeichen = WIDTH (Status)
```

Ergebnis: zeichen = 8

HEIGHT (feldname)

Die Makrofunktion HEIGHT liefert als Funktionswert die Höhe des angegebenen Feldes in Anzahl Zeilen.

Beispiel: ... [ Liste        ] ...  
 ... [ Liste        ] ...  
 ... [ Liste        ] ...

```
$$ SET zeilen = HEIGHT (Liste)
```

Ergebnis: zeilen = 3



COLOR (feldname)

Die Makrofunktion COLOR liefert als Funktionswert die aktuelle Farbe des angegebenen Feldes.

Beispiel: ... [ Status ] ...

```
$$ SET farbe = COLOR (Status)
```

Ergebnis: farbe = "F9"

## Prüfen von Klammern

Zum Prüfen von Klammern steht die Makrofunktion `CHECK_BRACKETS` (siehe Seite 569) zur Verfügung. Sie erwartet als Argument den Namen einer Austausch-Tabelle. Für diese Austausch-Tabelle gelten Sonderregelungen:

In der Austausch-Tabelle werden paarweise Zeichenfolgen angegeben, deren jeweils erste Zeichenfolge als Klammer bzw. als zu prüfende Zeichenfolge gilt, deren Eigenschaften durch die jeweils zweite Zeichenfolge in der Form  $x_n$  festgelegt wird. Außerdem können ggf. Zeichenfolgen (z. B. `»%{1--2}{%}«` für Akzent-Codierungen, die Klammern enthalten können, oder `»#({1--4}{!})«` für entsprechend codierte Sonderzeichen) angegeben werden, die übergangen werden sollen (Ausnahmezeichenfolgen).

In der jeweils zweiten Zeichenfolge steht  $x$  für eine oder mehrere gleiche Klammern oder für einen oder mehrere Klammeraffen;  $n$  steht für eine Zahl von 1 bis 99.

Mit Klammern wird festgelegt, ob es sich um eine öffnende oder schließende Klammer handelt, und auf welcher Klammerstufe die jeweilige Klammer vorkommen darf. `» (« und ») «` bezeichnen öffnende/schließende Klammern, die nur auf Klammerstufe 1 (also nicht innerhalb eines Klammerpaares) vorkommen dürfen; `» ( (« bzw. ») ) «` bezeichnen öffnende/schließende Klammern, die nur auf Klammerstufe 2 (also nur innerhalb eines Klammerpaares) vorkommen dürfen; `» ( ( (« bzw. ») ) ) «` bezeichnen öffnende/schließende Klammern, die nur auf Klammerstufe 3 (also nur innerhalb zweier Klammerpaare) vorkommen dürfen; usw.

Mit Klammeraffen wird festgelegt, dass es sich um eine zu prüfende Zeichenfolge handelt, und auf welcher Klammerstufe sie vorkommen darf. `»@«` bezeichnet eine Zeichenfolge, die nur auf Klammerstufe 1 (also nicht innerhalb eines Klammerpaares) vorkommen darf; `»@@«` bezeichnet eine Zeichenfolge, die nur auf Klammerstufe 2 (also nur innerhalb eines Klammerpaares) vorkommen darf; `»@@@«` bezeichnet eine Zeichenfolge, die nur auf Klammerstufe 3 (also nur innerhalb zweier Klammerpaare) vorkommen darf; usw.

Bei Klammern wird mit der Zahl festgelegt, welche Klammern zusammengehören und welche Klammern innerhalb welcher Klammern vorkommen dürfen. Zusammengehörende Klammern müssen mit der gleichen Zahl gekennzeichnet sein. Eine Klammer darf nur innerhalb einer anderen Klammer vorkommen, wenn sie auf der nächst höheren Klammerstufe vorkommen darf und mit der gleichen Zahl gekennzeichnet ist.

Bei Klammeraffen (zu prüfenden Zeichenfolgen) wird mit der Zahl festgelegt, innerhalb welcher Klammern die Zeichenfolgen vorkommen dürfen. Bei Zeichenfolgen, die auf Klammerstufe 1 (nur außerhalb von Klammern) vorkommen dürfen, ist die für  $n$  angegebene Zahl bedeutungslos; sie muss aber trotzdem angegeben werden.

Gegebenenfalls können mehrere Angaben der Form  $x_n$  durch Komma getrennt angegeben werden.

Neben den oben beschriebenen normalen Klammern gibt es zusätzlich übergeordnete Klammern. Für sie gelten die gleichen Regeln. Sie werden jedoch statt mit run-

den Klammern mit eckigen gekennzeichnet. Mit den übergeordneten Klammern können Klammerbereiche gebildet werden. Mit jeder übergeordneten öffnenden Klammer wird der jeweils aktuelle Klammerbereich unterbrochen und ein neuer begonnen; mit jeder übergeordneten schließenden Klammer wird der jeweils aktuelle Klammerbereich beendet und der zuvor unterbrochene Klammerbereich fortgesetzt. Wird statt eines Klammeraffen ein senkrechter Strich angegeben, so wird mit der entsprechenden Zeichenfolge der aktuelle Klammerbereich beendet und ein neuer begonnen. In jedem Klammerbereich wird die Paarigkeit und die Verschachtelung der normalen Klammern getrennt geprüft.

Beispiele:

Prüfen von runden und eckigen Klammern, wobei die eckigen Klammern auch innerhalb von runden vorkommen dürfen:

```
... (... ) ... (... [ ... ] ... ) ... [ ... ] ...
```

Da (... [ ... ] ...) automatisch auch (...) beinhaltet, genügt es

```
... (... [ ... ] ...) ... [ ... ] ... zu deklarieren.
```

Um die Angaben für die Austausch-Tabelle zusammenzustellen, empfiehlt sich folgendes Vorgehen: Jede Klammerkombination wird in eine eigene Zeile geschrieben, dahinter die Deklarationen der einzelnen Klammern. Die erste öffnende Klammer wird mit »(« deklariert, die zweite mit »((« usw. Die letzte schließende Klammer wird mit »)« deklariert, die zweitletzte mit »))« usw. In der ersten Zeile wird bei allen Klammern die Zahl 1 angegeben, in der zweiten die Zahl 2 usw.

```
( [ ] )      ( = (1, [ = ((1, ] = ))1, ) = )1
[ ]          [ = (2, ] = )2
```

Diese Deklarationen ergeben zusammengefasst:

```
' (' (1' [' ( (1, (2' ]' ) ) 1, ) 2' )' ) 1'
```

oder

```
' (' (1' )' ) 1' [' (2, ((1' ]' ) 2, ) ) 1'
```

Falls mit der MODE-Anweisung (siehe ab Seite 413) der Modus »{ }« eingestellt wurde, muss in der Austausch-Tabelle vor den eckigen Klammern jeweils noch »\« eingefügt werden.

Prüfen von runden und eckigen Klammern und darin erlaubten bzw. nicht erlaubten Zeichenfolgen: Die eckigen Klammern dürfen auch innerhalb von runden vorkommen. Der senkrechte Strich darf nur innerhalb von eckigen Klammern vorkommen. Der Stern darf nur innerhalb von runden Klammern vorkommen, nicht aber innerhalb von eckigen Klammern.

```
... (...*...) ... (...*... [ ... | ... ] ...*...) ... [ ... | ... ] ...
```

Da (... [ ... ] ...) automatisch auch (...) beinhaltet

und ...\*... [ ... ] auch [ ... ] ...\*... beinhaltet, genügt es

```
... (...*... [ ... | ... ] ...) ... [ ... | ... ] ... zu deklarieren.
```

Mit den Klammern wird wie im vorangehenden Beispiel beschrieben verfahren. Die zu prüfenden Zeichenfolgen werden mit so vielen Klammeraffen deklariert, wie eine öffnende Klammer an derselben Stelle mit Klammern deklariert würde.

```
( * [ | ] ) ( = (1, * = @@1, [ = ((1, | = @@@1, ] = ))1, ) = )1
[ | ]       [ = (2, | = @@2, ] = )2
```

Diese Deklarationen ergeben zusammengefasst:

```
' (' (1 '* '@@1' [ '( (1, (2 '| '@@@1, @@2' ) ) ) 1, ) 2' ) ) 1'
```

oder

```
' (' (1' ) ) 1' [ '(2, ((1' ) ) 2, ) ) 1 '* '@@1' | '@@2, @@@1'
```

Falls mit der MODE-Anweisung (siehe ab Seite 413) der Modus »{ }« eingestellt wurde, muss in der Austausch-Tabelle vor den eckigen Klammern und dem Stern jeweils noch »\« eingefügt werden.

Prüfen von »Klammern« in eigenständigen Klammerbereichen: Die Zeichenfolge #/+ gilt als öffnende Klammer, die Zeichenfolge #- als schließende. Mit @F+ wird ein Klammerbereich unterbrochen und ein neuer begonnen, mit @F- endet ein Klammerbereich und der zuvor unterbrochene wird fortgesetzt. Mit &! wird ein Klammerbereich beendet und ein neuer begonnen, sowohl innerhalb von @F+ und @F- als auch außerhalb davon.

Wenn @F+ für Fußnotenanzug, @F- für Fußnotenende und &! für neuer Abschnitt steht, müssen die Codierungen für Anfang (#+) und Ende (#-) der kursiven Schrift sowohl innerhalb der Fußnoten als auch innerhalb jedes Abschnitts paarig sein.

```
... #/+ ... @F+ ... &! ... #/+ ... #- ... @F- ... #- ... &! ... #/+ ... #- ...
```

Mit den übergeordneten Klammern für die Kennzeichnung der Klammerbereiche wird genauso verfahren wie mit den normalen Klammern, mit den Zeichenfolgen zur Unterteilung in Klammerbereiche genauso wie mit den zu prüfenden Zeichenfolgen.

```
#/+ #-          #/+ = (1, #- = )1
@F+ &! @F- &!   @F+ = [2, &! = ||2, @F- = ]2, &! = |2
```

Diese Deklarationen ergeben zusammengefasst:

```
' #/+ ' (1 '#/-' ) 1 '@F+' [2 '&! ' ||2, |2 '@F-' ] 2'
```

## Dateizugriffe – Allgemeines

Zum Zugriff auf Daten in Dateien stehen neben den in den folgenden Kapiteln beschriebenen Makroanweisungen vier schon weiter oben beschriebene Möglichkeiten zur Verfügung:

Mit der Makroanweisung `FILE` (siehe Seite 428 und Seite 430) und der Makrofunktion `WRITE` (siehe Seite 495) können Daten in eine Datei geschrieben werden.

Mit den Makrofunktionen `FILE` (siehe Seite 494) und `READ` (siehe Seite 496) können Daten von einer Datei eingelesen werden.

Der Zugriff auf Daten in Dateien oder Variablen mit den in den folgenden Kapiteln beschriebenen Makroanweisungen muss mit einer `ACCESS`-Anweisung begonnen und mit einer `ENDACCESS`-Anweisung beendet werden. Die `ACCESS`-Anweisung hat folgende Grundform:

```
$$ ACCESS daten: modus/option "dateiname" nummer, text, anzahl
```

Die genaue Form ist vom Zugriffsmodus abhängig und ist im jeweiligen Kapitel beschrieben. Die Makroanweisung

```
$$ ENDACCESS daten
```

beendet den Dateizugriff ohne eine Meldung auszugeben;

```
$$ ENDACCESS/PRINT daten
```

beendet den Dateizugriff und gibt eine Meldung mit Angaben zu den eingelesenen bzw. ausgegebenen Daten ins Ablaufprotokoll aus.

Als Zugriffsmodus sind drei alternative Angaben vorgesehen:

- `READ`, falls von der Datei nur gelesen werden soll,
- `UPDATE`, falls gelesen und geschrieben werden soll, und
- `WRITE`, falls nur ans Ende der Datei geschrieben werden soll.

Der Zugriffsmodus muss durch Angabe von Optionen noch genauer bestimmt werden. Insbesondere muss angegeben werden in welcher Weise die Textportionen bestimmt werden, die mit einer `READ`- bzw. `WRITE`-Anweisung gelesen bzw. geschrieben werden:

- `RECORDS`: Jeder Satz entspricht einer Textportion (siehe Seite 603).
- `LINES`: Jede Zeile (= alle aufeinander folgenden Sätze mit der gleichen Seiten- und Zeilennummer) entsprechen einer Textportion (siehe Seite 603).
- `PAGES`: Jede Seite (= alle aufeinander folgenden Sätze mit der gleichen Seitennummer) entsprechen einer Textportion (siehe Seite 603).
- `STREAM`: Daten werden entweder durch frei wählbare Zeichenfolgen, die vom Benutzer als »Anfangskennung«, als »Trennzeichenfolge« oder als »Endekennung« definiert wurden (siehe Seite 610), oder durch XML-Tags (siehe Seite 617) in Textportion unterteilt.
- `STRUCTURES`: Eine Textportion entspricht jeweils einer vom Benutzer vorgegebenen Datenstruktur (siehe Seite 627).

Wenn die Daten durch frei wählbare Zeichenfolgen oder XML-Tags in Portionen

eingeteilt werden (Option `STREAM`), können mit der `READ`-Anweisung auch Daten statt aus einer Datei aus einer Variablen gelesen werden und mit der `WRITE`-Anweisung statt in eine Datei in eine Variable geschrieben werden. In diesem Fall muss nach den Optionen statt eines Dateinamens der Name der entsprechenden Variablen angegeben werden.

Wird nach den Optionen statt eines Dateinamens in Anführungszeichen eine Variable angegeben, muss eine der beiden folgenden Optionen an erster Stelle angegeben werden:

- `FILE`, falls die Variable den Namen der Datei enthält, aus der gelesen bzw. in die geschrieben werden soll,
- `VARIABLE`, falls aus der angegebene Variablen gelesen bzw. in diese Variable geschrieben werden soll.

Bei den Modi `WRITE` und `UPDATE` kann zusätzlich die Option `ERASE` angegeben werden; in diesem Fall werden bei der Ausführung der `ACCESS`-Anweisung alle Daten in der Datei bzw. Variablen gelöscht. Die Option `ERASE` muss als erste Option bzw. unmittelbar nach der Option `FILE` oder `VARIABLE` angegeben werden.

Beim Zugriff auf Daten mit den in den folgenden Kapiteln beschriebenen Makroanweisungen müssen in `TUSTEP`-Dateien die Satznummern alle aufsteigend und voneinander verschieden sein, da die Daten über die Satznummern angesprochen werden; durch Angabe der Option `RAW` (siehe Seite 603) können jedoch auch Dateien sequentiell gelesen und beschrieben werden, bei denen die Satznummern diese Bedingung nicht erfüllen.

Unabhängig vom Zugriffsmodus gilt: Wenn bei der Ausführung einer `ACCESS`-Anweisung ein sofortiger Zugriff auf die Datei nicht möglich ist, kann durch eine optionale Angabe in der `ACCESS`-Anweisung gewartet werden, bis der Zugriff möglich ist. Dies ist nachfolgend in diesem Kapitel beschrieben.

Bei den in den folgenden Kapiteln beschriebenen Makroanweisungen `FIND` und `COUNT` ist die Angabe von »Suchbedingungen« erforderlich. Mit den Suchbedingungen wird festgelegt, welche Daten gesucht bzw. gezählt werden sollen. Die Suchbedingungen sind ebenfalls nachfolgend in diesem Kapitel beschrieben.

## Warten bis Dateizugriff möglich

Lesender Zugriff (Modus `READ`) auf eine Datei ist nur möglich, wenn kein Programm diese Datei gerade mit schreibendem Zugriff belegt hat; schreibender Zugriff (Modi `UPDATE` und `WRITE`) ist nur möglich, wenn kein Programm diese Datei gerade mit lesendem oder schreibendem Zugriff belegt hat. Ist der gewünschte Zugriff nicht möglich, wird die Bearbeitung des Makros abgebrochen.

Soll in diesem Fall eine bestimmte Zeit gewartet werden, bis die Datei wieder frei ist, kann dies durch Angabe eines Zeitlimits in der `ACCESS`-Anweisung erreicht werden. Das Zeitlimit gibt an, wieoft maximal versucht werden soll, auf die Datei zuzugreifen. Dabei wird vor jedem erneuten Versuch 1 Sekunde (bei großer Rechnerauslastung betriebssystembedingt etwas länger) gewartet. Das Zeitlimit muss nach der Angabe `modus` durch einen Schrägstrich getrennt angegeben werden.

Wenn das Zeitlimit erreicht wird, der gewünschte Zugriff also nicht möglich ist, erfolgt eine Fehlermeldung und das Fehlerflag wird gesetzt. Die Ausgabe der Fehlermeldung und das Setzen des Fehlerflags kann durch die Angabe `QUIET` unterdrückt werden; `QUIET` muss hinter dem Zeitlimit durch einen Schrägstrich getrennt angegeben werden.

Ob das Zeitlimit erreicht worden ist, kann in einer nachfolgenden `IF`-Anweisung mit der Bedingung `WAIT` abgefragt werden. Die Bedingung ist erfüllt, wenn noch länger gewartet werden müsste, und ist nicht erfüllt, wenn der gewünschte Zugriff möglich ist.

Beispiel:

```

$$ ACCESS daten: WRITE/STRUCTURES/10/QUIET "datei" num, str
$$ IF (WAIT) THEN
$$   - Zugriff jetzt nicht möglich
$$   ...
$$ ELSE
$$   - Zugriff gestartet
$$   ...
$$   WRITE/NEXT daten
$$   ...
$$   ENDACCESS daten
$$   - Zugriff beendet
$$ ENDIF

```

## Suchbedingungen

Eine Suchbedingung für `FIND`- und `COUNT`-Anweisungen hat folgende Form:

```
(anf,end,trenn,tausch; pos1,pos2,pos3; neg1,neg2,neg3; wdh)
```

Bestandteile einer Suchbedingung:

- Klammer auf
- `anf`: Name einer Such-Tabelle; die in der Tabelle angegebenen Zeichenfolgen kennzeichnen innerhalb einer Einheit den Anfang des zu prüfenden Textteils. Dieser beginnt nach der gefundenen Anfangskennung bzw. ist leer, falls keine gefunden wird. Soll der zu prüfende Textteil immer am Anfang einer Einheit beginnen, kann an Stelle der Such-Tabelle ein Minuszeichen angegeben werden.
- Komma
- `end`: Name einer Such-Tabelle; die in der Tabelle angegebenen Zeichenfolgen kennzeichnen innerhalb einer Einheit das Ende des zu prüfenden Textteils. Dieser endet vor der gefundenen Endekennung bzw. am Ende der Einheit, falls keine gefunden wird. Die Suche nach der Endekennung beginnt am Anfang des zu prüfenden Textteils. Soll der zu prüfende Textteil immer am Ende einer Einheit enden, kann an Stelle der Such-Tabelle ein Minuszeichen angegeben werden.
- Komma

- `trenn`: Name einer Such-Tabelle; die in der Tabelle angegebenen Zeichenfolgen dienen innerhalb des mit »anf« und »end« ausgewählten Textteils als Trennzeichen, falls der ausgewählte Textteil mehrere zu prüfende Textteile enthält. Soll der mit »anf« und »end« ausgewählte Textteil nicht weiter unterteilt werden, kann an Stelle des Namens der Such-Tabelle ein Minuszeichen angegeben werden.

- Komma

- `tausch`: Name einer Austausch-Tabelle; die in der Tabelle angegebenen Suchzeichenfolgen werden in jedem zu prüfenden Textteil durch die dazugehörigen Ersatzzeichenfolgen ersetzt. Sollen vor dem Prüfen keine Zeichenfolgen ersetzt werden, kann an Stelle des Namens der Austausch-Tabelle ein Minuszeichen angegeben werden.

Falls für »tausch« kein Tabellename angegeben wird, kann das Minuszeichen und das vorangehende Komma weggelassen werden. Falls für »trenn« ebenfalls kein Tabellename angegeben wird, kann auch das für »trenn« vorgesehene Minuszeichen und das vorangehende Komma weggelassen werden.

- Strichpunkt

- `pos1`: Minuszeichen oder Name einer Recherchier-Tabelle; es wird geprüft, ob (irgend) ein zu prüfender Textteil der Einheit mindestens eine (Tabelle mit Option OR) bzw. alle (Tabelle mit Option AND) in der Tabelle angegebenen Zeichenfolgen enthält.

Hinweis: Die Option muss beim Definieren der Recherchier-Tabelle mit der BUILD-Anweisung angegeben werden.

- Komma

- `pos2`: Minuszeichen oder Name einer Such-Tabelle; es wird geprüft, ob (irgend) ein zu prüfender Textteil der Einheit eine oder mehrere von den in der Tabelle angegebenen Zeichenfolgen enthält.

- Komma

- `pos3`: Minuszeichen oder Name einer Such-Tabelle; es wird geprüft, ob (irgend) ein zu prüfender Textteil der Einheit alle in der Tabelle angegebenen Zeichenfolgen enthält.

Falls für »pos3« kein Tabellename angegeben wird, kann das Minuszeichen und das vorangehende Komma weggelassen werden. Falls für »pos2« ebenfalls kein Tabellename angegeben wird, genügt die Angabe für `pos1`.

- Strichpunkt

- `neg1`: Minuszeichen oder Name einer Recherchier-Tabelle; es wird geprüft, ob (irgend) ein zu prüfender Textteil der Einheit mindestens eine (Tabelle mit Option OR) bzw. alle (Tabelle mit Option AND) in der Tabelle angegebenen Zeichenfolgen enthält.

Hinweis: Die Option muss beim Definieren der Recherchier-Tabelle mit der BUILD-Anweisung angegeben werden.

- Komma



– `neg2`: Minuszeichen oder Name einer Such-Tabelle; es wird geprüft, ob (irgend) ein zu prüfender Textteil der Einheit eine oder mehrere von den in der Tabelle angegebenen Zeichenfolgen enthält.

– Komma

– `neg3`: Minuszeichen oder Name einer Such-Tabelle; es wird geprüft, ob (irgend) ein zu prüfender Textteil der Einheit alle in der Tabelle angegebenen Zeichenfolgen enthält.

Falls für »`neg3`« keine Tabellenname angegeben wird, kann das Minuszeichen und das vorangehende Komma weggelassen werden. Falls für »`neg2`« ebenfalls kein Tabellenname angegeben wird, genügt die Angabe für `neg1`.

– Strichpunkt

– `wdh`: Minuszeichen, Pluszeichen oder Stern.

Minuszeichen: Mit »`anf`« und »`end`« wird nur der erste entsprechend gekennzeichnete Textteil einer Einheit ausgewählt und geprüft.

Pluszeichen: Mit »`anf`« und »`end`« werden nacheinander alle entsprechend gekennzeichnete Textteile einer Einheit ausgewählt und einzeln geprüft.

Stern: Mit »`anf`« und »`end`« werden alle entsprechend gekennzeichneten Textteile einer Einheit ausgewählt, zusammengefasst und dann geprüft. Beim Zusammenfassen wird zwischen den einzelnen ausgewählten Textteilen ein Leerzeichen ergänzt.

– Klammer zu

Die in Suchbedingungen angegebenen Recherchier-, Such- und Austausch-Tabellen müssen zuvor mit der `BUILD`-Anweisung definiert worden sein.

Eine Suchbedingung ist für eine Einheit erfüllt, wenn

– nur für `pos1` und/oder `pos2` und/oder `pos3` (nicht aber für `neg1` oder `neg2` oder `neg3`) Recherchier- bzw. Such-Tabellen angegeben sind und mindestens eine der angegebenen Bedingungen in einem zu prüfenden Textteil erfüllt ist.

– nur für `neg1` und/oder `neg2` und/oder `neg3` (nicht aber für `pos1` oder `pos2` oder `pos3`) Recherchier- bzw. Such-Tabellen angegeben sind und keine der angegebenen Bedingungen in einem zu prüfenden Textteil erfüllt ist.

– sowohl für `pos1` und/oder `pos2` und/oder `pos3` als auch für `neg1` und/oder `neg2` und/oder `neg3` Recherchier- bzw. Such-Tabellen angegeben sind und mindestens eine der Bedingungen `pos1`, `pos2`, `pos3` und keine der Bedingungen `neg1`, `neg2`, `neg3` in einem zu prüfenden Textteil erfüllt ist.

An Stelle einer einzigen Suchbedingung können auch mehrere Suchbedingungen, die jeweils durch einen der folgenden logischen Operatoren miteinander verbunden sind, angegeben werden.

.AND. beide Bedingungen müssen erfüllt sein

.OR. mindestens eine Bedingung muss erfüllt sein

Werden mehr als zwei Bedingungen mit logischen Operatoren verbunden, so wer-

den die Operatoren in der Reihenfolge berücksichtigt, in der sie oben aufgeführt sind. Von dieser Regelung kann abgewichen werden, indem die Priorität durch Setzen von Klammern festgelegt wird. Dazu können z. Z. jedoch nur mit `.OR.` verbundene Bedingungen eingeklammert werden.

Beispiel: `((bed1) .AND. ((bed2) .OR. (bed3)) .AND. (bed4))`

Für die logischen Operatoren

`.AND.` und `.OR.`

können auch die Formen

`&&` und `||`

verwendet werden.

## Dateizugriffe – Sätze, Zeilen, Seiten

Diese Dateizugriffe erlauben das Beschreiben, Lesen und Durchsuchen von TUSTEP- und Fremd-Dateien.

Eine Texteinheit entspricht jeweils einem Satz. Bei TUSTEP-Dateien kann bei Angabe der entsprechenden Option als Texteinheit auch eine Zeile oder eine Seite gewählt werden:

Eine »Zeile« umfasst alle aufeinander folgenden Sätze, die die gleiche Seiten- und Zeilennummer (ohne Unterscheidungsnummer) haben.

Eine »Seite« umfasst alle aufeinander folgenden Sätze, die die gleiche Seitennummer haben.

### Definieren des Dateizugriffs

Der Zugriff auf Daten in Dateien mit den in diesem Kapitel beschriebenen Makroanweisungen muss mit der folgenden ACCESS-Anweisung begonnen und mit einer ENDACCESS-Anweisung (siehe Seite 597) beendet werden.

```
$$ ACCESS daten: modus/option "dateiname" nmmr, text, anzahl
```

Bestandteile der ACCESS-Anweisung

- `daten`: Frei wählbarer Name für die Daten; er wird bei allen anderen Anweisungen für den Dateizugriff an Stelle eines Dateinamens verwendet, um anzugeben, auf welche Datei zugegriffen werden soll.
- Doppelpunkt
- `modus/option`: READ/RECORDS, READ/RAW, oder READ/RAW/REVERSE, falls von der nachfolgend angegebenen Datei nur gelesen werden soll; UPDATE/RECORDS, falls gelesen und geschrieben werden soll; WRITE/RECORDS oder WRITE/RAW, falls nur ans Ende der Datei geschrieben werden soll.

Mit der Option RAW erfolgt der Zugriff wie mit der Option RECORDS satzweise, jedoch müssen die Satznummern in der Datei nicht aufsteigend und nicht voneinander verschieden sein; beim Schreiben erhalten die Sätze jeweils die aktuelle Satznummer (siehe Angabe `nmmr`), beim Lesen wird die aktuelle Satznummer nicht ausgewertet, sondern nur dem gelesenen Satz entsprechend gesetzt. Wird hinter der Option RAW noch die Option REVERSE angegeben, so werden die Sätze beginnend mit dem letzten Satz in umgekehrter Reihenfolge gelesen. Wenn die Option RAW angegeben ist, sind bei den im Folgenden beschriebenen Anweisungen die Optionen PREVIOUS, REVERSE, UPDATE und CHECK nicht erlaubt; die Anweisung (!) ERASE ist ebenfalls nicht erlaubt.

Der Modus UPDATE ist nur für TUSTEP-Dateien vorgesehen.

Bei den Modi UPDATE und WRITE kann zusätzlich die Option ERASE angegeben werden; in diesem Fall werden bei der Ausführung der ACCESS-Anweisung alle Daten in der Datei gelöscht. Die Option ERASE muss als erste Option bzw. unmittelbar nach der Option FILE (vgl. Seite 598) angegeben werden.

Beim Modus `WRITE` kann zusätzlich die Option `PROGRAM` angegeben werden; in diesem Fall werden beim Schreiben mit der Anweisung `WRITE/NEXT` die Sätze nicht im Textmodus, sondern im Programmmodus nummeriert.

Bei `TUSTEP`-Dateien kann an Stelle der Option `RECORDS` die Option `LINES` bzw. die Option `PAGES` angegeben werden; der Dateizugriff erfolgt dann nicht satzweise, sondern zeilen- bzw. seitenweise und die einzelnen Sätze werden in einer Sternvariablen gespeichert (`READ`) bzw. erwartet (`WRITE` und `UPDATE`).

Bei Fremd-Dateien muss als zusätzliche Option der Code für die Daten angegeben werden; vorgesehen sind `ISO` (= `ISO-8859-1`), `UTF8` und `UTF16`.

Am Ende der Angabe `option` kann durch einen Schrägstrich getrennt ein Zeitlimit angegeben werden. Die Bedeutung des Zeitlimits ist unter »Warten bis Dateizugriff möglich« auf Seite 598 beschrieben.

- `dateiname`: Name der Datei; er muss zwischen Anführungszeichen angegeben werden. Wird eine Variable angegeben, die den Dateinamen enthält, muss als erste Option `FILE` angegeben werden (vgl. Seite 598).
- `nmmr`: Bei Fremd-Dateien wird der Name einer Variablen erwartet, die die »laufende Nummer« enthält. Bei `TUSTEP`-Dateien werden mit Modus `RECORDS` und `RAW` drei Variablennamen erwartet, die durch Punkt und Schrägstrich getrennt sind (z. B. `snr.znr/unr`); bei Modus `LINES` werden nur die ersten beiden Variablen (z. B. `snr.znr`), bei Modus `PAGES` wird nur die erste Variable (z. B. `snr`) erwartet. Die Variablen enthalten die »aktuelle Satznummer« (Seiten-, Zeilen- und Unterscheidungsnummer) des ersten (und evtl. einzigen) Satzes der Texteinheit. Diese Variablen werden von den nachfolgend beschriebenen Makroanweisungen ausgewertet oder evtl. geändert.
- Komma
- `text`: Name der Variablen, die den Text enthält. Falls eine der Optionen `LINES` oder `PAGES` angegeben ist, Name einer Sternvariablen, die die entsprechenden Sätze enthält.
- Komma
- `anzahl`: Name der Variablen, die bei der Makroanweisung `FIND/SKIP` die Anzahl der zu übergangenen Texteinheiten und bei der Makroanweisung `COUNT` die Anzahl der gefundenen Texteinheiten angibt. Wird keine der beiden Makroanweisungen verwendet, kann die Variable `anzahl` einschließlich des davor stehenden Kommas weggelassen werden.

Bei der Ausführung der `ACCESS`-Anweisung wird der Inhalt der in der Anweisung angegebenen Variablen auf Null gesetzt bzw. gelöscht.

## Schreiben einer Texteinheit

Beim Schreiben einer Texteinheit wird der Inhalt einer Variablen in die Datei ausgegeben. Die Variable kann in der `WRITE`-Anweisung nach dem Datennamen explizit angegeben werden; andernfalls wird die in der `ACCESS`-Anweisung für den Text vorgesehene Variable verwendet.

```
$$ WRITE daten
```

Nur für TUSTEP-Dateien: Schreibt eine Texteinheit mit der aktuellen Satznummer in die Datei. War in der `ACCESS`-Anweisung die Option `UPDATE` angegeben und gibt es in der Datei schon eine Texteinheit mit dieser Satznummer, erfolgt eine Fehlermeldung und das Fehlerflag wird gesetzt. War in der `ACCESS`-Anweisung die Option `WRITE` angegeben und ist die aktuelle Satznummer nicht höher als die Satznummer des letzten Satzes der Datei, erfolgt ebenfalls eine Fehlermeldung und das Fehlerflag wird gesetzt.

```
$$ WRITE/ADJUST daten
```

Wie `WRITE`, jedoch wird keine Fehlermeldung ausgegeben, falls in der `ACCESS`-Anweisung die Option `WRITE` angegeben war und die aktuelle Satznummer nicht höher als die des letzten Satzes ist, sondern die Satznummer automatisch erhöht und ein neuer Satz begonnen. Um wieviel die Satznummer erhöht wird, kann mit der `MODIFY`-Anweisung (siehe unten) festgelegt werden.

```
$$ WRITE/NEXT daten
```

Bei TUSTEP-Dateien: Wie `WRITE`, jedoch wird die aktuelle Satznummer zuvor erhöht. Um wieviel die Satznummer erhöht wird, kann mit der `MODIFY`-Anweisung (siehe unten) festgelegt werden. Hat die aktuelle Satznummer den Wert Null und stehen in der Datei schon Daten, wird zuvor die Satznummer der letzten Texteinheit als aktuelle Satznummer übernommen.

Bei Fremd-Dateien: Schreibt eine Texteinheit in die Datei; die laufende Nummer wird um 1 erhöht.

Wenn die Variable für den Text eine Sternvariable ist, wird für jede Zeile der Variablen ein neuer Satz begonnen. Um wieviel die Satznummer dabei für jede Zeile erhöht wird, kann mit der `MODIFY`-Anweisung (siehe unten) festgelegt werden.

```
$$ WRITE/UPDATE daten
```

Nur für TUSTEP-Dateien: Wie `WRITE`, jedoch muss die Texteinheit schon vorhanden sein; sie wird überschrieben.

```
$$ WRITE/CLEAR daten
```

Bei allen `WRITE`-Anweisungen kann `CLEAR` als letzte Option angegeben werden. Sie bewirkt, dass der Inhalt der Variablen, deren Inhalt in die Datei ausgegeben wurde, anschließend automatisch gelöscht wird.

```
$$ MODIFY ACCESS daten type schrittweite
```

Mit dieser Anweisung kann festgelegt werden, um wieviel die Satznummer ggf. erhöht wird. Ist für `type` `ADJUST` oder `NEXT` angegeben, so gilt die Schrittweite für die Anweisungen `WRITE/ADJUST` bzw. `WRITE/NEXT`; ist für `type *` angegeben, so gilt

die Schrittweite für die zweite und alle folgenden Zeilen, wenn eine Sternvariable in die Datei geschrieben wird.

Voreingestellt ist als Schrittweite 1000, d. h. dass die Zeilennummer um 1 erhöht wird und die Unterscheidungsnummer auf Null gesetzt wird. Ist eine kleinere Schrittweite angegeben, wird die Unterscheidungsnummer um den angegebenen Wert erhöht. Falls sich beim Erhöhen der Zeilennummer ein Wert größer als 999 ergibt, wird die Seitennummer um 1 erhöht und die Zeilennummer und die Unterscheidungsnummer auf Null gesetzt; falls sich beim Erhöhen der Unterscheidungsnummer ein Wert größer als 999 ergibt, wird die Zeilennummer um 1 erhöht und die Unterscheidungsnummer zurückgesetzt.

## Lesen einer Texteinheit

Beim Lesen einer Texteinheit wird der Text der gelesenen Texteinheit in eine Variable gespeichert. Die Variable kann in der READ-Anweisung nach dem Datennamen explizit angegeben werden; andernfalls wird die in der ACCESS-Anweisung für den Text vorgesehene Variable verwendet. Außerdem wird bei Fremd-Dateien die laufende Nummer, bei TUSTEP-Dateien die Satznummer der gelesenen Texteinheit in die in der ACCESS-Anweisung dafür vorgesehenen Variablen gespeichert.

```
$$ READ daten
```

Nur für TUSTEP-Dateien: Liest die Texteinheit mit der aktuellen Satznummer von der Datei. Gibt es in der Datei keine Texteinheit mit dieser Satznummer, erfolgt eine Fehlermeldung und das Fehlerflag wird gesetzt.

```
$$ READ/NEXT daten
```

Bei TUSTEP-Dateien: Wie READ, jedoch wird die Texteinheit gelesen, die in der Datei nach der mit der aktuellen Satznummer folgt. Gibt es in der Datei keine nachfolgende Texteinheit, wird eine Null bzw. eine leere Zeichenfolge in die für laufende Nummer/Satznummer bzw. Text vorgesehenen Variablen (vgl. ACCESS-Anweisung) gespeichert. War in der ACCESS-Anweisung die Option RAW angegeben, wird ohne Berücksichtigung der aktuellen Satznummer die nächste Texteinheit (d. i. der nächste Satz) gelesen.

Bei Fremd-Dateien: Liest die nächste Texteinheit von der Datei. Falls die aktuelle laufende Nummer Null ist, ist die erste Texteinheit die nächste. Gibt es in der Datei keine nächste Texteinheit, wird eine Null bzw. eine leere Zeichenfolge in die für laufende Nummer bzw. Text vorgesehenen Variablen (vgl. ACCESS-Anweisung) gespeichert.

Ob noch eine Texteinheit vorhanden war, kann auch mit einer nachfolgenden IF-Anweisung mit der Bedingung EOF (siehe Seite 476) abgefragt werden.

```
$$ READ/PREVIOUS daten
```

Nur für TUSTEP-Dateien: Wie READ, jedoch wird die Texteinheit gelesen, die in der Datei vor der mit der aktuellen Satznummer steht. Gibt es in der Datei keine vorangehende Texteinheit, wird eine Null bzw. eine leere Zeichenfolge in die für laufende Nummer/Satznummer bzw. Text vorgesehenen Variablen (vgl. ACCESS-Anweisung) gespeichert.

Ob noch eine Texteinheit vorhanden war, kann auch mit einer nachfolgenden IF-Anweisung mit der Bedingung EOF (siehe Seite 476) abgefragt werden.

```
$$ READ/IGNORE daten
```

Nur für TUSTEP-Dateien: Liest die Texteinheit, die auf die zuletzt gelesene folgt (unabhängig vom Inhalt der für laufende Nummer/Satznummer vorgesehenen Variablen); wurde noch keine Texteinheit gelesen, so wird die erste gelesen. Gibt es in der Datei keine folgende Texteinheit oder ist die Datei leer, wird eine Null bzw. eine leere Zeichenfolge in die für laufende Nummer/Satznummer bzw. Text vorgesehenen Variablen (vgl. ACCESS-Anweisung) gespeichert.

Ob noch eine Texteinheit vorhanden war, kann auch mit einer nachfolgenden IF-Anweisung mit der Bedingung EOF (siehe Seite 476) abgefragt werden.

Bei READ-Anweisungen mit Option der NEXT, PREVIOUS oder IGNORE kann noch zusätzlich die Option EXIT angegeben werden:

```
$$ READ ... /EXIT daten
```

Dies ist jedoch nur sinnvoll, wenn die READ-Anweisung innerhalb einer Schleife (zwischen LOOP und ENDOOP) ausgeführt wird. Die Option EXIT bewirkt, dass automatisch eine EXIT-Anweisung ausgeführt wird (d. h. dass die Schleife automatisch verlassen wird), wenn keine Texteinheit mehr vorhanden war.

### Prüfen einer Texteinheit

Beim Prüfen einer Texteinheit wird nur geprüft, ob die Texteinheit vorhanden ist. Es werden keine Daten übertragen.

```
$$ READ/CHECK daten
```

Nur für TUSTEP-Dateien: Prüft, ob die Texteinheit mit der aktuellen Satznummer in der TUSTEP-Datei vorhanden ist. Ob die Texteinheit vorhanden ist, kann mit einer nachfolgenden IF-Anweisung mit der Bedingung EXIST (siehe Seite 476) abgefragt werden.

### Suchen einer bestimmten Texteinheit

Beim Suchen von Texteinheiten werden alle Sätze einer Texteinheit zu einer einzigen Zeichenfolge zusammengefasst; dann wird geprüft, ob diese Zeichenfolge die Suchbedingungen erfüllt. Die Suchbedingungen sind ab Seite 599 beschrieben.

```
$$ FIND daten (suchbed1) WHILE (suchbed2) UPTO nmmr
```

Bei TUSTEP-Dateien: Die Suche beginnt mit der Texteinheit mit der aktuellen Satznummer und erfolgt zum Dateiende hin. Gesucht wird nur so lange, wie die durchsuchten Texteinheiten die hinter WHILE angegebenen Suchbedingungen erfüllen, höchstens jedoch bis vor die Texteinheit mit der Satznummer, die hinter dem Schlüsselwort UPTO angegeben ist. Sowohl die Angabe »WHILE (suchbed2)« als auch die Angabe »UPTO nmmr« können weggelassen werden, wenn die entsprechende Einschränkung der Suche nicht erforderlich ist.

Bei Fremd-Dateien ist die Option NEXT (s. u.) obligat; die Angabe

»WHILE (suchbed2)« kann weggelassen werden, wenn die entsprechende Einschränkung der Suche nicht erforderlich ist; die Angabe »UPTO `nmnr`« ist bei Fremd-Dateien nicht vorgesehen.

Wird eine Texteinheit gefunden, wird diese (wie mit READ) eingelesen; wird keine gefunden, werden Nullen bzw. eine leere Zeichenfolge in die für die laufende Nummer/Satznummer bzw. Text vorgesehenen Variablen (vgl. ACCESS-Anweisung) gespeichert.

Ob noch eine Texteinheit gefunden wurde, kann auch mit einer nachfolgenden IF-Anweisung mit der Bedingung EOF (siehe Seite 476) abgefragt werden.

Durch Angabe einer der folgenden drei Optionen kann die Texteinheit, mit der die Suche beginnen soll, und die Suchrichtung anders festgelegt werden; bei Fremd-Dateien ist die Option NEXT obligat.

```
$$ FIND/NEXT daten (suchbedingungen) ...
```

Bei TUSTEP-Dateien: Die Suche beginnt hinter der Texteinheit mit der aktuellen Satznummer und erfolgt zum Dateende hin. War in der ACCESS-Anweisung die Option RAW angegeben, wird ohne Berücksichtigung der aktuellen Satznummer ab der nächsten Texteinheit (d. i. der nächste Satz) gesucht; die Angabe UPTO ist in diesem Fall nicht erlaubt.

Bei Fremd-Dateien: Die Suche beginnt mit der nächsten Texteinheit und erfolgt zum Dateende hin. Falls die aktuelle laufende Nummer Null ist, ist die erste Texteinheit die nächste.

```
$$ FIND/REVERSE daten (suchbedingungen) ...
```

Nur für TUSTEP-Dateien: Die Suche beginnt mit der Texteinheit mit der aktuellen Satznummer und erfolgt zum Dateianfang hin.

```
$$ FIND/PREVIOUS daten (suchbedingungen) ...
```

Nur für TUSTEP-Dateien: Die Suche beginnt vor der Texteinheit mit der aktuellen Satznummer und erfolgt zum Dateianfang hin.

Bei allen oben aufgeführten FIND-Anweisungen können noch zusätzliche Optionen in der folgenden Reihenfolge angegeben werden:

```
$$ FIND ... /SKIP daten (suchbedingungen) ...
```

Die Option SKIP bewirkt, dass eine vorgegebenen Anzahl von Texteinheiten, die die Suchbedingungen erfüllen, übergangen werden. Die Anzahl wird aus der dafür vorgesehenen Variablen (siehe ACCESS-Anweisung) entnommen. Enthält diese Variable eine Null, so ist die Option SKIP wirkungslos.

```
$$ FIND ... /READ daten (suchbedingungen) ...
```

Die Option READ besagt, dass die gefundene Texteinheit automatisch (wie mit der READ-Anweisung) eingelesen wird. Diese Option ist voreingestellt und kann mit der nachfolgend beschriebenen Option aufgehoben werden.

```
$$ FIND ... /CHECK daten (suchbedingungen) ...
```

Die Option CHECK bewirkt, dass die gefundene Texteinheit nicht (wie mit der READ-



Anweisung) eingelesen wird. Von der gefundenen Texteinheit wird bei Fremd-Dateien nur die laufende Nummer, bei TUSTEP-Dateien nur die Satznummer in die in der ACCESS-Anweisung dafür vorgesehenen Variablen gespeichert.

Bei allen FIND-Anweisungen kann noch zusätzlich die Option EXIT angegeben werden:

```
$$ FIND ... /EXIT daten
```

Dies ist jedoch nur sinnvoll, wenn die FIND-Anweisung innerhalb einer Schleife (zwischen LOOP und ENDDLOOP) ausgeführt wird. Die Option EXIT bewirkt, dass automatisch eine EXIT-Anweisung ausgeführt wird (d. h. dass die Schleife automatisch verlassen wird), wenn keine Texteinheit gefunden wurde.

### Zählen bestimmter Texteinheiten

Die Anweisung COUNT zum Zählen bestimmter Texteinheiten entspricht (einschließlich der möglichen Optionen) der Anweisung FIND zum Suchen einer bestimmten Texteinheit, die auf Seite 607 beschrieben ist.

```
$$ COUNT daten (suchbed1) WHILE (suchbed2) UPTO nmmr
```

Es wird derselbe Datenbereich wie bei der FIND-Anweisung durchsucht. Jedoch wird nicht nur die erste Texteinheit gesucht, die die Suchbedingungen erfüllt, sondern festgestellt, wieviele Texteinheiten insgesamt die Suchbedingung erfüllen.

Die letzte gefundene Texteinheit wird (wie mit READ) eingelesen, falls nicht die Option CHECK angegeben ist; wird keine gefunden, werden Nullen bzw. eine leere Zeichenfolge in die für laufende Nummer/Satznummer bzw. Text vorgesehenen Variablen (vgl. ACCESS-Anweisung) gespeichert.

### Löschen einer Texteinheit

```
$$ ERASE daten
```

Nur für TUSTEP-Dateien: Löscht in der Datei die Texteinheit mit der aktuellen Satznummer.

## Dateizugriffe – Daten mit Anfangs- und Endekennungen

Diese Dateizugriffe erlauben das Beschreiben und Lesen von TUSTEP- und Fremd-Dateien sowie von Variablen (siehe Seite 598). Der »Dateizugriff« auf eine Variable erfolgt nach den gleichen Regeln wie auf eine Fremd-Datei, jedoch kann die Angabe eines Codes entfallen, wenn die Daten nicht umcodiert werden sollen.

Beim Lesen werden die Daten in Textportionen unterteilt. Hierfür können mit Such-Tabellen Zeichenfolgen definiert werden, die als »Anfangskennung«, als »Trennzeichenfolge« oder als »Endekennung« dienen. Mindestens eine der drei möglichen Such-Tabellen muss angegeben werden. Eine neue Textportion beginnt jeweils unmittelbar vor einer Anfangskennung, unmittelbar nach einer Endekennung sowie unmittelbar vor und nach einer Trennzeichenfolge. Anfangskennungen, Trennzeichenfolgen und Endekennungen müssen in den Daten jeweils vollständig in einem Satz stehen, damit sie erkannt werden. Im Übrigen ist die Einteilung der Daten in Sätze belanglos, wenn in der ACCESS-Anweisung nicht die Option RECORDS angegeben wird.

Beim Lesen einer Textportion werden jeweils an drei Variablen Daten zugewiesen; falls keine entsprechenden Daten vorhanden sind, wird eine leere Zeichenfolge zugewiesen. Dazu werden intern nacheinander folgende vier Schritte ausgeführt:

- Folgt in den Daten eine Trennzeichenfolge?
  - ja: diese Trennzeichenfolge der zweiten Variablen zuweisen; der ersten und dritten Variablen eine leere Zeichenfolge zuweisen; nachfolgende drei Schritte überspringen
  - nein: Weiter mit dem nächsten Schritt
- Folgt in den Daten eine Anfangskennung?
  - ja: Diese Anfangskennung der ersten Variablen zuweisen
  - nein: Der ersten Variablen eine leere Zeichenfolge zuweisen
- Folgt in den Daten Text?
  - ja: Diesen Text der zweiten Variablen zuweisen
  - nein: Der zweiten Variablen eine leere Zeichenfolge zuweisen
- Folgt in den Daten eine Endekennung?
  - ja: Diese Endekennung der dritten Variablen zuweisen
  - nein: Der dritten Variablen eine leere Zeichenfolge zuweisen

Beispiel:

In den folgenden Beispieldaten stehen die Zeichenfolgen »[a]« bzw. »[e]« für beliebige Anfangs- bzw. Endekennungen, die Zahlwörter für beliebigen Text, der jedoch keine Anfangs- bzw. Endekennungen enthält.

- Daten in der Datei

```
null[a]eins[e]zwei  
[a][a]drei[e]vier  
fünf[e]sechs
```

### – Makroanweisungen

```

$$ BUILD S_TABLE anftab = ":\[a\]:"
$$ BUILD S_TABLE endtab = ":\[e\]:"
$$ ACCESS d: READ/STREAM "datei" ...
$$          s.z/u, anfken/anftab + txt + endken/endtab, typ
$$ LOOP
$$   READ/EXIT d
$$   ...
$$ ENDLOOP
$$ ENDACCESS d

```

### – Inhalt der Variablen nach jedem READ

anfken = ""	txt = "null"	endken = ""	typ = 0
anfken = "[a]"	txt = "eins"	endken = "[e]"	typ = 3
anfken = ""	txt = "zwei"	endken = ""	typ = 0
anfken = "[a]"	txt = ""	endken = ""	typ = 1
anfken = "[a]"	txt = "drei"	endken = "[e]"	typ = 3
anfken = ""	txt = "vier fünf"	endken = "[e]"	typ = 2
anfken = ""	txt = "sechs"	endken = ""	typ = 0
anfken = ""	txt = ""	endken = ""	typ = 0

### – Mit zusätzlicher Option RECORDS nach der Option STREAM

anfken = ""	txt = "null"	endken = ""	typ = 0
anfken = "[a]"	txt = "eins"	endken = "[e]"	typ = 3
anfken = ""	txt = "zwei"	endken = ""	typ = 0
anfken = "[a]"	txt = ""	endken = ""	typ = 1
anfken = "[a]"	txt = "drei"	endken = "[e]"	typ = 3
anfken = ""	txt = "vier"	endken = ""	typ = 0
anfken = ""	txt = "fünf"	endken = "[e]"	typ = 2
anfken = ""	txt = "sechs"	endken = ""	typ = 0
anfken = ""	txt = ""	endken = ""	typ = 0

## Definieren des Dateizugriffs

Der Zugriff auf Daten in Dateien oder Variablen mit den in diesem Kapitel beschriebenen Makroanweisungen muss mit der folgenden ACCESS-Anweisung begonnen und mit einer ENDACCESS-Anweisung (siehe Seite 597) beendet werden.

```

$$ ACCESS daten: modus/option "dateiname" ...
$$          nmmr, aken/atab + text/ttab + eken/etab, typ

```

### Bestandteile der ACCESS-Anweisung

- daten: Frei wählbarer Name für die Daten; er wird bei allen anderen Anweisungen für den Dateizugriff an Stelle eines Dateinamens bzw. Variablennamens verwendet, um anzugeben, auf welche Datei bzw. Variable zugegriffen werden soll.
- Doppelpunkt
- modus/option: READ/STREAM, falls von der nachfolgend angegebenen Datei ge-

lesen werden soll; WRITE/STREAM, falls ans Ende der Datei geschrieben werden soll.

Bei Modus READ kann nach der Option STREAM zusätzlich die Option RECORDS oder die Option \* angegeben werden. Mit der Option RECORDS werden die Daten auch jeweils am Beginn eines neuen Satzes unterteilt, mit der Option \* wird der Text nicht zu einer Zeichenfolge zusammengefasst, sondern mit der gleichen Zeileneinteilung wie in der Datei als Sternvariable gespeichert.

Bei Modus WRITE kann zusätzlich die Option ERASE angegeben werden; in diesem Fall werden bei der Ausführung der ACCESS-Anweisung alle Daten in der Datei bzw. Variablen gelöscht. Die Option ERASE muss als erste Option bzw. unmittelbar nach der Option FILE oder VARIABLE (vgl. Seite 598) angegeben werden.

Bei Fremd-Dateien muss als zusätzliche Option der Code für die Daten angegeben werden; vorgesehen sind ISO (= ISO-8859-1), UTF8, UTF16 und BINARY (= Daten nicht umcodieren). Wird bei Modus READ nach dem Code noch die Option NSP angegeben, wird bei Zeilenwechsel kein Leerzeichen in die Daten eingefügt.

Am Ende der Angabe option kann durch einen Schrägstrich getrennt ein Zeitlimit angegeben werden. Die Bedeutung des Zeitlimits ist unter »Warten bis Dateizugriff möglich« auf Seite 598 beschrieben.

- `dateiname`: Name der Datei; er muss zwischen Anführungszeichen angegeben werden. Wird eine Variable angegeben, die den Dateinamen enthält, muss als erste Option FILE angegeben werden; wird eine Variable angegeben, in die geschrieben bzw. aus der gelesen werden soll, muss als erste Option VARIABLE angegeben werden (vgl. Seite 598).
- Drei Punkte: Sie zeigen an, dass die ACCESS-Anweisung in der nächsten Zeile fortgesetzt wird. Sie entfallen, wenn die ganze Anweisung in eine Zeile geschrieben wird.
- `nmnr`: Bei Fremd-Dateien wird der Name einer Variablen erwartet, die die »laufende Nummer« des ersten (und evtl. einzigen) Satzes der Textportion enthält. Bei TUSTEP-Dateien werden drei Variablennamen erwartet, die durch Punkt und Schrägstrich getrennt sind (z. B. `snr.znr/unr`). Die Variablen enthalten die »aktuelle Satznummer« (Seiten-, Zeilen- und Unterscheidungsnummer) des ersten (und evtl. einzigen) Satzes der Textportion. Diese Variablen werden von den nachfolgend beschriebenen Makroanweisungen ausgewertet oder evtl. geändert.
- Komma
- `aken/atab`: Name der Variablen für die Anfangskennung und Name der Such-Tabelle, die die möglichen Anfangskennungen enthält. Falls keine Anfangskennungen berücksichtigt werden sollen, entfällt die Angabe von `atab` einschließlich des davor stehenden Schrägstrichs. Bei Modus WRITE entfällt diese Angabe ebenfalls.
- Pluszeichen
- `text/ttab`: Name der Variablen für den Text bzw. die Trennzeichenfolge und Name der Such-Tabelle, die die möglichen Trennzeichenfolgen enthält. Falls keine Trennzeichenfolgen berücksichtigt werden sollen, entfällt die Angabe von

`ttab` einschließlich des davor stehenden Schrägstrichs; bei Modus `WRITE` entfällt diese Angabe ebenfalls.

- Pluszeichen
- `eken/etab`: Name der Variablen für die Endekennung und Name der Such-Tabelle, die die möglichen Endekennungen enthält. Falls keine Endekennungen berücksichtigt werden sollen, entfällt die Angabe von `etab` einschließlich des davor stehenden Schrägstrichs; bei Modus `WRITE` entfällt diese Angabe ebenfalls.
- Komma

- `typ`: Name der Variablen, die nach Ausführung der Makroanweisung `READ` angibt, ob die Textportion außer dem Text eine Anfangskennung und/oder eine Endekennung enthält, oder ob sie nur aus einer Trennzeichenfolge besteht:

0 = nur Text

1 = Anfangskennung, ggf. auch Text

2 = Endekennung, ggf. auch Text

3 = Anfangs- und Endekennung, ggf. auch Text

4 = nur Trennzeichenfolge

Wird diese Information nicht benötigt, kann die Variable `typ` einschließlich des davor stehenden Kommas weggelassen werden.

Für die Makroanweisung `WRITE` ist der Inhalt dieser Variablen (noch) ohne Bedeutung.

Bei der Ausführung der `ACCESS`-Anweisung wird der Inhalt der in der Anweisung angegebenen Variablen auf Null gesetzt bzw. gelöscht.

## Schreiben einer Textportion

Beim Schreiben einer Textportion wird der Inhalt von drei Variablen in die Datei ausgegeben. Die Variablen können jeweils durch ein Pluszeichen getrennt in der `WRITE`-Anweisung nach dem Datennamen explizit angegeben werden; andernfalls werden die drei in der `ACCESS`-Anweisung für Anfangskennung, Text und Endekennung vorgesehenen Variablen verwendet.

Nach dem Datennamen kann an Stelle einer Variablen auch eine zwischen Anführungszeichen stehende Zeichenfolge angegeben werden. Außerdem kann statt der drei Variablen auch nur eine Variable bzw. nur eine Zeichenfolge angegeben werden; dann wird nur der Inhalt dieser Variablen bzw. nur die angegebene Zeichenfolge in die Datei ausgegeben.

```
$$ WRITE daten
```

Schreibt eine Textportion in die Datei. Falls die laufende Nummer/Satznummer seit der letzten `WRITE`-Anweisung nicht verändert wurde, wird die Textportion im gleichen Satz hinzugefügt; falls die laufende Nummer/Satznummer erhöht wurde, wird ein neuer Satz begonnen; falls sie erniedrigt wurde, erfolgt eine Fehlermeldung und das Fehlerflag wird gesetzt.

Wenn die Variable für den Text eine Sternvariable ist, wird für jede Zeile der Variablen ein neuer Satz begonnen. Die Anfangskennung wird am Anfang des ersten Satzes, die

Endekennung am Ende des letzten Satzes ergänzt. Um wieviel die Satznummer dabei für jede Zeile erhöht wird, kann mit der `MODIFY`-Anweisung (siehe unten) festgelegt werden.

```
$$ WRITE/ADJUST daten
```

Wie `WRITE`, jedoch wird keine Fehlermeldung ausgegeben, falls die Nummer/Satznummer erniedrigt wurde, sondern die Nummer/Satznummer automatisch erhöht und ein neuer Satz begonnen. Um wieviel die Satznummer erhöht wird, kann mit der `MODIFY`-Anweisung (siehe unten) festgelegt werden.

```
$$ WRITE/LAST daten
```

Wie `WRITE`, jedoch wird die Textportion, auch wenn die Satznummer verändert wurde, im letzten Satz hinzugefügt.

```
$$ WRITE/NEXT daten
```

Bei `TUSTEP`-Dateien: Wie `WRITE`, jedoch wird die aktuelle Zeilennummer zuvor erhöht und ein neuer Satz begonnen. Um wieviel die Satznummer erhöht wird, kann mit der `MODIFY`-Anweisung (siehe unten) festgelegt werden. Hat die aktuelle Satznummer den Wert Null und stehen in der Datei schon Daten, wird zuvor die Satznummer des letzten Satzes als aktuelle Satznummer übernommen.

Bei Fremd-Dateien: Wie `WRITE`, jedoch wird die laufende Nummer zuvor um 1 erhöht und ein neuer Satz begonnen.

```
$$ WRITE/CLEAR daten
```

Bei allen oben beschriebenen `WRITE`-Anweisungen kann `CLEAR` als letzte Option angegeben werden. Sie bewirkt, dass der Inhalt der Variablen, deren Inhalt in die Datei ausgegeben wurde, anschließend automatisch gelöscht wird.

```
$$ WRITE/BREAK daten
```

Mit dieser Anweisung werden keine Daten in die Datei geschrieben. Sie bewirkt nur, dass mit der nächsten `WRITE`-Anweisung ein neuer Satz begonnen wird, so als wäre die Option `NEXT` angegeben, unabhängig davon, ob die Option `NEXT` oder die Option `LAST` angegeben ist oder nicht.

```
$$ MODIFY ACCESS daten type schrittweite
```

Mit dieser Anweisung kann festgelegt werden, um wieviel die Satznummer ggf. erhöht wird. Ist für `type` `ADJUST` oder `NEXT` angegeben, so gilt die Schrittweite für die Anweisungen `WRITE/ADJUST` bzw. `WRITE/NEXT`; ist für `type *` angegeben, so gilt die Schrittweite für die zweite und alle folgenden Zeilen, wenn eine Sternvariable in die Datei geschrieben wird.

Voreingestellt ist als Schrittweite 1000, d. h. dass die Zeilennummer um 1 erhöht wird und die Unterscheidungsnummer auf Null gesetzt wird. Ist eine kleinere Schrittweite angegeben, wird die Unterscheidungsnummer um den angegebenen Wert erhöht. Falls sich beim Erhöhen der Zeilennummer ein Wert größer als 999 ergibt, wird die Seitennummer um 1 erhöht und die Zeilennummer auf Null gesetzt; falls sich beim Erhöhen der Unterscheidungsnummer ein Wert größer als 999 ergibt, wird die Zeilennummer um 1 erhöht und die Unterscheidungsnummer zurückgesetzt.

Unmittelbar vor der Ausgabe eines Satzes in die Datei können Zeichenfolgen ausgetauscht werden. Mit der Anweisung

```
$$ MODIFY ACCESS daten X_TABLE xtab
```

wird dazu die Austausch-Tabelle eingestellt. Es gilt jeweils nur die zuletzt eingestellte Tabelle. Für `xtab` muss der Name einer mit der `BUILD`-Anweisung definierten Austausch-Tabelle angegeben werden. Wird für `xtab` ein Minuszeichen angegeben, so wird die Einstellung wieder aufgehoben, d.h es werden keine Zeichenfolgen mehr ausgetauscht.

## Lesen einer Textportion

Nach dem Einlesen eines jeden Satzes von der Datei und vor der Aufteilung der Daten in Textportionen können Zeichenfolgen ausgetauscht werden. Mit der Anweisung

```
$$ MODIFY ACCESS daten X_TABLE xtab
```

wird dazu die Austausch-Tabelle eingestellt. Es gilt jeweils nur die zuletzt eingestellte Tabelle. Für `xtab` muss der Name einer mit der `BUILD`-Anweisung definierten Austausch-Tabelle angegeben werden. Wird für `xtab` ein Minuszeichen angegeben, so wird die Einstellung wieder aufgehoben, d.h es werden keine Zeichenfolgen mehr ausgetauscht.

Beim Lesen einer Textportion werden Anfangskennung, Text/Trennzeichenfolge und Endekennung jeweils in eine Variable gespeichert. Die drei Variablen können jeweils durch ein Pluszeichen getrennt in der `READ`-Anweisung nach dem Datennamen explizit angegeben werden; andernfalls werden die in der `ACCESS`-Anweisung dafür vorgesehenen Variablen verwendet. Außerdem wird bei Fremd-Dateien die laufende Nummer, bei `TUSTEP`-Dateien die Satznummer des ersten Satzes der gelesenen Textportion in die in der `ACCESS`-Anweisung dafür vorgesehenen Variablen gespeichert.

```
$$ READ daten
```

Liest eine Textportion. Falls keine Daten mehr vorhanden sind, wird eine Null bzw. ein leere Zeichenfolge in die für laufende Nummer/Satznummer bzw. Textportion vorgesehenen Variablen (vgl. `ACCESS`-Anweisung) gespeichert.

Ob noch Daten vorhanden waren, kann auch mit einer nachfolgenden `IF`-Anweisung mit der Bedingung `EOF` (siehe Seite 476) abgefragt werden.

Mit einer nachfolgenden `IF`-Anweisung mit der Bedingung `SOR` bzw. `EOR` (siehe Seite 475) kann abgefragt werden, ob die gelesene Textportion an einem Satzanfang begann bzw. an einem Satzende endete.

```
$$ READ/EXIT daten
```

Die Option `EXIT` ist nur sinnvoll, wenn die `READ`-Anweisung innerhalb einer Schleife (zwischen `LOOP` und `ENDLOOP`) ausgeführt wird. Diese Option bewirkt, dass automatisch eine `EXIT`-Anweisung ausgeführt wird (d. h. dass die Schleife automatisch verlassen wird), wenn keine Daten mehr vorhanden waren.

```
$$ MODIFY ACCESS daten S_TABLE atab, ttab, etab
```

Legt die Such-Tabellen mit den Zeichenfolgen neu fest, die als »Anfangskennung«, als »Trennzeichenfolge« oder als »Endekennung« dienen. Es gilt jeweils nur die letzte MODIFY-Anweisung.

- atab: Minuszeichen oder Name einer Such-Tabelle. Ist eine Tabelle angegeben, gelten die darin enthaltenen Suchzeichenfolgen als Anfangskennungen.
- Komma
- ttab: Minuszeichen oder Name einer Such-Tabelle. Ist eine Tabelle angegeben, gelten die darin enthaltenen Suchzeichenfolgen als Trennzeichenfolgen.
- Komma
- etab: Minuszeichen oder Name einer Such-Tabelle. Ist eine Tabelle angegeben, gelten die darin enthaltenen Suchzeichenfolgen als Endekennungen.



## Dateizugriffe – Daten mit Tags

Ein Tag ist eine in spitzen Klammern eingeschlossene Zeichenfolge. Es gibt Anfangs-Tags, Ende-Tags und leere Tags:

Anfangs-Tag:

```
<name>
<name attribut1=wert1, attribut2=wert2, ... >
```

Ende-Tag:

```
</name>
```

Leeres Tag:

```
<name/>
<name attribut1=wert1, attribut2=wert2, ... />
```

Außerdem gibt es noch Kommentare, CDATA-Abschnitte, Verarbeitungsanweisungen (processing instructions) und DOCTYPE-Tags. Kommentare beginnen mit

```
<!-- und enden mit -->
```

CDATA-Abschnitte beginnen mit

```
<![CDATA[ und enden mit ]]>
```

Verarbeitungsanweisungen beginnen mit

```
<? und enden mit ?>
```

DOCTYPE-Tags beginnen mit

```
<!DOCTYPE und enden in der gleichen Zeile mit >
```

Enthalten Kommentare, CDATA-Abschnitte oder Verarbeitungsanweisungen spitze Klammern, so werden sie nicht als Anfangs- bzw. Endekennung von Tags gewertet.

Damit die nachfolgend beschriebenen Anweisungen die angegebene Wirkung haben, dürfen spitze Klammern außerhalb von Kommentaren, CDATA-Abschnitten und Verarbeitungsanweisungen nur als Anfangs- bzw. Endekennung von Tags, in Akzent-Codierungen (z. B. »%«) oder in den Codes für doppelte Anführungszeichen (»#.« und »#.«) vorkommen; andere spitze Klammern müssen z. B. mit ^< bzw. ^> codiert sein.

Bei der Suche nach Tags wird nach einer öffnenden spitzen Klammer (Anfangskennung eines Tags) die schließende spitze Klammer (Endekennung eines Tags) in der gleichen Zeile erwartet. Erstrecken sich Tags über Zeilenwechsel hinweg, muss mit den Optionen `amax` und/oder `emax` in der `ACCESS`-Anweisung angegeben werden, wieviele Zeichen ein Tag maximal umfassen kann. Folgt innerhalb dieser Zeichenzahl oder bis zur nächsten öffnenden spitzen Klammer keine schließende spitze Klammer, wird die Suche nach der Klammer abgebrochen.

Die im Folgenden beschriebenen Dateizugriffe erlauben das Beschreiben und Lesen von TUSTEP- und Fremd-Dateien sowie von Variablen (siehe Seite 598). Der »Dateizugriff« auf eine Variable erfolgt nach den gleichen Regeln wie auf eine Fremd-Datei,

jedoch kann die Angabe eines Codes entfallen, wenn die Daten nicht umcodiert werden sollen.

Beim Lesen werden die Daten in Textportionen unterteilt. Eine neue Textportion beginnt jeweils unmittelbar vor jedem Anfangs-Tag, vor jedem Kommentar-Anfang, vor jedem CDATA-Abschnitts-Anfang, vor jedem Verarbeitungsanweisungs-Anfang, unmittelbar nach jedem Ende-Tag, nach jedem Kommentar-Ende, nach jedem CDATA-Abschnitts-Ende, nach jedem Verarbeitungsanweisungs-Ende, sowie unmittelbar vor und nach jedem leeren Tag. Die Einteilung der Daten in Sätze ist für die Unterteilung der Daten in einzelne Portionen belanglos, wenn in der ACCESS-Anweisung nicht die Option RECORDS angegeben wird.

Beim Lesen einer Textportion werden jeweils an drei Variablen Daten zugewiesen; falls keine entsprechenden Daten vorhanden sind, wird eine leere Zeichenfolge zugewiesen. Dazu werden intern nacheinander folgende vier Schritte ausgeführt:

- Folgt in den Daten ein leeres Tag?
  - ja: Dieses leere Tag der zweiten Variablen zuweisen; der ersten und dritten Variablen eine leere Zeichenfolge zuweisen; nachfolgende drei Schritte überspringen
  - nein: Weiter mit dem nächsten Schritt
- Folgt in den Daten ein Anfangs-Tag, ein Kommentar-Anfang, ein CDATA-Abschnitts-Anfang oder ein Verarbeitungsanweisungs-Anfang?
  - ja: Dieses Anfangs-Tag bzw. den Kommentar-Anfang bzw. den CDATA-Abschnitts-Anfang bzw. den Verarbeitungsanweisungs-Anfang der ersten Variablen zuweisen.
  - nein: Der ersten Variablen eine leere Zeichenfolge zuweisen.
- Folgen in den Daten Text oder Kommentar oder CDATA-Daten oder Verarbeitungsanweisungen?
  - ja: Diesen Text bzw. den Kommentar bzw. die CDATA-Daten bzw. die Verarbeitungsanweisungen der zweiten Variablen zuweisen.
  - nein: Der zweiten Variablen eine leere Zeichenfolge zuweisen.
- Folgt in den Daten ein Ende-Tag oder ein Kommentar-Ende oder ein CDATA-Abschnitts-Ende oder ein Verarbeitungsanweisungs-Ende?
  - ja: Dieses Ende-Tag bzw. das Kommentar-Ende bzw. das CDATA-Abschnitts-Ende bzw. das Verarbeitungsanweisungs-Ende der dritten Variablen zuweisen.
  - nein: Der dritten Variablen eine leere Zeichenfolge zuweisen.

Beispiel:

In den folgenden Beispieldaten stehen die Zahlwörter für beliebigen Text, der jedoch keine Tags enthält.

- Daten in der Datei

```
null<a>eins</a>zwei  
<b><c>drei</c>vier  
fünf</b>sechs<d/>
```

### – Makroanweisungen

```

$$ ACCESS d: READ/STREAM "datei" ...
$$           s.z/u, anftag + txt + endtag, typ
$$ LOOP
$$   READ/EXIT d
$$   ...
$$ ENDLLOOP
$$ ENDACCESS d

```

### – Inhalt der Variablen nach jedem READ

anftag = ""	txt = "null"	endtag = ""	typ = 0
anftag = "<a>"	txt = "eins"	endtag = "</a>"	typ = 3
anftag = ""	txt = "zwei"	endtag = ""	typ = 0
anftag = "<b>"	txt = ""	endtag = ""	typ = 1
anftag = "<c>"	txt = "drei"	endtag = "</c>"	typ = 3
anftag = ""	txt = "vier fünf"	endtag = "</b>"	typ = 2
anftag = ""	txt = "sechs"	endtag = ""	typ = 0
anftag = ""	txt = "<d/>"	endtag = ""	typ = 4
anftag = ""	txt = ""	endtag = ""	typ = 0

### – Mit zusätzlicher Option RECORDS nach der Option STREAM

anftag = ""	txt = "null"	endtag = ""	typ = 0
anftag = "<a>"	txt = "eins"	endtag = "</a>"	typ = 3
anftag = ""	txt = "zwei"	endtag = ""	typ = 0
anftag = "<b>"	txt = ""	endtag = ""	typ = 1
anftag = "<c>"	txt = "drei"	endtag = "</c>"	typ = 3
anftag = ""	txt = "vier"	endtag = ""	typ = 0
anftag = ""	txt = "fünf"	endtag = "</b>"	typ = 2
anftag = ""	txt = "sechs"	endtag = ""	typ = 0
anftag = ""	txt = "<d/>"	endtag = ""	typ = 4
anftag = ""	txt = ""	endtag = ""	typ = 0

## Definieren des Dateizugriffs

Der Zugriff auf Daten in Dateien oder Variablen mit den in diesem Kapitel beschriebenen Makroanweisungen muss mit der folgenden ACCESS-Anweisung begonnen und mit einer ENDACCESS-Anweisung (siehe Seite 597) beendet werden.

```

$$ ACCESS daten: modus/option "dateiname" ...
$$           nmmr, atag/amax + text + etag/emax,
$$           typ, stck, stck_num

```

### Bestandteile der ACCESS-Anweisung

- daten: Frei wählbarer Name für die Daten; er wird bei allen anderen Anweisungen für den Dateizugriff an Stelle eines Dateinamens bzw. Variablennamens verwendet, um anzugeben, auf welche Datei bzw. Variable zugegriffen werden soll.
- Doppelpunkt

- `modus/option`: `READ/STREAM`, falls von der nachfolgend angegebenen Datei gelesen werden soll; `WRITE/STREAM`, falls ans Ende der Datei geschrieben werden soll.

Bei Modus `READ` kann nach der Option `STREAM` zusätzlich die Option `RECORDS` angegeben werden; in diesem Fall werden die Daten auch jeweils am Beginn eines neuen Satzes unterteilt.

Bei Modus `WRITE` kann zusätzlich die Option `ERASE` angegeben werden; in diesem Fall werden bei der Ausführung der `ACCESS`-Anweisung alle Daten in der Datei bzw. Variablen gelöscht. Die Option `ERASE` muss als erste Option bzw. unmittelbar nach der Option `FILE` oder `VARIABLE` (vgl. Seite 598) angegeben werden.

Bei Fremd-Dateien muss als zusätzliche Option der Code für die Daten angegeben werden; vorgesehen sind `ISO` (= `ISO-8859-1`), `UTF8`, `UTF16` und `BINARY` (= Daten nicht umcodieren). Wird bei Modus `READ` nach dem Code noch die Option `NSP` angegeben, wird bei Zeilenwechsel kein Leerzeichen in die Daten eingefügt.

Am Ende der Angabe `option` kann durch einen Schrägstrich getrennt ein Zeitlimit angegeben werden. Die Bedeutung des Zeitlimits ist unter »Warten bis Dateizugriff möglich« auf Seite 598 beschrieben.

- `dateiname`: Name der Datei; er muss zwischen Anführungszeichen angegeben werden. Wird eine Variable angegeben, die den Dateinamen enthält, muss als erste Option `FILE` angegeben werden; wird eine Variable angegeben, in die geschrieben bzw. aus der gelesen werden soll, muss als erste Option `VARIABLE` angegeben werden (vgl. Seite 598).
- Drei Punkte: Sie zeigen an, dass die `ACCESS`-Anweisung in der nächsten Zeile fortgesetzt wird. Sie entfallen, wenn die Anweisung in der gleichen Zeile fortgesetzt wird.
- `nmnr`: Bei Fremd-Dateien wird der Name einer Variablen erwartet, die die »laufende Nummer« des ersten (und evtl. einzigen) Satzes der Textportion enthält. Bei `TUSTEP`-Dateien werden drei Variablennamen erwartet, die durch Punkt und Schrägstrich getrennt sind (z. B. `snr.znr/unr`). Die Variablen enthalten die »aktuelle Satznummer« (Seiten-, Zeilen- und Unterscheidungsnummer) des ersten (und evtl. einzigen) Satzes der Textportion. Diese Variablen werden von den nachfolgend beschriebenen Makroanweisungen ausgewertet oder evtl. geändert.
- Komma
- `atag/amax`: Name der Variablen für das Anfangs-Tag und Zahl (keine Variable), die die maximale Länge der Anfangs-Tags und der leeren Tags angibt. Falls die Anfangs-Tags und die leeren Tags jeweils vollständig in einer Zeile stehen, kann die Angabe von `amax` einschließlich des davor stehenden Schrägstrichs weggelassen werden; bei Modus `WRITE` entfällt diese Längenangabe ebenfalls.
- Pluszeichen
- `text`: Name der Variablen für den Text bzw. für leere Tags.
- Pluszeichen
- `etag/emax`: Name der Variablen für das Ende-Tag und Zahl (keine Variable), die

die maximale Länge der Ende-Tags angibt. Falls die Ende-Tags jeweils vollständig in einer Zeile stehen, kann die Angabe von `emax` einschließlich des davor stehenden Schrägstrichs weggelassen werden; bei Modus `WRITE` entfällt diese Längenangabe ebenfalls.

- Komma
- `typ`: Name der Variablen, die nach Ausführung der Makroanweisung `READ` angibt, aus welchen Teilen die gelesene Textportion besteht:
  - 0 = nur Text
  - 1 = Anfangs-Tag, ggf. auch Text
  - 2 = Ende-Tag, ggf. auch Text
  - 3 = Anfangs- und Ende-Tag, ggf. auch Text
  - 4 = nur leeres Tag
  - 5 = nur als leeres Tag bewertetes Anfangs-Tag
  - 6 = nur als leeres Tag bewertetes Ende-Tag
  - 7 = nur zusätzlich definiertes Trennzeichen
  - 8 = nur DOCTYPE-Tag
  - 9 = nur nicht identifiziertes Tag
  - 10 = nur Kommentar, ohne Kommentar-Anfang oder -Ende
  - 11 = Kommentar-Anfang, ggf. auch Kommentar
  - 12 = Kommentar-Ende, ggf. auch Kommentar
  - 13 = Kommentar-Anfang und -Ende, ggf. auch Kommentar
  - 20 = nur CDATA-Daten, ohne CDATA-Abschnitts-Anfang oder -Ende
  - 21 = CDATA-Abschnitts-Anfang, ggf. auch CDATA-Daten
  - 22 = CDATA-Abschnitts-Ende, ggf. auch CDATA-Daten
  - 23 = CDATA-Abschnitts-Anfang und -Ende, ggf. auch CDATA-Daten
  - 30 = nur Verarbeitungsanweisungen, ohne Verarbeitungsanweisungs-Anfang oder -Ende
  - 31 = Verarbeitungsanweisungs-Anfang, ggf. auch Verarbeitungsanweisungen
  - 32 = Verarbeitungsanweisungs-Ende, ggf. auch Verarbeitungsanweisung
  - 33 = Verarbeitungsanweisungs-Anfang und -Ende, ggf. auch Verarbeitungsanweisung

Wenn in den Daten außerhalb von Kommentaren, CDATA-Daten und Verarbeitungsanweisungen unpaarige spitze Klammern oder unpaarige Tags gefunden werden, enthält die Variable eine negative Fehlernummer:

- 1 = `>><<` ohne passende `>><<`
- 2 = `>><<` ohne passende `>><<`
- 3 = Namen von Anfangs-Tag und Ende-Tag unterscheiden sich in Groß- und Kleinschreibung
- 4 = Anfangs-Tag ohne entsprechendes Ende-Tag
- 5 = Ende-Tag ohne entsprechendes Anfangs-Tag
- 6 = Noch offene Tags am Dateiende

Wird diese Information nicht benötigt und auch die nachfolgenden Variablen `stck` und `stck_num` nicht angegeben, kann die Variable `typ` einschließlich des davor stehenden Kommas weggelassen werden.

Für den Modus `WRITE` ist der Inhalt dieser Variablen (noch) ohne Bedeutung.

Bei Modus `WRITE` endet die Anweisung hier; bei Modus `READ` können ein oder

zwei weitere Variablen angegeben werden, falls die entsprechende Informationen benötigt werden.

- Komma
- `stck`: Name einer Variablen, die nach dem Lesen einer Textportion die Namen aller Tags enthält, die am Beginn der in der Variablen `text` gespeicherten Daten offen sind. Die Namen sind jeweils in spitze Klammern eingeschlossen und stehen ohne weitere Trennzeichen hintereinander.
- Komma
- `stck_num`: Name einer Variablen, die nach dem Lesen einer Textportion außer der in der Variablen `stck` enthaltene Information hinter jedem Tag eine in eckige Klammern eingeschlossene Nummer enthält. Diese Nummer gibt an, das wievielte gleichnamige Tag es ist, das jeweils dieselben (!) übergeordneten Tags hat.

Bei der Ausführung der `ACCESS`-Anweisung wird der Inhalt der in der Anweisung angegebenen Variablen auf Null gesetzt bzw. gelöscht.

### Schreiben einer Textportion

Beim Schreiben einer Textportion wird der Inhalt von drei Variablen in die Datei ausgegeben. Die Variablen können jeweils durch ein Pluszeichen getrennt in der `WRITE`-Anweisung nach dem Datennamen explizit angegeben werden; andernfalls werden die drei in der `ACCESS`-Anweisung für Anfangs-Tag, Text und Ende-Tag vorgesehenen Variablen verwendet.

Nach dem Datennamen kann an Stelle einer Variablen auch eine zwischen Anführungszeichen stehende Zeichenfolge angegeben werden. Außerdem kann statt der drei Variablen auch nur eine Variable bzw. nur eine Zeichenfolge angegeben werden; dann wird nur der Inhalt dieser Variablen bzw. nur die angegebene Zeichenfolge in die Datei ausgegeben.

```
$$ WRITE daten
```

Schreibt eine Textportion in die Datei. Falls die laufende Nummer/Satznummer seit der letzten `WRITE`-Anweisung nicht verändert wurde, wird die Textportion im gleichen Satz hinzugefügt, andernfalls wird ein neuer Satz begonnen.

Wenn die Variable für den Text eine Sternvariable ist, wird für jede Zeile der Variablen ein neuer Satz begonnen. Die Anfangskennung wird am Anfang des ersten Satzes, die Endkennung am Ende des letzten Satzes ergänzt.

```
$$ WRITE/LAST daten
```

Wie `WRITE`, jedoch wird die Textportion, auch wenn die Satznummer verändert wurde, im letzten Satz hinzugefügt.

```
$$ WRITE/NEXT daten
```

Bei `TUSTEP`-Dateien: Wie `WRITE`, jedoch wird die aktuelle Zeilennummer zuvor um 1 erhöht und ein neuer Satz begonnen. Hat die aktuelle Satznummer den Wert Null und stehen in der Datei schon Daten, wird zuvor die Satznummer des letzten Satzes als aktuelle Satznummer übernommen und die Zeilennummer um 1 erhöht. Falls

sich beim Erhöhen der Zeilennummer ein Wert größer als 999 ergibt, wird die Seitennummer um 1 erhöht und die Zeilennummer auf Null gesetzt.

Bei Fremd-Dateien: Wie `WRITE`, jedoch wird die laufende Nummer zuvor um 1 erhöht und ein neuer Satz begonnen.

```
$$ WRITE/CLEAR daten
```

Bei allen `WRITE`-Anweisungen kann `CLEAR` als letzte Option angegeben werden. Sie bewirkt, dass der Inhalt der Variablen, deren Inhalt in die Datei ausgegeben wurde, anschließend automatisch gelöscht wird.

Unmittelbar vor der Ausgabe eines Satzes in die Datei können Zeichenfolgen ausgetauscht werden. Mit der Anweisung

```
$$ MODIFY ACCESS daten X_TABLE xtab
```

wird dazu die Austausch-Tabelle eingestellt. Es gilt jeweils nur die zuletzt eingestellte Tabelle. Für `xtab` muss der Name einer mit der `BUILD`-Anweisung definierten Austausch-Tabelle angegeben werden. Wird für `xtab` ein Minuszeichen angegeben, so wird die Einstellung wieder aufgehoben, d.h es werden keine Zeichenfolgen mehr ausgetauscht.

## Lesen einer Textportion

Nach dem Einlesen eines jeden Satzes von der Datei und vor der Aufteilung der Daten in Textportionen können Zeichenfolgen ausgetauscht werden. Mit der Anweisung

```
$$ MODIFY ACCESS daten X_TABLE xtab
```

wird dazu die Austausch-Tabelle eingestellt. Es gilt jeweils nur die zuletzt eingestellte Tabelle. Für `xtab` muss der Name einer mit der `BUILD`-Anweisung definierten Austausch-Tabelle angegeben werden. Wird für `xtab` ein Minuszeichen angegeben, so wird die Einstellung wieder aufgehoben, d.h es werden keine Zeichenfolgen mehr ausgetauscht.

Beim Lesen einer Textportion werden Anfangs-Tag, Text/leeres Tag und Ende-Tag jeweils in eine Variable gespeichert. Die drei Variablen können jeweils durch ein Pluszeichen getrennt in der `READ`-Anweisung nach dem Datennamen explizit angegeben werden; andernfalls werden die in der `ACCESS`-Anweisung dafür vorgesehenen Variablen verwendet. Außerdem wird bei Fremd-Dateien die laufende Nummer, bei `TUSTEP`-Dateien die Satznummer des ersten Satzes der gelesenen Textportion in die in der `ACCESS`-Anweisung dafür vorgesehenen Variablen gespeichert.

```
$$ READ daten
```

Liest eine Textportion. Falls keine Daten mehr vorhanden sind, wird eine Null bzw. ein leere Zeichenfolge in die für laufende Nummer/Satznummer bzw. Textportion vorgesehenen Variablen (vgl. `ACCESS`-Anweisung) gespeichert.

Ob noch Daten vorhanden waren, kann auch mit einer nachfolgenden `IF`-Anweisung mit der Bedingung `EOF` (siehe Seite 476) abgefragt werden.

Mit einer nachfolgenden `IF`-Anweisung mit der Bedingung `SOR` bzw. `EOR` (siehe Seite 475) kann abgefragt werden, ob die gelesene Textportion in der Datei an einem Satzanfang begann bzw. an einem Satzende endete.

```
$$ READ/QUIET daten
```

Wenn in den Daten unpaarige spitze Klammern oder unpaarige Tags gefunden werden, erfolgt eine Fehlermeldung und das Fehlerflag wird gesetzt. Soll beides unterbleiben, muss die Option `QUIET` angegeben werden. In jedem Fall wird der Variablen `typ`, die in der `ACCESS`-Anweisung angegeben ist, eine entsprechende Fehlernummer zugewiesen.

```
$$ READ/EXIT daten
```

Die Option `EXIT` ist nur sinnvoll, wenn die `READ`-Anweisung innerhalb einer Schleife (zwischen `LOOP` und `ENDLOOP`) ausgeführt wird. Diese Option bewirkt, dass automatisch eine `EXIT`-Anweisung ausgeführt wird (d. h. dass die Schleife automatisch verlassen wird), wenn keine Daten mehr vorhanden waren.

```
$$ MODIFY ACCESS daten S_TABLE ttab
```

Für das Argument `ttab` kann ein Minuszeichen oder der Name einer Such-Tabelle angegeben werden. Ist eine Tabelle angegeben, wirken die darin enthaltenen Suchzeichenfolgen (in gleicher Weise wie leere Tags) zusätzlich als Trennzeichenfolgen. Es gilt jeweils nur die letzte `MODIFY`-Anweisung.

```
$$ MODIFY ACCESS daten R_TABLE pos, neg, stop, goon, empty
```

Legt Recherchier-Tabellen fest, die das Lesen der Textportionen zusätzlich steuern. Es gilt jeweils nur die letzte `MODIFY`-Anweisung.

- `pos`: Minuszeichen oder Name einer Recherchier-Tabelle. Ist eine Tabelle angegeben, wird für jede Textportion geprüft, ob der Inhalt der Variablen `stack` den Erfordernissen der Tabelle genügt; wenn dies nicht der Fall ist, wird die Textportion so übergangen, als wäre sie in den Daten nicht vorhanden.
- Komma
- `neg`: Minuszeichen oder Name einer Recherchier-Tabelle. Ist eine Tabelle angegeben, wird für jede Textportion geprüft, ob der Inhalt der Variablen `stack` den Erfordernissen der Tabelle genügt; wenn dies der Fall ist, wird die Textportion so übergangen, als wäre sie in den Daten nicht vorhanden.
- Komma
- `stop`: Minuszeichen oder Name einer Recherchier-Tabelle. Ist eine Tabelle angegeben, wird für jedes Tag geprüft, ob der Inhalt der Variablen `stack` den Erfordernissen der Tabelle genügt; wenn dies nicht der Fall ist, wird auf Grund dieses Tags keine neue Textportion begonnen und das Tag wie Text behandelt.
- Komma
- `goon`: Minuszeichen oder Name einer Recherchier-Tabelle. Ist eine Tabelle angegeben, wird für jedes Tag geprüft, ob der Inhalt der Variablen `stack` den Erfordernissen der Tabelle genügt; wenn dies der Fall ist, wird auf Grund dieses Tags keine neue Textportion begonnen und das Tag wie Text behandelt.
- Komma
- `empty`: Minuszeichen oder Name einer Recherchier-Tabelle. Ist eine Tabelle ange-



geben, wird für jedes Anfangs- und Ende-Tag geprüft, ob der Inhalt der Variablen `stack`, in die der Name des Tags zuvor eingefügt wurde, den Erfordernissen der Tabelle genügt; wenn dies der Fall ist, wird das Tag wie ein leeres Tag behandelt.

Ein Minuszeichen zu `positiv`, `negativ`, `stop`, `goon` und `empty` kann einschließlich des jeweils davor stehenden Kommas weggelassen werden, wenn es als letztes in der Anweisung stehen würde.

```
$$ MODIFY ACCESS daten DICTIONARY pos, neg, stop, goon, empty
```

Legt Wörterbücher fest, die das Lesen der Textportionen zusätzlich steuern. Die Angaben und deren Wirkungsweise entsprechen denen der vorangehenden Anweisung, mit der Recherchier-Tabellen festgelegt werden. Es gilt jeweils nur die letzte `MODIFY`-Anweisung. Die angegebenen Wörterbücher müssen mit der Option `TAGS` eingerichtet worden sein.

Beispiele:

– Daten in der Datei

```
<a1>1<b>2<c>3</c>4</b>5</a1>
<a2>6<b>7<c>8</c>9</b>0</a2>
```

– Makroanweisungen

```
$$ = {}
$$ ACCESS d: READ/STREAM "datei" s.z/u, at+txt+et, typ, stck
$$ LOOP
$$   READ/EXIT d
$$   ...
$$ ENDLLOOP
$$ ENDACCESS d
```

– Inhalt der Variablen nach jedem `READ`

<code>at="&lt;a1&gt;"</code>	<code>txt="1"</code>	<code>et=""</code>	<code>typ=1</code>	<code>stck="&lt;a1&gt;"</code>
<code>at="&lt;b&gt;"</code>	<code>txt="2"</code>	<code>et=""</code>	<code>typ=1</code>	<code>stck="&lt;a1&gt;&lt;b&gt;"</code>
<code>at="&lt;c&gt;"</code>	<code>txt="3"</code>	<code>et="&lt;/c&gt;"</code>	<code>typ=3</code>	<code>stck="&lt;a1&gt;&lt;b&gt;&lt;c&gt;"</code>
<code>at=""</code>	<code>txt="4"</code>	<code>et="&lt;/b&gt;"</code>	<code>typ=2</code>	<code>stck="&lt;a1&gt;&lt;b&gt;"</code>
<code>at=""</code>	<code>txt="5"</code>	<code>et="&lt;/a1&gt;"</code>	<code>typ=2</code>	<code>stck="&lt;a1&gt;"</code>
<code>at="&lt;a2&gt;"</code>	<code>txt="6"</code>	<code>et=""</code>	<code>typ=1</code>	<code>stck="&lt;a2&gt;"</code>
<code>at="&lt;b&gt;"</code>	<code>txt="7"</code>	<code>et=""</code>	<code>typ=1</code>	<code>stck="&lt;a2&gt;&lt;b&gt;"</code>
<code>at="&lt;c&gt;"</code>	<code>txt="8"</code>	<code>et="&lt;/c&gt;"</code>	<code>typ=3</code>	<code>stck="&lt;a2&gt;&lt;b&gt;&lt;c&gt;"</code>
<code>at=""</code>	<code>txt="9"</code>	<code>et="&lt;/b&gt;"</code>	<code>typ=2</code>	<code>stck="&lt;a2&gt;&lt;b&gt;"</code>
<code>at=""</code>	<code>txt="0"</code>	<code>et="&lt;/a2&gt;"</code>	<code>typ=2</code>	<code>stck="&lt;a2&gt;"</code>

– Gleiche Daten, modifiziertes Lesen

```
$$ BUILD R_TABLE positiv = ";<c>:"
$$ MODIFY ACCESS d R_TABLE positiv
```

<code>at="&lt;c&gt;"</code>	<code>txt="3"</code>	<code>et="&lt;/c&gt;"</code>	<code>typ=3</code>	<code>stck="&lt;a1&gt;&lt;b&gt;&lt;c&gt;"</code>
<code>at="&lt;c&gt;"</code>	<code>txt="8"</code>	<code>et="&lt;/c&gt;"</code>	<code>typ=3</code>	<code>stck="&lt;a2&gt;&lt;b&gt;&lt;c&gt;"</code>

## – Gleiche Daten, modifiziertes Lesen

```
$$ BUILD R_TABLE pos = " :<a1>*<c>:"
$$ MODIFY ACCESS d R_TABLE pos
```

```
at="<c>"   txt="3"           et="</c>"   typ=3   stck="<a1><b><c>"
```

## – Gleiche Daten, modifiziertes Lesen

```
$$ BUILD R_TABLE/OR negativ = " :<a1><b>:<a2>*<c>:"
$$ MODIFY ACCESS d R_TABLE -, negativ
```

```
at="<a1>"  txt="15"           et="</a1>"  typ=3   stck="<a1>"
at="<a2>"  txt="6"           et=""       typ=1   stck="<a2>"
at="<b>"   txt="79"           et="</b>"   typ=3   stck="<a2><b>"
at=""     txt="0"           et="</a2>"  typ=2   stck="<a2>"
```

## – Gleiche Daten, modifiziertes Lesen

```
$$ BUILD R_TABLE goon = " :<c>:"
$$ MODIFY ACCESS d R_TABLE -, -, -, goon
```

```
at="<a1>"  txt="1"           et=""       typ=1   stck="<a1>"
at="<b>"   txt="2<c>3</c>4" et="</b>"   typ=3   stck="<a1><b>"
at=""     txt="5"           et="</a1>"  typ=2   stck="<a1>"
at="<a2>"  txt="6"           et=""       typ=1   stck="<a2>"
at="<b>"   txt="7<c>8</c>9" et="</b>"   typ=3   stck="<a2><b>"
at=""     txt="0"           et="</a2>"  typ=2   stck="<a2>"
```

## Dateizugriffe – Daten mit definierten Strukturen

Diese Dateizugriffe erlauben das Beschreiben, Lesen und Durchsuchen von TUSTEP-Dateien mit strukturierten Daten. Dazu müssen die Daten in Einheiten eingeteilt sein, von denen jede eine eindeutige Nummer hat und einer definierten Datenstruktur entspricht. Eine Datei kann Einheiten mit gleichen oder unterschiedlichen Datenstrukturen enthalten.

### Nummerierung

Jede Einheit hat eine eindeutige, unveränderbare Nummer zwischen 1 und 999999. Die Nummer der jeweils aktuellen Einheit kann über eine Variable bestimmt bzw. abgefragt werden, deren Name in der ACCESS-Anweisung angegeben werden muss.

Genügt eine Nummerierung von 1 bis 999999 nicht, kann jeder dieser Hauptnummern eine Zusatznummer hinzugefügt werden. Die Zusatznummer kann Werte von 1 bis 999 annehmen. Der Variablenname für diese Zusatznummer muss in der ACCESS-Anweisung durch Punkt getrennt nach dem Variablennamen für die Hauptnummer angegeben werden. Damit steht für die Einheiten ein Nummernbereich von 1.1 bis 999999.999 zur Verfügung.

Werden die Daten mit anderen Programmen erstellt, so ist für die Nummerierung der Sätze folgendes zu beachten: Die Hauptnummer einer Einheit entspricht in der Datei der Seitennummer der Sätze. Werden für die Einheiten keine Zusatznummern verwendet, so müssen die einzelnen Zeilen einer Einheit mit 1 beginnend (bis max. 999) durchnummeriert sein; werden Zusatznummern verwendet, entspricht die Zusatznummer der Zeilennummer der Sätze und die einzelnen Zeilen einer Einheit müssen durch die Unterscheidungsnummer der Sätze bei /001 beginnend (bis max. /999) durchnummeriert sein.

### Datenstruktur

Die Daten jeder Einheit müssen einer definierten Datenstruktur entsprechen. Jede Datenstruktur hat einen frei wählbaren Namen. Der Strukturname der jeweils aktuellen Einheit kann über eine Variable bestimmt bzw. abgefragt werden, deren Name in der ACCESS-Anweisung angegeben werden muss.

Die Definition einer Datenstruktur besteht aus einzelnen Elementen:

```

$$ STRUCTURE strukturname
  Element_1
  Element_2 Element_3
  . . .
$$ ENDSTRUCTURE

```

Jedes Element muss einem von fünf möglichen Formaten entsprechen und gibt an, in welcher Weise der Inhalt einer Variablen in der Datei gekennzeichnet ist.

Für »normale« Variablen sind zwei Formate vorgesehen:

```
flag "anfang" variablenname "ende"  
flag "anfang" variablenname "trenner" ... "ende"
```

Für Sternvariablen sind ebenfalls zwei Formate vorgesehen:

```
flag "anfang1" "anfang2" variablenname "ende2" "ende1"  
* "anfang" variablenname "ende"
```

Für Zeilen mit einer Kennung, ohne variable Daten:

```
= "kennung"
```

Bestandteile eines Elements:

- `flag`: Minuszeichen, falls das Element nur in die Datei ausgegeben werden soll, wenn die zum Element gehörende Variable nicht leer ist; Pluszeichen, falls das Element unabhängig vom Inhalt der Variablen ausgegeben werden soll. Für das Lesen und Durchsuchen der Daten ist diese Angabe bedeutungslos, muss aber (außer beim ersten Element) immer angegeben werden.
- `anfang`: Anfangskennung, die in der Datei vor dem Inhalt der Variablen steht. Sie muss zwischen Anführungszeichen angegeben werden und darf nicht leer sein.
- `variablenname`: Name der Variablen, deren Inhalt in die Datei ausgegeben werden soll bzw. die die zwischen Anfangs- und Endekennung stehenden Daten aufnehmen soll.
- `ende`: Endekennung, die in der Datei nach dem Inhalt der Variablen steht. Sie muss zwischen Anführungszeichen angegeben werden und darf auch leer sein. Wenn sie leer ist, dürfen die Anführungszeichen weggelassen werden. Als Endekennung dient in diesem Fall die Anfangskennung des nächsten Elements bzw. das Zeilenende, falls kein weiteres Element in der gleichen Zeile folgt.
- `trenner`: Trennzeichenfolge; vor der Ausgabe soll jeder in der Variablen enthaltene Apostroph durch sie ersetzt werden, bei der Eingabe soll die Trennzeichenfolge durch Apostroph ersetzt werden. Sie muss zwischen Anführungszeichen angegeben werden und darf nicht leer sein.
- `...`: Drei Punkte (zur Unterscheidung der beiden Formate)
- `anfang1`: Anfangskennung, die in der Datei am Anfang der ersten Zeile der Sternvariablen steht. Sie muss zwischen Anführungszeichen angegeben werden und darf nicht leer sein.
- `anfang2`: Anfangskennung, die in der Datei am Anfang der zweiten bis letzten Zeile der Sternvariablen steht. Sie muss zwischen Anführungszeichen angegeben werden und darf auch leer sein.
- `ende2`: Endekennung, die in der Datei am Ende der ersten bis zweitletzten Zeile der Sternvariablen steht. Sie muss zwischen Anführungszeichen angegeben werden und darf auch leer sein.
- `ende1`: Endekennung, die in der Datei am Ende der letzten Zeile der Sternvariablen steht. Sie muss zwischen Anführungszeichen angegeben werden und darf nicht leer sein.

Für das erste Element einer Datenstruktur gelten drei Besonderheiten:

- Es muss bei der Ausgabe einer Einheit in die Datei in jedem Fall mit ausgegeben werden, bzw. es muss in jeder Einheit in der Datei vorhanden sein, auch wenn der Inhalt der dazugehörigen Variablen leer ist. Die Angabe eines Pluszeichens in der Definition der Datenstruktur entfällt für dieses Element.
- Die Anfangskennung ist zugleich die Strukturkennung. Dies bedeutet, dass beim Lesen einer Einheit auf Grund dieser Kennung (ohne Berücksichtigung der nachfolgenden Daten, die der Variablen zugeordnet werden oder als Endekennung dienen) entschieden wird, welcher Datenstruktur die Daten der gelesenen Einheit entsprechen. Für sämtliche in einem Makro definierten Datenstrukturen muss diese Kennung deshalb eindeutig sein.
- Falls in der Datei in der ersten Zeile der Struktur nur die Strukturkennung stehen soll bzw. steht, kann die Variable im ersten Element der Strukturdefinition weggelassen werden. Wird eine Variable angegeben, so darf sie keine Sternvariable sein, d. h. das erste Element muss einem der beiden Formate für »normale« Variablen entsprechen.

Die Anfangs- und Endekennungen müssen so gewählt werden, dass sie nicht mit Zeichenfolgen verwechselt werden können, die in den Daten vorkommen. Am sichersten ist es, wenn die Daten vor der Ausgabe in die Datei geprüft werden, ob sie Zeichenfolgen enthalten, die mit einer Kennung verwechselt werden können.

Eine Zeile in der Definition einer Datenstruktur entspricht auch einer Zeile in der Datei. Je Zeile dürfen ein oder mehrere Elemente für »normale« Variablen angegeben werden. Bei Zeilen für Sternvariablen ist es vom verwendeten Format abhängig. Beginnt die Zeile in der Strukturdefinition mit einem Plus- oder Minuszeichen, so darf nur ein einziges Element mit einer Sternvariablen angegeben werden. Beginnt die Zeile in der Strukturdefinition mit einem Stern, so dürfen mehrere Elemente angegeben werden; diese Elemente müssen aber mit einem Plus- oder Minuszeichen beginnen und alle Variablen müssen Sternvariablen sein.

Beispiele:

- Daten

```
** W123
*a Schmid, Paula; Müller, Peter
*t Die Dolomiten
*j 2019 *u 234
*i Wanderführer zu den
schönsten Gipfeln zwischen
Rosengarten und Drei Zinnen. *
```

## - Strukturdefinition

```

$$ STRUCTURE buch
    "** " signatur
  + "*a " autor "; " ...
  + "*t " titel
  + "*j " jahr + " *u " umfang
  + "*i " "" inhalt "" " *"
$$ ENDSTRUCTURE

```

## - Variablenbelegung

```

signatur = "W123"
autor    = "Schmid, Paula'Müller, Peter"
titel    = "Die Dolomiten"
jahr     = "2019"
umfang   = "234"
inhalt   = *
Wanderführer zu den
schönsten Gipfeln zwischen
Rosengarten und Drei Zinnen.

```

## - Daten

```

** 123
*n Schmid, Paula *p 72074 *o Tübingen
*n Müller, Peter *p 80012 *o München

```

## - Strukturdefinition

```

$$ STRUCTURE tabelle
    "** " nummer
  * "*n " name + " *p " plz + " *o " ort
$$ ENDSTRUCTURE

```

## - Variablenbelegung

```

nummer = "123"
name = *                plz = *                ort = *
Schmid, Paula          72074                Tübingen
Müller, Peter          80012                München

```

## - Daten

```

<eintrag>
  <vorname>Josef</vorname>
  <nachname>Maier</nachname>
  <wohnort>Trier</wohnort>
</eintrag>

```

### – Strukturdefinition

```

$$ STRUCTURE person
    "<eintrag>"
    + " <vorname>"  vname "</vorname>"
    + " <nachname>" nname "</nachname>"
    + " <wohnort>"  ort   "</wohnort>"
    = "</eintrag>"
$$ ENDSTRUCTURE

```

### – Variablenbelegung

```

vname = "Josef"
nname = "Maier"
ort    = "Trier"

```

## Definieren des Dateizugriffs

Der Zugriff auf Daten in Dateien mit den in diesem Kapitel beschriebenen Makroanweisungen muss mit der folgenden ACCESS-Anweisung begonnen und mit einer ENDACCESS-Anweisung (siehe Seite 597) beendet werden.

```

$$ ACCESS daten: modus/option "dateiname" nmmr, strktr, anzahl

```

### Bestandteile der ACCESS-Anweisung

- `daten`: Frei wählbarer Name für die Daten; er wird bei allen anderen Anweisungen für den Dateizugriff an Stelle eines Dateinamens verwendet, um anzugeben, auf welche Datei zugegriffen werden soll.
- Doppelpunkt
- `modus/option`: READ/STRUCTURES, falls von der nachfolgend angegebenen Datei nur gelesen werden soll; UPDATE/STRUCTURES, falls gelesen und geschrieben werden soll; WRITE/STRUCTURES, falls nur ans Ende der Datei geschrieben werden soll.

Bei den Modi UPDATE und WRITE kann zusätzlich die Option ERASE angegeben werden; in diesem Fall werden bei der Ausführung der ACCESS-Anweisung alle Daten in der Datei gelöscht. Die Option ERASE muss als erste Option bzw. unmittelbar nach der Option FILE (vgl. Seite 598) angegeben werden.

Am Ende der Angabe `option` kann durch einen Schrägstrich getrennt ein Zeitlimit angegeben werden. Die Bedeutung des Zeitlimits ist unter »Warten bis Dateizugriff möglich« auf Seite 598 beschrieben.

- `dateiname`: Name der Datei; er muss zwischen Anführungszeichen angegeben werden. Wird eine Variable angegeben, die den Dateinamen enthält, muss als erste Option FILE angegeben werden (vgl. Seite 598).
- `nmmr`: Name der Variablen, die die »aktuelle Nummer« enthält. Diese Variable wird von den nachfolgend beschriebenen Makroanweisungen ausgewertet oder evtl. geändert. Sollen Zusatznummern (s. o.) verwendet werden, müssen zwei durch einen Punkt getrennte Variablenamen angegeben werden.

- Komma
- `strktr`: Name der Variablen, die den »aktuellen Strukturnamen« enthält. Diese Variable wird von den nachfolgend beschriebenen Makroanweisungen ausgewertet oder evtl. geändert.
- Komma
- `anzahl`: Name der Variablen, die bei der Makroanweisung `FIND/SKIP` die Anzahl der zu übergehenden Einheiten und bei der Makroanweisung `COUNT` die Anzahl der gefundenen Einheiten angibt. Wird keine der beiden Makroanweisungen verwendet, kann die Variable `anzahl` einschließlich des davor stehenden Kommas weggelassen werden.

Bei der Ausführung der `ACCESS`-Anweisung wird der Inhalt der in der Anweisung angegebenen Variablen auf Null gesetzt bzw. gelöscht.

### Schreiben einer Einheit

Beim Schreiben einer Einheit wird diese der aktuellen Datenstruktur entsprechend aus Kennungen und in Variablen gespeicherten Daten zusammengestellt. Die aktuelle Datenstruktur ist jeweils die Struktur, deren Name in der dafür vorgesehenen Variablen (vgl. `ACCESS`-Anweisung) gespeichert ist.

```
$$ WRITE daten
```

Schreibt eine Einheit mit der aktuellen Nummer in die Datei. Gibt es in der Datei schon eine Einheit mit dieser Nummer, erfolgt eine Fehlermeldung und das Fehlerflag wird gesetzt.

```
$$ WRITE/NEXT daten
```

Wie `WRITE`, jedoch wird die aktuelle Nummer zuvor um 1 erhöht. Hat die aktuelle Nummer den Wert Null und stehen in der Datei schon Daten, wird zuvor die Nummer der letzten Einheit als aktuelle Nummer übernommen und diese um 1 erhöht.

```
$$ WRITE/UPDATE daten
```

Wie `WRITE`, jedoch muss die Einheit schon vorhanden sein; sie wird überschrieben.

### Lesen einer Einheit

Beim Lesen einer Einheit wird der Anfang der ersten Zeile mit der Anfangskennung des jeweils ersten Elements der momentan definierten Datenstrukturen verglichen, um festzustellen, welcher Datenstruktur die gelesenen Daten entsprechen. Dann werden die Daten dieser Datenstruktur entsprechend in die Variablen gespeichert. Die Nummer und der Strukturname der gelesenen Einheit werden in die vorgesehenen Variablen (vgl. `ACCESS`-Anweisung) gespeichert.

```
$$ READ daten
```

Liest die Einheit mit der aktuellen Nummer von der Datei. Gibt es in der Datei keine Einheit mit dieser Nummer, erfolgt eine Fehlermeldung und das Fehlerflag wird gesetzt.



```
$$ READ/NEXT daten
```

Wie READ, jedoch wird die Einheit gelesen, die in der Datei nach der mit der aktuellen Nummer (mit der nächst höheren Nummer) folgt. Gibt es in der Datei keine nachfolgende Einheit, wird eine Null bzw. eine leere Zeichenfolge in die für Nummer bzw. Strukturname vorgesehenen Variablen (vgl. ACCESS-Anweisung) gespeichert.

```
$$ READ/PREVIOUS daten
```

Wie READ, jedoch wird die Einheit gelesen, die in der Datei vor der mit der aktuellen Nummer (mit der nächst niedrigeren Nummer) steht. Gibt es in der Datei keine vorangehende Einheit, wird eine Null bzw. eine leere Zeichenfolge in die für Nummer bzw. Strukturname vorgesehenen Variablen (vgl. ACCESS-Anweisung) gespeichert.

Bei READ-Anweisungen mit Option der NEXT oder PREVIOUS kann noch zusätzlich die Option EXIT angegeben werden:

```
$$ READ ... /EXIT daten
```

Dies ist jedoch nur sinnvoll, wenn die READ-Anweisung innerhalb einer Schleife (zwischen LOOP und ENDOOP) ausgeführt wird. Die Option EXIT bewirkt, dass automatisch eine EXIT-Anweisung ausgeführt wird (d. h. dass die Schleife automatisch verlassen wird), wenn keine Einheit mehr vorhanden war.

## Prüfen einer Einheit

Beim Prüfen einer Einheit wird nur geprüft, ob die Einheit vorhanden ist und ggf. der Anfang der ersten Zeile mit der Anfangskennung des jeweils ersten Elements der momentan definierten Datenstrukturen verglichen, um festzustellen, welcher Datenstruktur die gelesenen Daten entsprechen. Es werden jedoch keine Daten in die Variablen dieser Datenstruktur gespeichert.

```
$$ READ/CHECK daten
```

Prüft, ob die Einheit mit der aktuellen Nummer in der Datei vorhanden ist. Wenn ja, wird der Strukturname der geprüften Einheit in die vorgesehene Variable (vgl. ACCESS-Anweisung) gespeichert. Wenn nicht, wird eine leere Zeichenfolge in diese Variable gespeichert. Ob die Texteinheit vorhanden ist, kann auch mit einer nachfolgenden IF-Anweisung mit der Bedingung EXIST (siehe Seite 476) abgefragt werden.

## Suchen einer bestimmten Einheit

Beim Suchen von Einheiten werden alle Zeilen (Kennungen und Daten) einer Einheit zu einer einzigen Zeichenfolge zusammengefasst; dann wird geprüft, ob diese Zeichenfolge die Suchbedingungen erfüllt. Die Suchbedingungen sind ab Seite 599 beschrieben.

```
$$ FIND daten (suchbed1) WHILE (suchbed2) UPTO nmmr
```

Sucht eine Einheit, die die angegebenen Suchbedingungen erfüllt. Die Suche beginnt mit der Einheit mit der aktuellen Nummer und erfolgt zum Dateiende hin. Gesucht wird nur so lange, wie die durchsuchten Einheiten die hinter WHILE angegebenen

Suchbedingungen erfüllen, höchstens jedoch bis vor den Eintrag mit der Nummer, die hinter dem Schlüsselwort `UPTO` angegeben ist. Sowohl die Angabe `»WHILE (suchbed2)«` als auch die Angabe `»UPTO nmmr«` können weggelassen werden, wenn die entsprechende Einschränkung der Suche nicht erforderlich ist.

Wird eine Einheit gefunden, wird diese (wie mit `READ`) eingelesen; wird keine gefunden, wird eine Null bzw. eine leere Zeichenfolge in die für Nummer bzw. Strukturname vorgesehenen Variablen (vgl. `ACCESS`-Anweisung) gespeichert.

Durch Angabe einer der folgenden drei Optionen kann die Einheit, mit der die Suche beginnen soll, und die Suchrichtung anders festgelegt werden:

```
$$ FIND/NEXT daten (suchbedingungen) ...
```

Die Suche beginnt hinter der Einheit mit der aktuellen Nummer und erfolgt zum Dateiende hin.

```
$$ FIND/REVERSE daten (suchbedingungen) ...
```

Die Suche beginnt mit der Einheit mit der aktuellen Nummer und erfolgt zum Dateianfang hin.

```
$$ FIND/PREVIOUS daten (suchbedingungen) ...
```

Die Suche beginnt vor der Einheit mit der aktuellen Nummer und erfolgt zum Dateianfang hin.

Bei allen oben aufgeführten `FIND`-Anweisungen können noch zusätzliche Optionen in der folgenden Reihenfolge angegeben werden:

```
$$ FIND ... /SKIP daten (suchbedingungen) ...
```

Die Option `SKIP` bewirkt, dass eine vorgegebenen Anzahl von Einheiten, die die Suchbedingungen erfüllen, übergangen werden. Die Anzahl wird aus der dafür vorgesehenen Variablen (siehe `ACCESS`-Anweisung) entnommen. Enthält diese Variable eine Null, so ist die Option `SKIP` wirkungslos.

```
$$ FIND ... /READ daten (suchbedingungen) ...
```

Die Option `READ` besagt, dass die gefundene Einheit automatisch (wie mit der `READ`-Anweisung) eingelesen wird. Diese Option ist voreingestellt und kann mit der nachfolgend beschriebenen Option aufgehoben werden.

```
$$ FIND ... /CHECK daten (suchbedingungen) ...
```

Die Option `CHECK` bewirkt, dass die gefundene Einheit nicht (wie mit der `READ`-Anweisung) eingelesen wird. Von der gefundenen Einheit werden nur die Nummer und der Strukturname in die dafür vorgesehenen Variablen (vgl. `ACCESS`-Anweisung) gespeichert.

Bei allen `FIND`-Anweisungen kann noch zusätzlich die Option `EXIT` angegeben werden:

```
$$ FIND ... /EXIT daten
```

Dies ist jedoch nur sinnvoll, wenn die `FIND`-Anweisung innerhalb einer Schleife (zwischen `LOOP` und `ENDLOOP`) ausgeführt wird. Die Option `EXIT` bewirkt, dass automatisch eine `EXIT`-Anweisung ausgeführt wird (d. h. dass die Schleife automatisch verlassen wird), wenn keine Einheit gefunden wurde.

## Zählen bestimmter Einheiten

Die Anweisung `COUNT` zum Zählen bestimmter Einheiten entspricht (einschließlich der möglichen Optionen) der Anweisung `FIND` zum Suchen einer bestimmten Einheit, die auf Seite 633 beschrieben ist.

```
$$ COUNT daten (suchbed1) WHILE (suchbed2) UPTO nmmr
```

Es wird genau derselbe Datenbereich wie bei der `FIND`-Anweisung durchsucht. Jedoch wird nicht nur die erste Einheit gesucht, die die Suchbedingungen erfüllt, sondern festgestellt, wieviele Einheiten insgesamt die Suchbedingung erfüllen.

Die letzte gefundene Einheit wird (wie mit `READ`) eingelesen, falls nicht die Option `CHECK` angegeben ist; wird keine gefunden, wird eine Null bzw. eine leere Zeichenfolge in die für Nummer bzw. Strukturname vorgesehenen Variablen (vgl. `ACCESS`-Anweisung) gespeichert.

## Löschen einer Einheit

```
$$ ERASE daten
```

Löscht in der Datei die Einheit mit der aktuellen Nummer.

## Beispiel für Dateizugriffe

```

$$ - Dieses Makro gibt strukturierte Daten in eine Datei
$$ - aus und durchsucht anschließend diese Daten.
$$ - Es ist in dieser Form jedoch nur als Beispiel für die
$$ - Anwendung der oben beschriebenen Anweisungen sinnvoll.
$$ -
$$! datei
$$-
$$ MODE {}
$$ -
$$ - Struktur der Daten festlegen
$$ -
$$ STRUCTURE test
    "*n " nname - ", " vname
    - "*p " plz " " + "*o " ort
$$ ENDSTRUCTURE
$$ -
$$ - Zugriff auf Datei beginnen
$$ -
$$ ACCESS daten: UPDATE/FILE/STRUCTURES datei num, str, anz
$$ -
$$ - Name der Datenstruktur bestimmen
$$ -
$$ SET str = "test"
$$ -
$$ - Variablen definieren und erste Einheit ausgeben
$$ -
$$ SET nname="Schmid", vname="Paula", plz=72074, ort="Tübingen"
$$ -
$$ WRITE/NEXT daten
$$ -
$$ - Variablen definieren und zweite Einheit ausgeben
$$ -
$$ SET nname="Müller", vname="Peter", plz=80012, ort="München"
$$ -
$$ WRITE/NEXT daten
$$ -
$$ - In der Datei stehen jetzt folgende Daten:
$$ -
$$ - 1.1 *n Schmid, Paula
$$ - 1.2 *p 72074 *o Tübingen
$$ - 2.1 *n Müller, Peter
$$ - 2.2 *p 80012 *o München
$$ -
$$ - Nun nach bestimmten Ortsangaben suchen.
$$ - Kennzeichen für die Ortsangaben ist "*o ".
$$ - Die Ortsangabe geht bis zum Strukturende.

```

```

$$ - Deshalb genügt eine Such-Tabelle, um den
$$ - Anfang der Ortsangabe zu bestimmen.
$$ -
$$ BUILD S_TABLE ort_anf = ":\*o :"
$$ -
$$ - Akzente ignorieren, Sonderbuchstaben wie Grundbuchstaben,
$$ - ä, ö, ü und ß wie ae, oe, ue und ss behandeln
$$ -
$$ BUILD X_TABLE norm = ":%{1--2}{%}::#...ä:ae:ö:oe:ü:ue:ß:ss:"
$$ -
$$ - Gesucht werden soll "München"
$$ -
$$ SET suche = "München"
$$ -
$$ - Auch die Suchzeichenfolge muss normiert werden
$$ -
$$ Set suche_norm = EXCHANGE (suche, norm)
$$ -
$$ - Suchzeichenfolge wird als Such- oder Recherchier-Tabelle
benötigt
$$ -
$$ BUILD R_TABLE suchen = "/<suche_norm>/"
$$ -
$$ - Suche am Anfang der Daten beginnen
$$ -
$$ SET num = 0
$$ -
$$ FIND/NEXT daten (ort_anf,-,-,norm; suchen; -; -)
$$ -
$$ IF (num.NE.0) THEN
$$   + Einheit mit der Nummer <num> gefunden und die
$$   + Daten in die entsprechenden Variablen übertragen.
$$ ELSE
$$   + Keine Einheit mit "München" gefunden.
$$ ENDIF
$$ -
$$ - Weitersuchen
$$ -
$$ FIND/NEXT daten (ort_anf,-,-,norm; suchen; -; -)
$$ -
$$ IF (num.NE.0) THEN
$$   + Hier wäre eine weitere Einheit gefunden worden.
$$   + Kann nur der Fall sein, wenn die Datei vor dem Aufruf
$$   + dieses Makros schon entsprechende Daten enthält.
$$   + (Wenn dieses Makro z. B. mehrmals aufgerufen wird.)
$$ ELSE
$$   + Keine weitere Einheit mit "München" gefunden.
$$ ENDIF

```

```
$$ -  
$$ - Zugriff auf die Datei beenden  
$$ -  
$$ ENDACCESS daten
```

## Datei-Transfer

### Import einer Datei in die aktuelle TUSTEP-Sitzung

Die Makroanweisung

```
$$ UPLOAD/option "dateiname" FROM "pfad"
```

überträgt die Daten aus der mit »pfad« angegebenen Datei in die Datei mit dem Namen »dateiname«.

Die Datei »dateiname« muss zum Schreiben angemeldet sein. Wenn sie bereits Daten enthält, werden diese überschrieben. Die Angabe »pfad« muss den vollständigen Pfad samt Dateiname einer existierenden Datei enthalten. Sie braucht nicht angemeldet zu sein.

Ob die Datei übertragen werden konnte, kann in einer nachfolgenden IF-Anweisung mit der Bedingung DONE (siehe Seite 476) abgefragt werden.

Die Daten werden unverändert (binär) übertragen. Wenn jedoch in einer REMOTE-Sitzung eine Fremd-Datei von einem Rechner unter Windows auf einen Rechner unter Linux übertragen wird, werden die Trennzeichen zwischen den Datensätzen dem im jeweiligen Betriebssystem dafür üblichen Code angepasst: Die Codes 0D0A werden durch die Codes 0A ersetzt.

Nach dem Schlüsselwort UPLOAD können jeweils durch Schrägstrich getrennt Optionen angegeben werden:

– BINARY

Datei binär, d. h. unverändert übertragen.

– QUIET

Keine Meldung nachdem die Datei übertragen wurde.

Beispiel: `$$ UPLOAD/QUIET "testdaten" FROM "a:\testdaten"`

### Export einer Datei aus der aktuellen TUSTEP-Sitzung

Die Makroanweisung

```
$$ DOWNLOAD/option "dateiname" TO "pfad"
```

überträgt die Daten aus der Datei mit dem Namen »dateiname« in die mit »pfad« angegebene Datei.

Die Datei »dateiname« muss zum Lesen oder zum Schreiben angemeldet sein. Die Angabe »pfad« muss den vollständigen Pfad samt Dateiname für eine Datei enthalten. Sie braucht nicht angemeldet zu sein. Falls sie schon existiert, wird gefragt, ob sie überschrieben werden darf.

Ob die Datei übertragen worden ist, kann in einer nachfolgenden IF-Anweisung mit der Bedingung DONE (siehe Seite 476) abgefragt werden.

Die Daten werden unverändert (binär) übertragen. Wenn jedoch in einer REMOTE-

Sitzung eine Fremd-Datei von einem Rechner unter Linux auf einen Rechner unter Windows übertragen wird, werden die Trennzeichen zwischen den Datensätzen dem im jeweiligen Betriebssystem dafür üblichen Code angepasst: Der Code 0A wird durch die Codes 0D0A ersetzt.

Nach dem Schlüsselwort `DOWNLOAD` können jeweils durch Schrägstrich getrennt zusätzliche Optionen angegeben werden:

– `BINARY`

Datei binär, d. h. unverändert übertragen.

– `REPLACE`

Falls die mit »pfad« angegebene Datei schon existiert, darf sie ohne Rückfrage überschrieben werden.

– `QUIET`

Keine Meldung nachdem die Datei übertragen wurde.

– `PREVIEW`

Die zu übertragende Datei ist mit dem Kommando `#DRUCKE` beschrieben worden, wobei zur Spezifikation `DATEI` diese Datei und zur Spezifikation `TYP` entweder `WIN-10` oder `WIN-12` angegeben wurde. Nach dem Übertragen der Datei wird der Inhalt der Datei auf dem Bildschirm angezeigt.

– `BROWSE`

Nach dem Übertragen der Datei, wird auf dem Windows-Rechner die Anwendung, die für die Endung der Datei im Betriebssystem definiert ist, gestartet.

Beispiel `$$ DOWNLOAD/QUIET "testdaten" TO "a:\testdaten"`



## Makrofenster

Für die Interaktion mit dem Benutzer können außer ASK-Anweisungen auch Makrofenster mit Eingabefeldern verwendet werden.

Für die häufigsten Standardfälle stehen die Makrofunktionen DISPLAY, ASK, EDIT und CLICK zur Verfügung; sie sind ab Seite 581 beschrieben.

### Anzeigen eines Makrofensters

Bevor Makrofenster angezeigt werden können, müssen sie definiert werden. Erst danach können sie mit der Makroanweisung

```
$$ DISPLAY fenstername
```

im TUSTEP-Fenster angezeigt werden. Die in der Definition eines Makrofensters enthaltenen Makroanweisungen werden noch nicht bei der Definition des Makrofensters ausgeführt, sondern erst, wenn das Makrofenster angezeigt wird.

Soll das Makrofenster nicht in der Mitte des TUSTEP-Fensters angezeigt werden, kann zusätzlich die Position für die linke obere Ecke des Makrofensters angegeben werden:

```
$$ DISPLAY fenstername zeile:spalte
```

Sowohl für die Zeilen- als auch für die Spaltenangabe gilt folgendes: 0 (Null) bedeutet, dass das Makrofenster in die Mitte des TUSTEP-Fensters bzw. des gerade angezeigten Makrofensters positioniert werden soll. Eine Zahl ohne Vorzeichen gilt als Absolutposition innerhalb des TUSTEP-Fensters; eine Zahl mit Vorzeichen gilt als Relativposition bezüglich der linken oberen Ecke des gerade aktiven Feldes (nicht Fensters) bzw. der Mitte des TUSTEP-Fensters, falls noch kein Makrofenster angezeigt wird.

### Definition eines Makrofensters

```
$$ WINDOW fenstername: [] fensterfarbe lstfeld,aktfeld, nxtfeld
$$ ...
    Fensterinitialisierung
$$ ...
$$ FIELD name_1: typ_1 farbe_1 txtvar_1, numvar_1, posvar_1
$$     Makroanweisungen für Feld 1
$$ FIELD name_2: typ_2 farbe_2 txtvar_2, numvar_2, posvar_2
$$     Makroanweisungen für Feld 2
$$ ...
$$ ...
$$ FIELD name_n: typ_n farbe_n txtvar_n, numvar_n, posvar_n
$$     Makroanweisungen für Feld n
$$ ENDWINDOW
```

### Bestandteile der WINDOW-Anweisung

- `fenstername`: Frei wählbarer Name des Makrofensters; er dient zur Identifikation des Makrofensters in der `DISPLAY`-Anweisung.
- Doppelpunkt
- Klammerpaar, mit dem in den Daten der Fensterinitialisierung die Felder begrenzt werden; möglich sind folgende Klammerpaare: `[]`, `()`, `<>` und `{}`. Es dürfen jedoch nicht die Klammern verwendet werden, die gerade für die Variablensetzung eingestellt sind.
- `fensterfarbe`: Hexadezimal-Code der Farbe des Makrofensters; die möglichen Codes können der Tabelle entnommen werden, die im Editor durch die Tastenkombination `Ctrl+F` bzw. `Strg+F` angezeigt wird.
- `1stfeld`: Frei wählbarer Name der Variablen, in der jeweils automatisch der Name des zuvor/zuletzt aktiven Feldes gespeichert wird.
- Komma
- `aktfeld`: Frei wählbarer Name der Variablen, in der jeweils automatisch der Name des gerade aktiven Feldes gespeichert wird.
- Komma
- `nxtfeld`: Frei wählbarer Name der Variablen, in der die `EDIT`-Anweisung jeweils automatisch den Name des Feldes gespeichert, das als nächstes aktiviert wird, falls nicht durch eine nachfolgende Anweisung explizit etwas anderes bestimmt wird.

### Ändern der Vorgaben der WINDOW-Anweisung

Während das Makrofenster angezeigt wird, kann die Farbe des Fensters mit folgender Anweisung geändert werden:

```
$$ MODIFY WINDOW COLOR fensterfarbe
```

Dabei ist für `fensterfarbe` die Farbe des Fensters in der gleichen Form wie in der `WINDOW`-Anweisung anzugeben.

### Fensterinitialisierung

Die Fensterinitialisierung muss die Daten für das Makrofenster enthalten. Sie bestehen aus beschreibendem Text und aus Feldern, in die z. B. Daten eingetragen werden können, nachdem das Makrofenster mit der `DISPLAY`-Anweisung angezeigt worden ist. Anfangs- und Endposition eines Feldes werden durch die öffnende bzw. schließende Klammer des in der `WINDOW`-Anweisung angegebenen Klammerpaares bestimmt. Bei mehrzeiligen Feldern müssen die öffnenden bzw. die schließenden Klammern in jeder Zeile in der gleichen Spalte angegeben werden. Zwischen den Klammern muss jeweils der Name des Feldes angegeben werden; er ist frei wählbar. Die Felder werden über diesen Namen in den `FIELD`-Anweisungen identifiziert. In den `FIELD`-Anweisungen werden die Eigenschaften der Felder festgelegt.

Die Fensterinitialisierung kann außer den Daten für das Makrofenster auch noch Makroanweisungen enthalten. Sie werden von der `DISPLAY`-Anweisung ausgeführt, bevor das Makrofenster angezeigt wird. Mit Auswahlanweisungen können z. B. nur bestimmte Daten für das Makrofenster ausgewählt werden. Dadurch ist es möglich, das Aussehen des Makrofensters erst unmittelbar vor der Anzeige endgültig festzulegen.

Für jedes Feld, das in den Daten der Fensterinitialisierung vorkommt, muss eine `FIELD`-Anweisung angegeben werden.

#### Bestandteile der `FIELD`-Anweisung

- `fieldname`: Name des Feldes
- Doppelpunkt
- `feldtyp`: Typ des Feldes; die möglichen Feldtypen sind weiter unten beschrieben. Hinter der Angabe des Feldtyps können jeweils durch einen Schrägstrich getrennt Optionen angegeben werden. Sie bestimmen die Eigenschaften des Feldes genauer und sind jeweils bei den einzelnen Feldtypen beschrieben. Bei allen Feldtypen, außer bei Feldtyp `DUMMY`, kann als letzte eine der Optionen `ON` oder `OFF` angegeben werden. Mit diesen Optionen wird der Status festgelegt. Er gibt an, ob das Feld im Makrofenster angezeigt wird (`ON`, Voreinstellung) oder nicht (`OFF`).
- `feldfarbe`: Hexadezimal-Codes von Farben; die möglichen Codes können der Tabelle entnommen werden, die im Editor durch die Tastenkombination `Ctrl+F` bzw. `Strg+F` angezeigt wird. Es können ein bis sechs durch Minuszeichen getrennte Farbwerte angegeben werden. Der erste Farbwert wird für normalen Text verwendet, der zweite für hervorgehobenen. Sind mehr als zwei Farbwerte angegeben, so wird das Feld auch im Status `OFF` angezeigt und dabei der dritte und vierte Farbwert für normalen bzw. hervorgehobenen Text verwendet. Sind mehr als vier Farbwerte angegeben, so wird der normale Text in zeilenweise abwechselnden Farben angezeigt und dabei im Status `ON` für jede zweite Zeile statt des ersten Farbwertes der fünfte Farbwert und im Status `OFF` für jede zweite Zeile statt des dritten Farbwertes der sechste Farbwert verwendet.
- `txtvar`: Name einer Variablen, die Text enthält. Dieser Text wird im jeweiligen Feld angezeigt. Bei Feldern vom Typ `FLAGS` entfällt diese Variable (einschließlich des danach stehenden Kommas).
- Komma
- `numvar`: Name einer Variablen, die einen Zahlenwert enthält. Die Bedeutung dieses Zahlenwertes ist vom Typ des Feldes abhängig. Bei Feldern vom Typ `BUTTON` entfällt diese Variable (einschließlich des davor stehenden Kommas).  
Bei manchen Feldtypen gibt der Zahlenwert die Cursor-Position an. Die Cursor-Position ist die laufende Nummer des entsprechenden Zeichens in der Variablen `txtvar`. Bei Sternvariablen wird der Zeilenwechsel als ein Zeichen gezählt; um aus Zeilen- und Zeichennummer die entsprechende Cursor-Position und umgekehrt zu ermitteln stehen die Makrofunktionen `CURSOR_POSITION`, `CURSOR_ROW` und `CURSOR_COLUMN` (siehe Seite 592) zur Verfügung.
- Komma
- `posvar`: Name einer Variablen, die einen Zahlenwert enthält. Mit ihr kann vorgegeben werden, die wievielte Teilzeichenfolge der Variablen `txtvar` in der ersten Zeile des Feldes angezeigt wird. Bei Feldern vom Typ `FLAGS` und `BUTTON` entfällt diese Variable (einschließlich des davor stehenden Kommas), bei allen anderen Feldern ist sie optional.  
Wird für `posvar` bei zwei oder mehreren Feldern dieselbe Variable angegeben, so werden diese Felder ggf. parallel verschoben; d. h. wenn der Inhalt eines dieser Felder durch Steuerbefehlen vertikal verschoben wird, werden alle Inhalte dieser Felder in gleicher Weise nach oben bzw. nach unten verschoben.

### Ändern der Vorgaben der FIELD-Anweisung

Während das Makrofenster angezeigt wird, kann der Status eines Feldes mit folgender Anweisung geändert werden:

```
$$ MODIFY FIELD feldname status
```

Dabei ist für `feldname` der Name des Feldes und für `status` entweder ON oder OFF anzugeben.

Die Farbe eines Feldes kann ebenfalls geändert werden, während das Makrofenster angezeigt wird:

```
$$ MODIFY FIELD feldname COLOR feldfarbe
```

Dabei ist für `feldname` der Name des Feldes und für `feldfarbe` die Farbe des Feldes in der gleichen Form wie in der FIELD-Anweisung anzugeben.

## Makroanweisungen für die einzelnen Felder

Die für die einzelnen Felder angegebenen Makroanweisungen werden abgearbeitet, sobald das jeweilige Feld aktiviert, d. h. der Cursor in das Feld gebracht wird. Felder vom Typ OUTPUT und OUTPUT/CLICK können (im Gegensatz zu allen anderen) den Cursor nicht aufnehmen und werden übergangen.

Nach der Initialisierung des Makrofensters wird automatisch das erste Feld aktiviert, das den Cursor aufnehmen kann. Soll ein anderes Feld aktiviert werden, muss innerhalb der Fensterinitialisierung der in der WINDOW-Anweisung angegebenen Variablen `aktfeld` der Name des zu aktivierenden Feldes zugewiesen werden.

Für jedes Feld, das den Cursor aufnehmen kann, muss mindestens die Makroanweisung

```
$$ EDIT oder $$ EDIT/option
```

angegeben werden. Wenn ein Feld aktiviert wird, werden alle Anweisungen bis zur EDIT-Anweisung abgearbeitet. Dann werden alle Felder entsprechend dem momentanen Inhalt der zugehörigen Variablen `txtvar`, `numvar` und `posvar` aktualisiert. Anschließend werden Tastatureingaben und Mausklicks verarbeitet, bis die Eingabe für das Feld abgeschlossen und damit die EDIT-Anweisung beendet wird. Danach werden die auf die EDIT-Anweisung folgenden Anweisungen abgearbeitet.

Bei bestimmten Feldern können mit der EDIT-Anweisung noch Optionen angegeben werden. Welche Optionen möglich sind und welche Wirkung sie haben, ist jeweils in der Beschreibung dieser Feldtypen angegeben.

Ob während der Ausführung der EDIT-Anweisung der Feldinhalt verändert wurde, kann mit einer nachfolgenden IF-Anweisung mit der Bedingung MODIFIED (siehe Seite 474) abgefragt werden.

Neben der EDIT-Anweisung gibt es noch drei weitere Makroanweisungen, die nur innerhalb der Anweisungen für Felder angegeben werden können:

```
$$ FINISH "Zeichenfolge"
```

schließt das Makrofenster und speichert die angegebene Zeichenfolge in die Variable `1stfeld`, deren Name in der `WINDOW`-Anweisung angegeben ist. Die Zeichenfolge (einschließlich der Anführungszeichen) kann weggelassen werden; in diesem Fall wird der Name des zuletzt aktiven Makrofeldes in diese Variable gespeichert.

```
$$ GOTO FIELD feldname
```

beendet das Abarbeiten der Anweisungen im aktiven Feld und aktiviert im selben Makrofenster das Feld mit dem angegebenen Namen.

```
$$ GOTO WINDOW fensternamen
```

beendet das Abarbeiten der Anweisungen des aktuellen Feldes und schließt das Makrofenster. Dann wird das Fenster mit dem angegebenen Namen initialisiert und angezeigt. Eine Rückkehr ist durch eine entsprechende `GOTO-WINDOW`-Anweisung möglich; das Makrofenster wird dann wieder initialisiert.

Das Wechseln in ein anderes Makrofenster kann auch mit der `DISPLAY`-Anweisung erfolgen. Damit wird das Abarbeiten der Makroanweisungen des aktuellen Makrofensters unterbrochen und das in der `DISPLAY`-Anweisung angegebene Makrofenster angezeigt. Nach Beenden dieses Fensters wird das vorangehende Makrofenster (ohne erneute Initialisierung) wieder angezeigt und das Abarbeiten der Makroanweisungen nach der `DISPLAY`-Anweisung fortgesetzt. In diesem Fall muss jedoch entweder mit Hilfe von `LOOP` und `ENDLOOP` zur `EDIT`-Anweisung zurückgesprungen werden oder durch Ausführen einer `GOTO-FIELD`-Anweisung bestimmt werden, welches Feld als nächstes aktiviert wird. Geschieht beides nicht, führt dies zu einer Fehlermeldung, nachdem alle Anweisungen des Feldes abgearbeitet sind.

Wenn alle Anweisungen für ein Feld abgearbeitet sind, ohne dass eine `GOTO`- oder `FINISH`-Anweisung ausgeführt wurde, wird in der Regel das nächstfolgende Feld, das den Cursor aufnehmen kann, aktiviert. Dies gilt insbesondere auch, wenn die Eingabe mit `TAB` abgeschlossen wird. Die Reihenfolge, in der die Felder dabei aktiviert werden, ist durch die Reihenfolge vorgegeben, in der die Felder mit `FIELD`-Anweisungen definiert sind (nicht durch die Reihenfolge, in der sie in der Fensterinitialisierung angegeben sind). Von dieser Regel gibt es folgende Ausnahmen:

- Wenn die Eingabe mit `BACKTAB` abgeschlossen wurde, wird das nächstvorhergehende Feld, das den Cursor aufnehmen kann, aktiviert.
- Wenn die Eingabe mit `ENTER` abgeschlossen wurde und das Feld nicht vom Typ `BUTTON/ENTER` ist, wird das Feld mit dem Typ `BUTTON/ENTER` aktiviert. Falls kein solches Feld definiert ist, oder das Feld selbst vom Typ `BUTTON/ENTER` ist, wird das Makrofenster geschlossen.
- Wenn die Eingabe mit `CANCEL` abgeschlossen wurde und das Feld nicht vom Typ `BUTTON/CANCEL` ist, wird das Feld mit dem Typ `BUTTON/CANCEL` aktiviert. Falls kein solches Feld definiert ist, oder das Feld selbst vom Typ `BUTTON/CANCEL` ist, wird das Makrofenster geschlossen.
- Wenn die Eingabe mit `HELP` abgeschlossen wurde und das Feld nicht vom Typ `BUTTON/HELP` ist, wird das Feld mit dem Typ `BUTTON/HELP` aktiviert. Falls kein solches Feld definiert ist, oder das Feld selbst vom Typ `BUTTON/HELP` ist, wird das Makrofenster geschlossen.
- Wenn die Eingabe mit einer Funktionstaste abgeschlossen wurde und das Feld nicht vom Typ `BUTTON/FKEY` ist, wird das Feld mit dem Typ `BUTTON/FKEY` akti-

- viert. Falls kein solches Feld definiert ist, oder das Feld selbst vom Typ `BUTTON/FKEY` ist, wird das Makrofenster geschlossen.
- Wenn die Eingabe durch Anklicken eines anderen Feldes, das den Cursor aufnehmen kann, abgeschlossen wurde, wird das angeklickte Feld aktiviert.
  - Wenn die Eingabe durch Anklicken eines Feldes vom Typ `OUTPUT/CLICK` abgeschlossen wurde, wird das zum angeklickten Feld gehörende Feld vom Typ `FLAGS` aktiviert.
  - Wenn die Eingabe mit `CR` abgeschlossen wurde und das Feld vom Typ `BUTTON/ENTER`, `BUTTON/CANCEL`, `BUTTON/HELP`, `BUTTON/FKEY` oder vom Typ `BUTTON` ist, wird das Makrofenster geschlossen.

## Feldtypen, ihre Eigenschaften und Steuerbefehle

Die Steuerbefehle `TAB`, `BACKTAB`, `ENTER`, `CANCEL`, `HELP` und die Funktionstasten (`F1` bis `F60`) schließen in allen Feldern die Eingabe ab. Ihre Besonderheiten sind oben beschrieben.

- **INPUT** (= `textbox`)

Dieses Feld dient zum Eingeben und/oder Verändern von Text. Die Variable `txtvar` enthält den Text. Je Zeile des Feldes wird eine Teilzeichenfolge des Textes angezeigt. Die Variable `numvar` enthält die Cursor-Position innerhalb des in der Variablen `txtvar` stehenden Textes.

Die Variable `txtvar` kann bei diesem Feldtyp auch eine Sternvariable sein. In diesem Fall werden die Zeilen der Sternvariablen wie die Teilzeichenfolgen einer »normalen« Variablen behandelt.

Bei der zu diesem Feld gehörenden `EDIT`-Anweisung kann mit der Option `INSERT` bzw. `REPLACE` angegeben werden, ob der Einfügemodus oder der Überschreibmodus eingestellt werden soll. Wird keine dieser Optionen angegeben, bleibt der Modus eingestellt, der zuletzt in einem Feld des Typs `INPUT` eingestellt wurde.

`CUR_LE` (Pfeil nach links)

Positioniert den Cursor um ein Zeichen nach links; bei Feldern mit der Option `SHIFT` (s. u.): falls der Cursor ganz links im Feld steht und noch weiterer Text links vor dem angezeigten vorhanden ist, wird der Inhalt des Feldes um eine Position nach rechts verschoben.

`CUR_RI` (Pfeil nach rechts)

Positioniert den Cursor um ein Zeichen nach rechts; bei Feldern mit der Option `SHIFT` (s. u.): falls der Cursor ganz rechts im Feld steht und noch weiterer Text rechts nach dem angezeigten vorhanden ist, wird der Inhalt des Feldes um eine Position nach links verschoben.

`CUR_DN` (Pfeil nach unten)

Positioniert den Cursor um eine Zeile nach unten; bei Feldern mit den Optionen `SCROLL` oder `EDIT` (s. u.): falls der Cursor in der untersten Zeile des Feldes steht und noch weiterer Text nach dem in dieser Zeile angezeigten vorhanden ist, wird der Inhalt des Feldes um eine Zeile nach oben verschoben

und eine weitere Zeile des nachfolgenden Textes in der letzten Zeile des Feldes angezeigt.

**CUR\_UP** (Pfeil nach oben)

Positioniert den Cursor um eine Zeile nach oben; bei Feldern mit den Optionen **SCROLL** oder **EDIT** (s. u.): falls der Cursor in der obersten Zeile des Feldes steht und noch weiterer Text vor dem in dieser Zeile angezeigten vorhanden ist, wird der Inhalt des Feldes um eine Zeile nach unten verschoben und eine weitere Zeile des vorangehenden Textes in der ersten Zeile des Feldes angezeigt.

**SHW\_DN** (Bild runter, PgDn)

Bei Feldern mit der Option **SCROLL** oder **EDIT** (s. u.): Blättert vorwärts weiter; falls der Cursor nicht in der obersten Zeile steht, wird danach die Zeile, in der er steht, als oberste Zeile angezeigt.

**SHW\_UP** (Bild rauf, PgUp)

Bei Feldern mit der Option **SCROLL** oder **EDIT** (s. u.): Blättert rückwärts weiter; falls der Cursor nicht in der untersten Zeile steht, wird danach die Zeile, in der er steht, als unterste Zeile angezeigt.

**SKP\_RI** (Minus)

Positioniert den Cursor auf das erste Zeichen des nachfolgenden Wortes.

**SKP\_LE** (Ctrl+Pfeil nach links), (Strg+Pfeil nach links)

Positioniert den Cursor auf das erste Zeichen des vorangehenden Wortes.

**SKP\_BEG** (Pos1, Home)

Positioniert den Cursor auf das erste Zeichen der aktuellen Zeile.

**SKP\_END** (Ende, End)

Positioniert den Cursor hinter das letzte Zeichen der aktuellen Zeile.

**DEL** (Entf, Del, Shift+Backspace, Shift+Rücktaste)

Löscht das Zeichen unter dem Cursor.

**BSP** (Rücktaste)

Löscht das Zeichen links vom Cursor.

**UND\_CHAR** (Ctrl+Z, Strg+Z)

Fügt die zuletzt mit **DEL** und/oder **BSP** gelöschten Zeichen vor der aktuellen Cursor-Position ein.

**DEL\_WORD** (Plus-Minus)

Löscht den Rest des Wortes (einschließlich der unmittelbar dahinter stehenden Leerzeichen) rechts ab der aktuellen Cursor-Position.

UND\_WORD (Plus-Plus-Minus)

Fügt die zuletzt mit DEL\_WORD gelöschten Zeichen vor der aktuellen Cursor-Position ein.

DEL\_BEG (Plus-Pos1, Plus-Home)

Löscht in der aktuellen Zeile alle Zeichen links vom Cursor.

UND\_BEG (Plus-Plus-Pos1, Plus-Plus-Home)

Fügt die zuletzt mit DEL\_BEG gelöschten Zeichen vor der aktuellen Cursor-Position ein.

DEL\_END (Plus-Ende, Plus-End)

Löscht in der aktuellen Zeile das Zeichen unter dem Cursor und alle Zeichen rechts vom Cursor.

UND\_END (Plus-Plus-Ende, Plus-Plus-End)

Fügt die zuletzt mit DEL\_END gelöschten Zeichen und vor der aktuellen Cursor-Position ein.

INSERT\_CB (Ctrl+V, Strg+V)

Fügt den Inhalt der Zwischenablage vor der aktuellen Cursor-Position ein.

CLEAR (Plus-Stern)

Löscht alle Zeichen des Feldes.

TGL\_INS (Ins, Einfg, Plus-Enter)

Schaltet vom Überschreibmodus in den Einfügemodus und umgekehrt.

CR (Return, Ctrl+M, Strg+M)

Schließt die Eingabe für das Feld ab.

Wird in einem Feld vom Typ INPUT eine Stelle mit der Maus angeklickt, so wird der Cursor auf diese Stelle positioniert.

Zur Dateneingabe können mit der folgenden Makroanweisung Tastaturkürzel für Zeichenfolgen definiert werden, die danach mit ALT+x und Plus-x aufgerufen werden können:

```
$$ KEY x = "Zeichenfolge"
```

An Stelle des x muss einer der Buchstaben von A bis Z stehen. Wird eine leere Zeichenfolge angegeben, wird das entsprechende Tastaturkürzel gelöscht.

Hinweis: Die Makrofunktion KEYS (siehe Seite 591) liefert alle jeweils definierten Tastaturkürzel.

#### – INPUT/SHIFT

Wie INPUT, jedoch kann der Text im Fenster horizontal mit Steuerbefehlen verschoben werden. Bei horizontalen Cursor-Positionierungen wird der Text automatisch so verschoben, dass der Cursor immer im Feld bleibt. Die Optionen SHIFT und SCROLL können kombiniert werden.



– **INPUT/SCROLL**

Wie **INPUT**, jedoch kann der Text im Fenster vertikal mit Steuerbefehlen verschoben werden. Bei vertikalen Cursor-Positionierungen wird der Text automatisch so verschoben, dass der Cursor immer im Feld bleibt. Die Optionen **SHIFT** und **SCROLL** können kombiniert werden.

– **INPUT/EDIT**

Wie **INPUT**, jedoch wird der Text automatisch umbrochen, wenn eine Teilzeichenfolge bzw. eine Zeile des Textes aus mehr Zeichen besteht als in einer Zeile des Fensters angezeigt werden können. Bei Cursor-Positionierungen wird der Text automatisch vertikal so verschoben, dass der Cursor immer im Feld bleibt.

Bei der zu diesem Feld gehörenden **EDIT**-Anweisung kann mit der Option **LF** angegeben werden, dass der Steuerbefehl **CR** in diesem Feld die gleiche Wirkung wie der Steuerbefehl **LF** haben soll.

Folgender Steuerbefehl ist zusätzlich möglich:

**LF** (Ctrl+Return, Strg+Return, Ctrl+J, Strg+J)

Beginnt mit dem Zeichen unter dem Cursor eine neue Zeile. Das Ende der vorangehenden Zeile wird mit einer Tilde gekennzeichnet. Der Text kann auf diese Weise in der Sternvariablen `txtvar` in mehrere Zeilen aufgeteilt werden.

– **SELECT**

ist gleichbedeutend mit **SELECT/SINGLE**.

– **SELECT/SINGLE** (= listbox, Listenfeld)

Dieses Feld dient zur Auswahl einer Zeichenfolge aus vorgegebenen Zeichenfolgen. Je Zeile des Feldes wird eine Teilzeichenfolge vom Inhalt der Variablen `txtvar` angezeigt. Enthält die Variable `txtvar` mehr Teilzeichenfolgen als das Feld Zeilen umfasst, können die angezeigten Zeilen durch entsprechende Cursor-Positionierungen nach oben bzw. unten verschoben werden. Eine der angezeigten Teilzeichenfolgen kann ausgewählt werden. Wird eine weitere Teilzeichenfolge ausgewählt, so wird die Auswahl der anderen automatisch aufgehoben. Die Zeile mit der ausgewählten Teilzeichenfolge wird hervorgehoben. Die laufende Nummer der ausgewählten Teilzeichenfolge innerhalb der Variablen `txtvar` wird in der Variablen `numvar` gespeichert. Falls keine Teilzeichenfolge ausgewählt wird, wird 0 (Null) in die Variable `numvar` gespeichert.

Die Variable `txtvar` kann bei diesem Feldtyp auch eine Sternvariable sein. In diesem Fall werden die Zeilen der Sternvariablen wie die Teilzeichenfolgen einer »normalen« Variablen behandelt.

**CUR\_DN** (Pfeil nach unten)

Positioniert den Cursor um eine Zeile nach unten; falls der Cursor in der untersten Zeile des Feldes steht und noch eine weitere Teilzeichenfolge nach der in dieser Zeile angezeigten vorhanden ist, wird der Inhalt des Feldes um eine Zeile nach oben verschoben und diese Teilzeichenfolge in der letzten Zeile des Feldes angezeigt.

CUR\_UP (Pfeil nach oben)

Positioniert den Cursor um eine Zeile nach oben; falls der Cursor in der obersten Zeile des Feldes steht und noch eine weitere Teilzeichenfolge vor der in dieser Zeile angezeigten vorhanden ist, wird der Inhalt des Feldes um eine Zeile nach unten verschoben und diese Teilzeichenfolge in der ersten Zeile des Feldes angezeigt.

SHW\_DN (Bild runter, PgDn)

Blättert vorwärts weiter.

SHW\_UP (Bild rauf, PgUp)

Blättert rückwärts weiter.

SKP\_BEG (Pos1, Home)

Zeigt den Anfang.

SKP\_END (Ende, End)

Zeigt das Ende.

a bis z (Buchstaben)

Sucht die erste Zeile, die mit dem eingegebenen Buchstaben beginnt, und positioniert den Cursor auf diesen Buchstaben; falls sich diese Zeile nicht innerhalb des Feldes befindet, wird sie ins Feld geholt.

Werden unmittelbar danach weitere Buchstaben eingegeben, so werden sie mit dem ersten zu einer Zeichenfolge zusammengefügt, und der Cursor wird jeweils auf den zuletzt eingegebenen Buchstaben in der ersten Zeile positioniert, die mit dieser Zeichenfolge beginnt.

Beginnt keine Zeile mit dem eingegebenen Buchstaben bzw. mit der eingegebenen Zeichenfolge, so wird ein Signalton ausgegeben und der Cursor auf die in der alphabetischen Ordnung folgende Zeile positioniert.

Damit der Cursor jeweils richtig positioniert werden kann, müssen die Zeilen so sortiert sein, wie sie mit der Makrofunktion ALPHA\_SORT sortiert würden. Sind sie so sortiert, wie sie mit der Makrofunktion MIXED\_SORT sortiert würden (z. B. Dateilisten), muss bei der zu diesem Feld gehörenden EDIT-Anweisung die Option MIXED\_SORT angegeben werden, damit der Cursor korrekt positioniert werden kann.

CR (Return, Ctrl+M, Strg+M)

Wählt die Teilzeichenfolge aus, die in der Zeile des Feldes angezeigt wird, in der der Cursor steht, und schließt die Eingabe für das Feld ab. Die Zeile mit der ausgewählten Teilzeichenfolge wird hervorgehoben angezeigt. Wurde bereits eine andere Teilzeichenfolge in diesem Feld ausgewählt, so wird ihre Auswahl aufgehoben und die Zeile mit dieser Teilzeichenfolge wieder normal angezeigt.

DEL (Entf, Del, Shift+Backspace, Shift+Rücktaste)

Hebt die Auswahl der Teilzeichenfolge auf, die in der Zeile des Feldes angezeigt wird, in der der Cursor steht, und schließt die Eingabe für das Feld ab. Die Zeile mit der Teilzeichenfolge wird wieder normal angezeigt.

CLEAR (Plus-Stern)

Hebt die Teilzeichenfolgen-Auswahl auf.

Wird in einem Feld vom Typ `SELECT` eine Zeile mit der Maus angeklickt, so wird bei Ausführung der `EDIT`-Anweisung in diesem Feld der Cursor in diese Zeile positioniert und automatisch ein Steuerbefehl erzeugt: `CR`, wenn die Teilzeichenfolge dieser Zeile noch nicht ausgewählt ist, und `DEL`, wenn die Teilzeichenfolge dieser Zeile schon ausgewählt ist. In beiden Fällen wird die Eingabe jedoch nicht abgeschlossen; der Cursor bleibt in der angeklickten Zeile stehen.

– **SELECT/SINGLE/CLICK**

Wie `SELECT/SINGLE`, jedoch wird die Eingabe abgeschlossen, wenn eine Zeile mit der Maus angeklickt wird.

– **SELECT/MULTIPLE**

Wie `SELECT/SINGLE`, jedoch können mehrere Teilzeichenfolgen ausgewählt werden. In diesem Fall werden die einzelnen laufenden Nummern der ausgewählten Teilzeichenfolgen durch Apostroph getrennt in der Variablen `numvar` gespeichert. Falls keine Teilzeichenfolge ausgewählt wird, enthält diese Variable eine leere Zeichenfolge.

CR (Return, Ctrl+M, Strg+M)

Wählt die Teilzeichenfolge aus, die in der Zeile des Feldes angezeigt wird, in der der Cursor steht, und zeigt die Zeile hervorgehoben an. Der Cursor wird um eine Zeile nach unten positioniert. Falls der Cursor jedoch schon in der letzten Zeile des Feldes steht, wird die Eingabe für das Feld abgeschlossen.

DEL (Entf, Del, Shift+Backspace, Shift+Rücktaste)

Hebt die Auswahl der Teilzeichenfolge auf, die in der Zeile des Feldes angezeigt wird, in der der Cursor steht, und zeigt die Zeile wieder normal an. Der Cursor wird um eine Zeile nach unten positioniert. Falls der Cursor jedoch schon in der letzten Zeile des Feldes steht, wird die Eingabe für das Feld abgeschlossen.

– **SELECT/MULTIPLE/CLICK**

Wie `SELECT/MULTIPLE`, jedoch wird die Eingabe abgeschlossen, wenn eine Zeile mit der Maus angeklickt wird.

– **SELECT/SINGLE/MULTIPLE**

Wie `SELECT/MULTIPLE`, jedoch kann die erste Teilzeichenfolge nicht zusammen mit einer anderen ausgewählt werden. Wird die erste ausgewählt, wird die Auswahl aller anderen automatisch aufgehoben; wird eine andere als die erste ausgewählt, wird die Auswahl der ersten automatisch aufgehoben.

– **SELECT/SINGLE/MULTIPLE/CLICK**

Wie **SELECT/SINGLE/MULTIPLE**, jedoch wird die Eingabe abgeschlossen, wenn eine Zeile mit der Maus angeklickt wird.

– **FLAGS**

ist gleichbedeutend mit **FLAGS/MULTIPLE**.

– **FLAGS/SINGLE** (= radio button, Optionsfeld )

Dieses Feld dient zur Auswahl von maximal einer der vorgegebenen Möglichkeiten. Die Bedeutung der einzelnen Möglichkeiten sollte in der jeweiligen Zeile links oder rechts von diesem Feld durch beschreibenden Text oder in einem Feld vom Typ **OUTPUT/CLICK** angegeben werden. In Feldern vom Typ **FLAGS/SINGLE** wird in jeder Zeile ein Schalter angezeigt, der auf »ein« (ja, wahr, zutreffend) oder »aus« (nein, falsch, nicht zutreffend) gestellt werden kann. In einem solchen Feld kann jedoch immer nur ein einziger Schalter auf »ein« gestellt sein. Wird ein weiterer Schalter auf »ein« gestellt, so wird der andere automatisch auf »aus« gestellt. Die Variable `txtvar` entfällt bei diesem Feldtyp. Die Schalter werden (intern) durchnummeriert, und in der Variablen `numvar` wird jeweils die laufende Nummer des Schalters gespeichert, der auf »ein« gestellt ist. Falls kein Schalter auf »ein« gestellt ist, enthält diese Variable den Wert 0 (Null).

**CUR\_UP / CUR\_DN** (Pfeil nach oben / Pfeil nach unten)

Positioniert den Cursor innerhalb des Feldes um eine Zeile nach oben/unten.

**CR** (Return, Ctrl+M, Strg+M)

Setzt den Schalter, auf dem der Cursor steht, auf »ein« und schließt die Eingabe für das Feld ab.

**DEL** (Entf, Del, Shift+Backspace, Shift+Rücktaste)

Setzt den Schalter, auf dem der Cursor steht, auf »aus« und schließt die Eingabe für das Feld ab.

**CLEAR** (Plus-Stern)

Setzt den auf »ein« gesetzten Schalter auf »aus«.

Wird ein Schalter mit der Maus angeklickt, so wird bei Ausführung der **EDIT**-Anweisung in diesem Feld der Cursor auf diesen Schalter positioniert und automatisch ein Steuerbefehl erzeugt: **CR**, wenn der Schalter auf »aus« steht, und **DEL**, wenn der Schalter auf »ein« steht. In beiden Fällen wird die Eingabe jedoch nicht abgeschlossen; der Cursor bleibt in der angeklickten Zeile stehen.

– **FLAGS/SINGLE/CLICK**

Wie **FLAGS/SINGLE**, jedoch wird die Eingabe abgeschlossen, wenn ein Schalter mit der Maus angeklickt wird.

– **FLAGS/MULTIPLE** (= checkbox)

Dieses Feld dient zur Auswahl von beliebig vielen der vorgegebenen Möglichkeiten. Die Bedeutung der einzelnen Möglichkeiten sollte in der jeweiligen Zeile links oder rechts von diesem Feld durch beschreibenden Text oder in einem Feld vom Typ `OUTPUT/CLICK` angegeben werden. In Feldern vom Typ `FLAGS/MULTIPLE` wird in jeder Zeile ein Schalter angezeigt, der auf »ein« (ja, wahr, zutreffend) oder »aus« (nein, falsch, nicht zutreffend) gestellt werden kann. In einem solchen Feld dürfen (im Gegensatz zu einem Feld vom Typ `FLAGS/SINGLE`) beliebig viele Schalter auf »ein« gestellt sein. Die Variable `txtvar` entfällt bei diesem Feldtyp. Die Schalter werden (intern) durchnummeriert, und in der Variablen `numvar` werden jeweils die laufenden Nummern der Schalter gespeichert, die auf »ein« gestellt sind. Sind dies mehrere Schalter, so werden die einzelnen Nummern durch Apostroph getrennt. Falls kein Schalter auf »ein« gestellt ist, enthält diese Variable eine leere Zeichenfolge.

`CUR_UP / CUR_DN` (Pfeil nach oben / Pfeil nach unten)

Positioniert den Cursor innerhalb des Feldes um eine Zeile nach oben/unten.

`CR` (Return, Ctrl+M, Strg+M)

Setzt den Schalter, auf dem der Cursor steht, auf »ein« und positioniert den Cursor um eine Zeile nach unten. Falls der Cursor jedoch schon in der letzten Zeile des Feldes stand, wird die Eingabe für das Feld damit abgeschlossen.

`DEL` (Entf, Del, Shift+Backspace, Shift+Rücktaste)

Setzt den Schalter, auf dem der Cursor steht, auf »aus« und positioniert den Cursor um eine Zeile nach unten. Falls der Cursor jedoch schon in der letzten Zeile des Feldes stand, wird die Eingabe für das Feld damit abgeschlossen.

`CLEAR` (Plus-Stern)

Setzt alle auf »ein« gesetzten Schalter auf »aus«.

Wird ein Schalter mit der Maus angeklickt, so wird bei Ausführung der `EDIT`-Anweisung in diesem Feld automatisch der Cursor auf diesen Schalter positioniert und ein Steuerbefehl erzeugt: `CR`, wenn der Schalter auf »aus« steht, und `DEL`, wenn der Schalter auf »ein« steht.

– **FLAGS/MULTIPLE/CLICK**

Wie `FLAGS/MULTIPLE`, jedoch wird die Eingabe abgeschlossen, wenn ein Schalter mit der Maus angeklickt wird.

– **FLAGS/SINGLE/MULTIPLE**

Wie `FLAGS/MULTIPLE`, jedoch kann der erste Schalter des Feldes nicht zusammen mit einem anderen auf »ein« gestellt sein. Wird der erste auf »ein« gestellt, werden alle anderen automatisch auf »aus« gestellt; wird ein anderer als der erste auf »ein« gestellt, wird der erste automatisch auf »aus« gestellt.

– **FLAGS/SINGLE/MULTIPLE/CLICK**

Wie **FLAGS/SINGLE/MULTIPLE**, jedoch wird die Eingabe abgeschlossen, wenn ein Schalter mit der Maus angeklickt wird.

– **BUTTON** (= Schaltfläche)

Dieses Feld dient zum Auslösen einer Aktion. Unter welchen Umständen eine Aktion ausgelöst wird, hängt von den Anweisungen ab, die auf die **EDIT**-Anweisung folgen. Üblicherweise wird eine Aktion ausgelöst, wenn die Eingabe für ein solches Feld mit **CR** abgeschlossen wird. Dies muss abgefragt werden und die gewünschte Aktion ggf. mit entsprechenden Makroanweisungen ausgeführt werden.

**CUR\_UP** / **CUR\_DN** (Pfeil nach oben / Pfeil nach unten)

Falls das unmittelbar davor/danach definierte Feld ebenfalls vom Typ **BUTTON** ist und in der vorangehenden/nachfolgenden Zeile auf der gleichen Position angezeigt wird, wirkt dieser Steuerbefehl wie der Steuerbefehl **BACKTAB/TAB**.

**CUR\_LE** / **CUR\_RI** (Pfeil nach links / Pfeil nach rechts)

Falls das unmittelbar davor/danach definierte Feld ebenfalls vom Typ **BUTTON** ist und in der gleichen Zeile unmittelbar davor/danach angezeigt wird, wirkt dieser Steuerbefehl wie der Steuerbefehl **BACKTAB/TAB**.

**CR** (Return, Ctrl+M, Strg+M)

Schließt die Eingabe für das Feld ab.

Wird ein Feld vom Typ **BUTTON** mit der Maus angeklickt, so wird bei Ausführung der **EDIT**-Anweisung in diesem Feld automatisch der Steuerbefehl **CR** erzeugt.

– **BUTTON/CENTER**

wie **BUTTON**, jedoch wird der Inhalt der zum Feld gehörenden Variablen **txtvar** nicht linksbündig im Feld, sondern in der Mitte des Feldes angezeigt. Die Option **CENTER** kann auch zusammen mit einer der nachfolgend beschriebenen Optionen angegeben werden.

– **BUTTON/ENTER**

Dieses Feld darf in einem Makrofenster nur einmal vorkommen und hat außer den Eigenschaften eines Feldes vom Typ **BUTTON** noch folgende zwei Besonderheiten:

Wenn in einem anderen Feld die Eingabe mit dem Steuerbefehl **ENTER** abgeschlossen wurde und alle Anweisungen des anderen Feldes abgearbeitet sind, wird dieses Feld aktiviert. Bei der Ausführung der **EDIT**-Anweisung wird in diesem Fall die Eingabe sofort automatisch abgeschlossen.

Wenn in diesem Feld die Eingabe mit der Return-Taste abgeschlossen wird, wird **CR** durch **ENTER** ersetzt. Dies ist bei evtl. nachfolgenden Abfragen des letzten Steuerbefehls von Bedeutung.

**– BUTTON/CANCEL**

Dieses Feld darf in einem Makrofenster nur einmal vorkommen und hat außer den Eigenschaften eines Feldes vom Typ `BUTTON` noch folgende zwei Besonderheiten:

Wenn in einem anderen Feld die Eingabe mit dem Steuerbefehl `CANCEL` abgeschlossen wurde und alle Anweisungen des anderen Feldes abgearbeitet sind, wird dieses Feld aktiviert. Bei der Ausführung der `EDIT`-Anweisung wird in diesem Fall die Eingabe sofort automatisch abgeschlossen.

Wenn in diesem Feld die Eingabe mit der Return-Taste abgeschlossen wird, wird `CR` durch `CANCEL` ersetzt. Dies ist bei evtl. nachfolgenden Abfragen des letzten Steuerbefehls von Bedeutung.

**– BUTTON/HELP**

Dieses Feld darf in einem Makrofenster nur einmal vorkommen und hat außer den Eigenschaften eines Feldes vom Typ `BUTTON` noch folgende zwei Besonderheiten:

Wenn in einem anderen Feld die Eingabe mit dem Steuerbefehl `HELP` abgeschlossen wurde und alle Anweisungen des anderen Feldes abgearbeitet sind, wird dieses Feld aktiviert. Bei der Ausführung der `EDIT`-Anweisung wird in diesem Fall die Eingabe sofort automatisch abgeschlossen.

Wenn in diesem Feld die Eingabe mit der Return-Taste abgeschlossen wird, wird `CR` durch `HELP` ersetzt. Dies ist bei evtl. nachfolgenden Abfragen des letzten Steuerbefehls von Bedeutung.

**– BUTTON/FKEY**

Dieses Feld darf in einem Makrofenster nur einmal vorkommen und hat außer den Eigenschaften eines Feldes vom Typ `BUTTON` noch folgende zwei Besonderheiten:

Wenn in einem anderen Feld die Eingabe mit einer Funktionstaste abgeschlossen wurde und alle Anweisungen des anderen Feldes abgearbeitet sind, wird dieses Feld aktiviert. Bei der Ausführung der `EDIT`-Anweisung wird in diesem Fall die Eingabe sofort automatisch abgeschlossen.

Wenn in diesem Feld die Eingabe mit der Return-Taste abgeschlossen wird, wird `CR` durch `FKEY` ersetzt. Dies ist bei evtl. nachfolgenden Abfragen des letzten Steuerbefehls von Bedeutung.

**– OUTPUT**

Dieses Feld dient zum Anzeigen eines Textes, der sich farblich vom Hintergrund des Makrofensters abheben soll, oder der während der Anzeige eines Makrofensters (durch Ändern des Inhalts der zum Feld gehörenden Variablen `txtvar`) variiert werden soll. Je Zeile des Feldes wird eine Teilzeichenfolge vom Inhalt der zum Feld gehörenden Variablen `txtvar` angezeigt. Die Variable `numvar` entfällt bei diesem Feldtyp. Der Cursor kann nicht in dieses Feld positioniert werden.

Die Variable `txtvar` kann bei diesem Feldtyp auch eine Sternvariable sein. In

diesem Fall werden die Zeilen der Sternvariablen wie die Teilzeichenfolgen einer »normalen« Variablen behandelt.

– **OUTPUT/CENTER**

wie **OUTPUT**, jedoch wird der Inhalt der zum Feld gehörenden Variablen `txtvar` nicht linksbündig im Feld, sondern in der Mitte des Feldes angezeigt.

– **OUTPUT/SHIFT**

Wie Feldtyp **OUTPUT**, jedoch kann der Cursor in dieses Feld positioniert werden, um dann den Inhalt des Makrofensters nach links/rechts zu verschieben, falls die Teilzeichenfolgen bzw. Zeilen des Textes aus mehr Zeichen bestehen, als in einer Zeile des Feldes angezeigt werden können. Die Variable `numvar` enthält die Cursor-Position innerhalb des in der Variablen `txtvar` stehenden Textes. Die Optionen **SHIFT** und **SCROLL** können kombiniert werden.

`CUR_LE` (Pfeil nach links)

Positioniert den Cursor um ein Zeichen nach links; falls der Cursor ganz links im Feld steht und noch weiterer Text links vor dem angezeigten vorhanden ist, wird der Inhalt des Feldes um eine Position nach rechts verschoben.

`CUR_RI` (Pfeil nach rechts)

Positioniert den Cursor um ein Zeichen nach rechts; falls der Cursor ganz rechts im Feld steht und noch weiterer Text rechts nach dem angezeigten vorhanden ist, wird der Inhalt des Feldes um eine Position nach links verschoben.

– **OUTPUT/SCROLL**

Wie Feldtyp **OUTPUT**, jedoch kann der Cursor in dieses Feld positioniert werden, um dann den Inhalt des Makrofensters nach oben/unten zu verschieben, falls die Variable mehr Teilzeichenfolgen enthält, als das Feld Zeilen umfasst. Die Variable `numvar` enthält die Cursor-Position innerhalb des in der Variablen `txtvar` stehenden Textes. Die Optionen **SHIFT** und **SCROLL** können kombiniert werden.

`CUR_DN` (Pfeil nach unten)

Positioniert den Cursor um eine Zeile nach unten; falls der Cursor in der untersten Zeile des Feldes steht und noch eine weitere Teilzeichenfolge nach der in dieser Zeile angezeigten vorhanden ist, wird der Inhalt des Feldes um eine Zeile nach oben verschoben und diese Teilzeichenfolge in der letzten Zeile des Feldes angezeigt.

`CUR_UP` (Pfeil nach oben)

Positioniert den Cursor um eine Zeile nach oben; falls der Cursor in der obersten Zeile des Feldes steht und noch eine weitere Teilzeichenfolge vor der in dieser Zeile angezeigten vorhanden ist, wird der Inhalt des Feldes um eine Zeile nach unten verschoben und diese Teilzeichenfolge in der ersten Zeile des Feldes angezeigt.



---

– **OUTPUT/WRAP**

Wie **OUTPUT**, jedoch wird der Text automatisch umbrochen, wenn eine Teilzeilenfolge bzw. eine Zeile des Textes aus mehr Zeichen besteht als in einer Zeile des Fensters angezeigt werden können. Bei Cursor-Positionierungen wird der Text automatisch vertikal so verschoben, dass der Cursor immer im Feld bleibt. Die Variable `numvar` enthält die Cursor-Position innerhalb des in der Variablen `txtvar` stehenden Textes.

– **OUTPUT/CLICK**

Wie Feldtyp **OUTPUT**, jedoch muss das unmittelbar davor definierte Feld vom Typ **FLAGS** oder **SELECT** sein und in den gleichen Zeilen des Makrofensters angezeigt werden wie dieses Feld. Wird mit der Maus eine Zeile dieses Feldes (vom Typ **OUTPUT/CLICK**) angeklickt, so hat dies die gleiche Wirkung, wie wenn der in der gleichen Zeile stehende Schalter bzw. Text des davor definierten Feldes (vom Typ **FLAGS** bzw. **SELECT**) angeklickt wird.

– **DUMMY**

Dieses Feld wird nicht angezeigt. Es kann die gleichen Anweisungen enthalten wie andere Felder auch. Mit der Anweisung **GO TO FIELD** (siehe Seite 645) kann es aktiviert werden. Während es aktiv ist, bleibt der Cursor unsichtbar.

## Testhilfen

Diese Testhilfen eignen sich nur für Makros, die eine Kommandofolge zusammenstellen. Zum Testen von Makros, die ihre Leistung allein durch Ausführen von Makroanweisungen erbringen, können die unter »Fehlersuche« (siehe Seite 459) aufgeführten Makroanweisungen verwendet werden.

Beim Aufruf eines Makros kann unmittelbar hinter dem Makronamen angegeben werden,

- ob alle vom Makro erzeugten Kommandos ausgeführt werden sollen oder nur ein bestimmter Teil.
- ob die erzeugten Kommandos (evtl. zusätzlich) protokolliert werden sollen.
- ob die erzeugten Kommandos ausgeführt werden sollen.

#`$Makroname-BEREICH;PROTOKOLL;AUSFUEHRUNG, . . .`

BEREICH	= pos	Nur die Kommandofolge ausführen, die ab der Satzposition <code>pos</code> im Makro erzeugt wird.
	= pos1-pos2	Nur die Kommandofolge ausführen, die von der Satzposition <code>pos1</code> bis zur Satzposition <code>pos2</code> im Makro (je einschließlich) erzeugt wird.
		»pos« steht jeweils für eine Satznummer, deren Schreibweise dem Programmmodus (vgl. Seite 33) entspricht.
PROTOKOLL	= -	* Keine zusätzliche Protokollierung der erzeugten Kommandofolge.
	= +	Erzeugte Kommandofolge ins Ablaufprotokoll ausgeben.
	= datei	Name der Datei für das Protokoll der erzeugten Kommandofolge.
AUSFUEHRUNG	= +	* Erzeugte Kommandofolge ausführen.
	= -	Erzeugte Kommandofolge nicht ausführen (sinnvoll, wenn die Kommandofolge zu Testzwecken nur protokolliert werden soll).
	= datei	Erzeugte Kommandofolge nur in die angegebene Datei ausgeben (nicht ausführen).

## Beispiele

Beispiele für Makros, die ihre Leistung allein durch Ausführen von Makroanweisungen erbringen, sind außer einigen einfachen CGI-Makros aus Platzgründen hier nicht aufgeführt. Mit dem Aufruf des Makros \*BEISPIEL werden solche Beispiele in die im Aufruf angegebene Datei geschrieben. Weitere Information darüber wird mit dem Kommando #INFORMIERE, \*BEISPIEL ausgegeben.

Das Makro \*KOMA schreibt ein Makro, das als Vorlage zum satz- oder portionsweisen Kopieren von Dateien verwendet werden kann und nur noch entsprechend angepasst und ergänzt werden muss, in eine Datei. Weitere Informationen über das Makro werden mit dem Kommando #INFORMIERE, \*KOMA ausgegeben.

### Beispiele für CGI-Makros

Diese Beispiele setzen Grundkenntnisse in HTML voraus.

Es soll die aktuelle Uhrzeit und eine Schaltfläche zum Aktualisieren der Uhrzeit angezeigt werden.

```

$$ MODE TUSCRIPT, {}

FETCH server = SERVER_NAME
FETCH port   = SERVER_PORT
FETCH script = SCRIPT_NAME

SET url = "http://{server}:{port}{script}"

SET time = TIME (3)

MODE DATA
<HTML>

Aktuelle Uhrzeit: {time}

<FORM ACTION="{url}" METHOD="GET">
<INPUT TYPE="SUBMIT" VALUE=" Aktualisieren ">
</FORM>

</HTML>

```

Den CGI-Konventionen entsprechend sind Servername, Portnummer und Scriptname in den System-Variablen SERVER\_NAME, SERVER\_PORT und SCRIPT\_NAME gespeichert. Der Inhalt dieser System-Variablen wird mit den FETCH-Anweisungen in die Makrovariablen server, port und script geholt. Dann wird in der Variablen url die Adresse, an die der Browser wieder antworten soll, und in der Variablen time die aktuelle Uhrzeit gespeichert. Zum Schluss wird eine »HTML-Datei« zusammengestellt, die an den Browser geschickt wird.

Wenn zusätzlich angezeigt werden soll, wann das CGI-Makro zuvor aufgerufen wurde, muss dafür die Uhrzeit des jeweils letzten Aufrufs gemerkt werden. Da die Makrovariablen bei jedem Aufruf neu definiert und nach dem Ausführen des Makros wieder gelöscht werden, können in Makrovariablen keine Werte von einem Aufruf bis zum nächsten gemerkt werden. Werte können aber so in die HTML-Datei eingefügt werden, dass sie vom Browser nicht angezeigt, sondern nur wieder in die Antwort eingefügt werden.

```

$$ MODE TUSCRIPT, {}

FETCH server = SERVER_NAME
FETCH port   = SERVER_PORT
FETCH script = SCRIPT_NAME

SET url = "http://{server}:{port}{script}"

FETCH query = QUERY_STRING

SET list = DECODE (query, CGI)
SET old_time = GET_VALUE (list, "time")

SET new_time = TIME (3)

MODE DATA
<HTML>

Aktuelle Uhrzeit: {new_time} - Letzter Aufruf: {old_time}

<FORM ACTION="{url}" METHOD="GET">
<INPUT TYPE="SUBMIT" VALUE=" Aktualisieren " >
<INPUT TYPE="HIDDEN" NAME="time" VALUE="{new_time}" >
</FORM>

</HTML>

```

Den CGI-Konventionen entsprechend wird die Antwort des Servers in der System-Variablen `QUERY_STRING` gespeichert. Der Inhalt dieser System-Variablen wird mit der `FETCH`-Anweisung in die Makrovariable `query` geholt. Da die Antwort entsprechend den CGI-Konventionen codiert ist, wird sie zuerst decodiert, daraus eine Werteliste erstellt und diese in die Sternvariable `list` gespeichert. Aus dieser Werteliste wird der unter dem Namen `time` enthaltene Wert geholt und in die Makrovariable `old_time` gespeichert. Der Name `time` entspricht dem Namen des unsichtbaren Feldes (`TYPE="HIDDEN"`) in der HTML-Datei, in dem jeweils die aktuelle Uhrzeit des Aufrufs »gemerkt« wird.

Müssen mehrere Werte von Aufruf zu Aufruf »gemerkt« werden, kann für jeden Wert ein unsichtbares Feld mit einem frei wählbaren Namen angegeben werden.

Wenn in der HTML-Datei `METHOD="GET"` angegeben ist, wird die Antwort des Browser jeweils auch in der Adress-Zeile des Browsers angezeigt. Wenn dies unter-

bleiben soll, oder wenn die Antworten lang sind, muss `METHOD="POST"` angegeben werden. In diesem Fall wird die Antwort nicht in der System-Variablen `QUERY_STRING` gespeichert, sondern in der Standard-Eingabe bereitgestellt. Die Antwort kann in diesem Fall mit folgender Makroanweisung in die Makrovariable `query` geholt werden:

```
FETCH query = -STD-
```

Wenn unter Windows der in TUSTEP enthaltene WWW-Server verwendet wird, kann ein Makro in den Server integriert werden (siehe Seite 101). In diesem Fall wird das Makro nur einmal gestartet und die Makrovariablen und deren Werte bleiben von Aufruf zu Aufruf erhalten. Bei einem integrierten Makro muss `METHOD="POST"` verwendet werden.

```
$$ MODE TUSCRIPT, {}

FETCH server = SERVER_NAME
FETCH port   = SERVER_PORT
FETCH script = SCRIPT_NAME

SET url = "http://{server}:{port}{script}"

SET old_time = ""

LOOP/999999

    FETCH query = -STD-

    SET new_time = TIME (3)

    MODE DATA
    <HTML>

    Aktuelle Uhrzeit: {new_time} - Letzter Aufruf: {old_time}

    <FORM ACTION="{url}" METHOD="POST">
    <INPUT TYPE="SUBMIT" VALUE=" Aktualisieren ">
    </FORM>

    </HTML>
    $$ MODE STATEMENT

    SET old_time = new_time

ENDLOOP
```

Dieses Makro entspricht dem zuvor aufgeführten. Wenn das Makro jedoch von mehr als einem Browser-Fenster (egal, ob von einem Rechner oder von verschiedenen Rechnern) aus aufgerufen wird, ergibt sich ein Unterschied: Im oberen Makro wird die Uhrzeit vom jeweiligen Browser »gemerkt«. Als Uhrzeit des letzten Aufrufs wird

deshalb jeweils die Zeit angezeigt, zu der der jeweilige Browser das Makro zuletzt aufgerufen hat. Im unter Makro wird die Uhrzeit vom Makro selbst (in der Variablen `old_time`) gemerkt. Als Uhrzeit des letzten Aufrufs wird deshalb jeweils die Zeit angezeigt, zu der das Makro (egal von welchem Browser-Fenster aus) zuletzt aufgerufen wurde.

## Beispiel für ein Makro, das eine Kommandofolge zusammenstellt

```

$$- Leistung: Mit dem Makro PRINT kann eine Datei
$$-           auf einem Drucker ausgedruckt werden.
$$-
$$- Aufruf:   # $PRINT, DATEI=..., MODUS=..., DRUCKER=...
$$-
$$-           Modus=F   formatieren und drucken
$$-           Modus=T   im Textmodus drucken
$$-           Modus=P   im Programmmodus drucken
$$-
$$! DATEI, MODUS, DRUCKER=HPLJ
$$-
$$- Falls kein Drucker angegeben, wird HPLJ angenommen
$$-
$$- Falls kein Dateiname angegeben, nachfordern
$$- (bei leerer Eingabe werden keine Kommandos erzeugt)
$$-
$$ IF ("".EQ."") THEN
$$   ASK "Gib Dateiname ": DATEI = "-"
$$   IF ("".EQ."-") STOP
$$ ENDIF
$$-
$$- Falls kein Modus angegeben, nachfordern
$$- (bei leerer Eingabe wird Modus F angenommen)
$$-
$$ IF ("".EQ."") THEN
$$   + Zu Modus sind folgende Angaben vorgesehen:
$$   +           F für Formatieren und Drucken
$$   +           T für Drucken im Textmodus
$$   +           P für Drucken im Programmmodus
$$   +           leere Eingabe ist gleichwertig mit F
$$   ASK "Gib Modus ": MODUS = "F"
$$ ENDIF
$$-
$$- Zusammenstellen der Kommandos
$$-
$$ IF ("".EQ."F") THEN
#formatiere,quelle=<DATEI>,loeschen=+,parameter=*
drt      <DRUCKER>
*EOF
$$ ELSE
#dvorbereite,quelle=<DATEI>,modus=<MODUS>,loeschen=+,parameter=*
drt      <DRUCKER>
*EOF
$$ ENDIF
#drucke,,<DRUCKER>

```

## Beispiel für die Verwendung des Kommandos #MAKRO

In diesem Beispiel wird das Makro PRINT aus dem vorangehenden Beispiel in seiner Leistung erweitert.

Beim Formatieren von Texten können Fehler auftreten, einerseits durch falsches Codieren von Formatieranweisungen und Sonderzeichen und andererseits dadurch, dass nicht alle definierten Sonderzeichen auf allen Druckertypen darstellbar sind. Falls FORMATIERE zu viele Fehler meldet, sollen eventuell erst nochmal die Eingabedaten korrigiert werden. Deshalb kann es wünschenswert sein, dass im obigen Beispiel eine Entscheidung möglich ist, ob das Ergebnis von FORMATIERE ausgedruckt werden soll oder nicht. Dazu liegt folgender Lösungsversuch nahe:

```
$$ ASK "Protokoll drucken? (ja/nein) ": ANTWORT = ""
$$ IF ("".AB."ja") THEN
#drucke,, <DRUCKER>
$$ ENDIF
```

Diese Anweisungsfolge hat aber nicht die gewünschte Wirkung. Sie wird nämlich nicht erst bei der Ausführung der durch das Makro erzeugten Kommandos ausgeführt, sondern bereits beim Aufruf des Makros PRINT. Zu diesem Zeitpunkt ist aber FORMATIERE noch nicht ausgeführt und es ist deshalb noch nicht bekannt, ob ausgedruckt werden soll. Die Makroanweisungen für die Entscheidung dürfen erst nach dem Formatieren ausgeführt werden. Dies kann mit Hilfe des Kommandos #MAKRO erreicht werden:

```
#makro
$$ ASK "Protokoll drucken? (ja/nein) ": ANTWORT = ""
$$ IF ("".AB."ja") THEN
#drucke,, <DRUCKER>
$$ ENDIF
*EOF
```

In dieser Form würden die Makroanweisungen jedoch auch schon beim Aufruf des Makros ausgeführt. Um dies zu verhindern, muss das Kennzeichen (hier das Dollarzeichen) für Makroanweisungen noch geändert werden, damit die Makroanweisungen für das Kommando #MAKRO nicht schon beim Aufruf des Makros PRINT als solche erkannt werden, sondern erst wenn das Kommando #MAKRO ausgeführt wird:

```
$$= *
#makro
$$ ASK "Protokoll drucken? (ja/nein) ": ANTWORT = ""
$$ IF ("".AB."ja") THEN
#drucke,, <DRUCKER>
$$ ENDIF
*EOF
**= $
```

Bleibt noch ein Problem: Für die in spitzen Klammern stehende Makrovariable ANTWORT darf erst während der Ausführung des Kommandos #MAKRO der aktuelle Wert eingesetzt werden, denn erst dann ist sie definiert. Deshalb müssen außer dem Kennzeichen für Makroanweisungen auch noch die Klammern geändert werden,



mit denen zu ersetzende Makrovariablen eingeschlossen werden. Die Makrovariable DRUCKER muss jedoch schon während des Abarbeitens des Makros PRINT eingesetzt werden und muss deshalb mit den zu dieser Zeit gültigen Klammern eingeschlossen werden. Die folgenden Anweisungen erbringen nun die gewünschte Leistung:

```

$$= * []
#makro
$$ ASK "Protokoll drucken? (ja/nein) ": ANTWORT = ""
$$ IF ("".AB."ja") THEN
#drucke,, [DRUCKER]
$$ ENDIF
*EOF
**= $ <>

```

Da im Makro PRINT diese Anfrage nur nach dem FORMATIERE und nicht nach dem DVORBEREITE erfolgen soll, kann das Kommando #DRUCKE in der letzten Zeile durch folgende Zeilen ersetzt werden:

```

$$ IF ("".EQ."F") THEN
$$= * []
#makro
$$ ASK "Protokoll drucken? (ja/nein) ": ANTWORT = ""
$$ IF ("".AB."ja") THEN
#drucke,, [DRUCKER]
$$ ENDIF
*EOF
**= $ <>
$$ ELSE
#drucke,, <DRUCKER>
$$ ENDIF

```

Wird das so geänderte Makro PRINT mit dem Kommando

```
#$PRINT, quelledat, f
```

aufgerufen, wird dadurch folgende Kommandofolge erzeugt:

```

#formatiere, quelle=quelledat, loeschen=+, parameter=*
drt          HPLJ
*EOF
#makro
$$ ASK "Protokoll drucken? (ja/nein) ": ANTWORT = ""
$$ IF ("".AB."ja") THEN
#drucke,, HPLJ
$$ ENDIF
*EOF

```



**Suche**

---

Allgemeines . . . . .	669
Tastenbezeichnungen . . . . .	669
Gemeinsame Steuerbefehle für alle Menü-Fenster . . . . .	670
Gemeinsame Steuerbefehle für alle Daten-Fenster . . . . .	671
Steuerbefehle innerhalb der Eingabefelder . . . . .	671
Gehe-zu-Fenster . . . . .	673
Steuerbefehle . . . . .	673
Register-Fenster . . . . .	674
Steuerbefehle . . . . .	674
Auswahl-Fenster . . . . .	675
Manuelles Setzen/Löschen von Markierungen . . . . .	675
Automatisches Setzen/Löschen von Markierungen . . . . .	676
Steuerbefehle . . . . .	677
Eingabefelder . . . . .	677
Index-Fenster . . . . .	679
Steuerbefehle . . . . .	680
Beispiel . . . . .	681
KWIC-Fenster . . . . .	683
Steuerbefehle und Eingaben . . . . .	683
Beispiel . . . . .	684
Umgebungs-Fenster . . . . .	685
Steuerbefehle und Eingaben . . . . .	686
Volltext-Fenster . . . . .	686
Steuerbefehle und Eingaben . . . . .	686
Beispiel . . . . .	688
Editorfenster . . . . .	689
Fenstergröße ändern, Fenster verschieben . . . . .	690
Ändern/Verschieben mit der Maus . . . . .	690
Ändern/Verschieben mit der Tastatur . . . . .	690
Beispiel . . . . .	691
Wechseln in ein anderes Fenster . . . . .	692
Wechseln mit der Maus . . . . .	692
Wechseln mit der Tastatur . . . . .	692
Farbeinstellung ändern . . . . .	693
Ändern mit der Maus . . . . .	693
Ändern mit der Tastatur . . . . .	694
Aufbereiten der Daten . . . . .	695
Beispieltext . . . . .	696

## Allgemeines

Um mit dem Kommando #SUCHE (siehe Seite 228) bestimmte Textstellen aufsuchen zu können, muss der Text mit dem Makro \*SUCHE aufbereitet werden. Dabei wird ein Register mit allen vorkommenden Wortformen erstellt.

Falls der Text entsprechende Kennzeichen enthält, können neben dem Wortformenregister auch noch andere Register (z. B. Personen- und Ortsregister) erstellt werden. Die Einträge, die in eines dieser zusätzlichen Register kommen, werden nicht in das Wortformenregister aufgenommen.

Um bestimmte Registereinträge im Text aufzufinden, müssen diese in einem Index markiert werden. Der Index enthält die Einträge aller Register, falls keine Auswahl der Register getroffen wird.

Nachdem Indexeinträge markiert worden sind, können die jeweiligen Textstellen angezeigt werden. Für die Anzeige sind drei Formate möglich: KWIC-Format, Umgebung (z. B. die Zeile mit dem betreffenden Indexeintrag und davor und danach jeweils 2 weitere Zeilen) und Volltext.

Nach dem Aufruf des Kommandos erscheint das Gehe-zu-Fenster, mit dem jeweils der nächste Arbeitsschritt (das nächste Fenster) ausgewählt bzw. das Programm beendet werden kann.

## Tastenbezeichnungen

Shift	»Umschalttaste« für Großbuchstaben.
Ctrl	»Control-Taste« links im Buchstabenblock unter der Shift-Taste; Leertaste; auf deutschen Tastaturen meist mit »Strg« beschriftet.
Return	»Eingabetaste« rechts im Buchstabenblock über der Shift-Taste.
Enter	»Eingabetaste« rechts unten im Ziffernblock. Alternativ kann in TUSTEP auch Shift+Return oder Ctrl+E verwendet werden.
Backspace	»Rücktaste« rechts im Buchstabenblock über der Return-Taste.
Delete	»Löschtaste«, sie ist mit »Entf«, »Del« oder »Delete« beschriftet. Alternativ kann in TUSTEP auch Shift+Backspace verwendet werden.
Escape	Taste ganz oben links, sie ist mit »Esc« beschriftet.

Die folgenden Tasten befinden sich im Mittelblock zwischen Buchstabenblock und Ziffernblock. Gleichbeschriftete Tasten im Ziffernblock können eventuell alternativ verwendet werden. Bei Tastaturen ohne Ziffernblock sind die folgenden Tasten in der rechten oberen und rechten unteren Ecke der Tastatur und erfordern eventuell ein zusätzliches Drücken der Fn-Taste.

---

Pos1	»Anfangtaste«, sie ist mit »Pos1«, »Home« oder einem Pfeil nach links oben beschriftet. Unter macOS kann alternativ $f_n+P_fL$ verwendet werden.
End	»Endetaste«, sie ist mit »Ende«, »End« oder einem Pfeil nach rechts unten beschriftet. Unter macOS kann alternativ $f_n+P_fR$ verwendet werden.
PgUp	»Bild-rauf-Taste«, sie ist mit »Page Up«, »PgUp« oder mit »Bild« und einem Pfeil nach oben beschriftet. Unter macOS kann alternativ $f_n+P_fO$ verwendet werden.
PgDn	»Bild-runter-Taste«, sie ist mit »Page Down«, »PgDn« oder mit »Bild« und einem Pfeil nach unten beschriftet. Unter macOS kann alternativ $f_n+P_fU$ verwendet werden.

## Gemeinsame Steuerbefehle für alle Menü-Fenster

CUR\_DN / CUR\_UP (Pfeil nach unten / Pfeil nach oben)

Positioniert den Cursor um eine Zeile nach unten/oben.

CR (Return, Ctrl+M)

Wählt das aus, was in der Zeile angegeben ist, in der der Cursor steht.

Um diese Auswahl zu treffen, kann (unabhängig von der Cursor-Position) alternativ auch die entsprechende Zeile mit der linken Maustaste angeklickt werden.

ENTER (Enter, Shift+Return, Ctrl+E)

Bestätigt die im Fenster getroffene Auswahl und führt entsprechende Aktionen aus, wobei ggf. das Fenster gewechselt wird.

Um die getroffene Auswahl zu betätigen, kann alternativ auch das OK-Feld rechts oben im Fenster mit der linken Maustaste angeklickt werden.

CANCEL (Esc, Ctrl+D)

Verwirft die vorgenommenen Änderungen und kehrt ins vorherige Fenster zurück bzw. beendet das Programm.

REFRESH (Ctrl+W)

Stellt den durch Systemmeldungen, Leitungsstörungen u. ä. zerstörten Inhalt des TUSTEP-Fensters wieder her.

COLOR (Ctrl+F)

Leitet das Ändern der Farbeinstellung ein.

PRINT (Ctrl+P)

Gibt den Inhalt des TUSTEP-Fensters in das Zweitprotokoll aus; dies geschieht unabhängig davon, ob das Zweitprotokoll ein- oder ausgeschaltet ist (vgl. Kommando #PROTOKOLL Seite 209).

## Gemeinsame Steuerbefehle für alle Daten-Fenster

SKP\_BEG / SKP\_END (Pos1 / End)

Zeigt den Anfang / das Ende an.

SHW\_DN / SHW\_UP (Bild runter, PgDn / Bild rauf, PgUp)

Blättert vorwärts / rückwärts (überlappend).

CUR\_DN / CUR\_UP (Pfeil nach unten / Pfeil nach oben)

Positioniert den Cursor um eine Zeile nach unten/oben; falls der Cursor unten/oben im Fenster steht, wird der Inhalt des Fensters um eine Zeile nach oben/unten verschoben.

CUR\_RI / CUR\_LE (Pfeil nach rechts / Pfeil nach links)

Sucht den (von der Cursor-Position aus gesehen) nachfolgenden/vorhergehenden markierten Indexeintrag und zeigt die entsprechenden Daten an.

ENTER (Enter, Shift+Return, Ctrl+E)

Wechselt zum vorherigen Fenster.

Um zum vorherigen Fenster zu wechseln, kann alternativ auch das Pfeil-Feld rechts oben im Fenster mit der linken Maustaste angeklickt werden.

REFRESH (Ctrl+W)

Stellt den durch Systemmeldungen, Leitungsstörungen u. ä. zerstörten Inhalt des TUSTEP-Fensters wieder her.

COLOR (Ctrl+F)

Leitet das Ändern der Farbeinstellung ein.

WINDOW (Ctrl+G)

Leitet das Ändern der Fenstergröße bzw. das Verschieben des Fensters ein.

Das Ändern der Fenstergröße bzw. das Verschieben des Fensters kann alternativ auch mit der Maus vorgenommen werden und ist im Kapitel »Fenstergröße ändern, Fenster verschieben« ab Seite 690 beschrieben.

PRINT (Ctrl+P)

Gibt den Inhalt des TUSTEP-Fensters in das Zweitprotokoll aus; dies geschieht unabhängig davon, ob das Zweitprotokoll ein- oder ausgeschaltet ist (vgl. Kommando #PROTOKOLL Seite 209).

## Steuerbefehle innerhalb der Eingabefelder

CUR\_RI / CUR\_LE (Pfeil nach rechts / Pfeil nach links)

Positioniert den Cursor um ein Zeichen nach rechts / links.

SKP\_BEG / SKP\_END (Pos1 / End)

Positioniert den Cursor an den Anfang des Eingabefeldes / hinter das letzte eingegebene Zeichen im Eingabefeld.

DEL\_BEG (Plus-Pos1)

Löscht im Eingabefeld alle Zeichen links von der Cursor-Position. Alle im Eingabefeld folgenden Zeichen werden um die entsprechende Anzahl Zeichen nach links verschoben. Der Cursor steht danach am Anfang des Eingabefeldes.

DEL\_END (Plus-End)

Löscht im Eingabefeld alle Zeichen von der Cursor-Position bis zum Ende des Eingabefeldes.

TGL\_INS (Einfüg, Ins, Plus-Enter)

Schaltet vom Überschreibmodus in den Einfügemodus und umgekehrt. Im Überschreibmodus werden bereits vorhandene Zeichen überschrieben; im Einfügemodus werden die neu eingegebenen Zeichen an der Cursor-Position eingeschoben, d. h. alle Zeichen von der Cursor-Position bis zum Ende des Eingabefeldes werden nach rechts verschoben. Ist das Eingabefeld voll, können keine weiteren Zeichen mehr eingegeben werden.

DEL (Entf, Del, Shift+Backspace)

Löscht das Zeichen auf der Cursor-Position und verschiebt alle im Eingabefeld folgenden Zeichen um eine Position nach links.

BSP (Backspace, Rücktaste)

Löscht das Zeichen links von der Cursor-Position und verschiebt alle im Eingabefeld folgenden Zeichen um eine Position nach links. Der Cursor rückt ebenfalls um eine Position nach links.

CLEAR (Plus-Stern)

Löscht alle Zeichen im Eingabefeld und positioniert den Cursor an den Anfang des Eingabefeldes.

CR (Return, Ctrl+M)

Prüft den Inhalt des Eingabefeldes auf formale Fehler. Werden keine festgestellt, wird der Cursor in das nächste Eingabefeld bzw. in die nächste Zeile positioniert. Andernfalls wird der Cursor auf die Stelle im Eingabefeld positioniert, an der der Fehler festgestellt worden ist.



## Gehe-zu-Fenster

TUSTEP-Recherche: Beispieltext		Gehe zu
Gehe zu	Auswählen des nächsten Arbeitsschrittes	O K
<p>Anzeigen der Liste aller Register und</p> <ul style="list-style-type: none"> <li><input type="radio"/> Indexeinträge definieren</li> </ul> <p>Markieren von Einträgen im Index</p> <ul style="list-style-type: none"> <li><input type="radio"/> mit automatischer Auswahl</li> <li><input checked="" type="radio"/> manuell (mit CR bzw. DEL)</li> </ul> <p>Anzeigen der Daten mit den markierten Einträgen</p> <ul style="list-style-type: none"> <li><input type="radio"/> im KWIC-Format</li> <li><input type="radio"/> mit Umgebung</li> <li><input type="radio"/> im Volltext</li> </ul> <p>Programm vorübergehend unterbrechen und</p> <ul style="list-style-type: none"> <li><input type="radio"/> Editor aufrufen</li> </ul> <p>Löschen der Einträge im Index und</p> <ul style="list-style-type: none"> <li><input type="radio"/> Programm beenden</li> </ul>		

Zum »Anzeigen der Liste aller Register« wird ins Register-Fenster gewechselt, zum »Markieren von Einträgen« wird ins Auswahl-Fenster oder ins Index-Fenster gewechselt, zum »Anzeigen der Daten« ins KWIC-Fenster, ins Umgebungs-Fenster oder ins Volltext-Fenster; beim Aufruf des Editors wird ins Editorfenster gewechselt.

### Steuerbefehle

CUR\_DN / CUR\_UP (Pfeil nach unten / Pfeil nach oben)

Positioniert den Cursor um eine Zeile nach unten/oben.

CR (Return, Ctrl+M)

Wählt den Arbeitsschritt aus, der in der Zeile angegeben ist, in der der Cursor steht.

Um einen Arbeitsschritt auszuwählen, kann (unabhängig von der Cursor-Position) alternativ auch die entsprechende Zeile mit der linken Maustaste angeklickt werden.

ENTER (Enter, Shift+Return, Ctrl+E)

Bestätigt die im Fenster getroffene Auswahl und wechselt in das entsprechende Fenster.

Um die getroffene Auswahl zu betätigen, kann alternativ auch das OK-Feld rechts oben im Fenster mit der linken Maustaste angeklickt werden.

CANCEL (Esc, Ctrl+D)

Beendet das Programm.

## Register-Fenster

TUSTEP-Recherche: Beispieltext		Gehe zu
Register	Auswählen der Einträge für den Index	<input type="radio"/> <b>OK</b>
	<ul style="list-style-type: none"> <li><input type="radio"/> o Ortsnamen</li> <li><input type="radio"/> p Personennamen</li> <li><input type="radio"/> - sonstige Wörter</li> </ul>	

Im Register-Fenster sind alle für den jeweiligen Text vorhandenen Register aufgelistet. Vor den Namen der Register (z. B. »Personennamen«) ist jeweils ein Kennzeichen (z. B. »p«) für das Register angegeben. Über dieses Kennzeichen kann im Index-, KWIC- und Umgebungs-Fenster erkannt werden, aus welchem Register ein Indexeintrag stammt. Das Wortformenregister, das alle Wörter enthält, die in keinem anderen Register enthalten sind, hat immer das Kennzeichen »-« und ist hier unter »sonstige Wörter« aufgeführt.

## Steuerbefehle

CUR\_DN / CUR\_UP (Pfeil nach unten / Pfeil nach oben)

Positioniert den Cursor um eine Zeile nach unten/oben.

CR (Return, Ctrl+M)

Wählt das Register für den Index aus, das in der Zeile angegeben ist, in der der Cursor steht.

Um ein Register auszuwählen, kann (unabhängig von der Cursor-Position) alternativ auch die entsprechende Zeile mit der linken Maustaste angeklickt werden.

DEL (Entf, Del, Shift+Backspace, Shift+Rücktaste)

Hebt die Auswahl des Registers, das in der Zeile angegeben ist, in der der Cursor steht, für den Index auf.

Um eine Auswahl aufzuheben, kann (unabhängig von der Cursor-Position) alternativ auch die entsprechende Zeile mit der linken Maustaste angeklickt werden.

ENTER (Enter, Shift+Return, Ctrl+E)

Bestätigt die im Fenster getroffene Auswahl und wechselt in das vorherige Fenster.

Um die getroffene Auswahl zu betätigen, kann alternativ auch das OK-Feld rechts oben im Fenster mit der linken Maustaste angeklickt werden.

CANCEL (Esc, Ctrl+D)

Macht die im Fenster vorgenommenen Änderungen rückgängig und wechselt zum vorherigen Fenster.

## Auswahl-Fenster

TUSTEP-Recherche: Beispieltext		Gehe zu
Auswahl	Auswählen der Index-Einträge zum Markieren	<input type="radio"/> <b>OK</b>
<p>Auswahl (zum Markieren) aller im Index  x vorkommenden Einträge (254)  <input type="radio"/> schon markierten Einträge (0)  <input type="radio"/> noch nicht markierten Einträge (254)</p> <p>Davon nur die, die eine der folg. Zeichenfolgen enthalten</p> <div style="border: 1px solid black; height: 20px; width: 100%;"></div> <p>Davon nur die, die keine der folg. Zeichenfolgen enthalten</p> <div style="border: 1px solid black; height: 20px; width: 100%;"></div> <p>Markierung der ausgewählten Einträge</p> <input checked="" type="radio"/> manuell setzen (mit CR) bzw. löschen (mit DEL) <input type="radio"/> automatisch setzen (ohne Anzeige) <input type="radio"/> automatisch löschen (ohne Anzeige) <input type="radio"/> abschließen / Daten anzeigen (bzw. Programm beenden)		

## Manuelles Setzen/Löschen von Markierungen

Es wird geprüft, auf wieviele Indexeinträge die angegebenen Bedingungen zutreffen. Treffen sie auf keinen zu, wird dies im Kopftext angezeigt; diese Meldung muss mit der Enter-Taste oder Anklicken des OK-Feldes mit der linken Maustaste bestätigt werden:

Auswahl	0 Treffer	O K
---------	-----------	-----

Andernfalls wird gefragt, ob weitergemacht werden soll:

Auswahl	90 Treffer	Weiter ?	Nein	J a
---------	------------	----------	------	-----

Wird mit »j« (für ja) geantwortet oder die Enter-Taste gedrückt oder das JA-Feld rechts oben im Fenster mit der linken Maustaste angeklickt, wird ins Index-Fenster gewechselt. Dort werden nur die im Auswahl-Fenster ausgewählten Indexeinträge angezeigt.

Wird mit »n« (für nein) geantwortet, die Delete-Taste gedrückt oder das NEIN-Feld rechts oben im Fenster mit der linken Maustaste angeklickt, wird nichts weiter gemacht (auch nicht ins Index-Fenster gewechselt).

### Automatisches Setzen/Löschen von Markierungen

Es wird geprüft, auf wieviele Indexeinträge die angegebenen Bedingungen zutreffen. Treffen sie auf keinen zu, wird dies im Kopftext angezeigt; diese Meldung muss mit der Enter-Taste oder Anklicken des OK-Feldes mit der linken Maustaste bestätigt werden:

Auswahl	0 Treffer	O K
---------	-----------	-----

Andernfalls wird gefragt, ob die Markierungen gesetzt/gelöscht werden sollen:

Auswahl	90 Treffer	70 Markierungen setzen ?	Nein	J a
---------	------------	--------------------------	------	-----

bzw.

Auswahl	90 Treffer	20 Markierungen löschen ?	Nein	J a
---------	------------	---------------------------	------	-----

Wird mit »j« (für ja) geantwortet oder die Enter-Taste gedrückt oder das JA-Feld rechts oben im Fenster mit der linken Maustaste angeklickt, werden die Markierungen der betroffenen Indexeinträge ohne weitere Anzeige gesetzt/gelöscht.

Wird mit »n« (für nein) geantwortet, die Delete-Taste gedrückt oder das NEIN-Feld rechts oben im Fenster mit der linken Maustaste angeklickt, werden keine Markierungen gesetzt/gelöscht.

Falls von den Indexeinträgen, auf die die Bedingungen zutreffen, schon welche markiert sind (beim Setzen) bzw. nicht alle markiert sind (beim Löschen), unterscheidet sich in der Anzeige die Anzahl der Treffer und die Anzahl der Markierungen, die gesetzt/gelöscht werden.

Falls von den Indexeinträgen, auf die die Bedingungen zutreffen, schon alle markiert sind (beim Setzen) bzw. keiner markiert ist (beim Löschen), wird dies im Kopftext angezeigt; diese Meldung muss mit der Enter-Taste oder Anklicken des OK-Feldes mit der linken Maustaste bestätigt werden:

Auswahl	90 Treffer	Jeder Eintrag schon markiert	<input type="radio"/> K
---------	------------	------------------------------	-------------------------

bzw.

Auswahl	90 Treffer	Kein Eintrag davon markiert	<input type="radio"/> K
---------	------------	-----------------------------	-------------------------

## Steuerbefehle

CUR\_DN / CUR\_UP (Pfeil nach unten / Pfeil nach oben)

Positioniert den Cursor um eine Zeile nach unten/oben.

CR (Return, Ctrl+M)

Wählt die Indexeinträge bzw. den Arbeitsschritt aus, der in der Zeile angegeben ist, in der der Cursor steht.

Um diese Auswahl zu treffen, kann (unabhängig von der Cursor-Position) alternativ auch die entsprechende Zeile mit der linken Maustaste angeklickt werden.

ENTER (Enter, Shift+Return, Ctrl+E)

Bestätigt die im Fenster getroffene Auswahl und führt den entsprechenden Arbeitsschritt aus.

Um die getroffene Auswahl zu betätigen, kann alternativ auch das OK-Feld rechts oben im Fenster mit der linken Maustaste angeklickt werden.

CANCEL (Esc, Ctrl+D)

Macht die seit dem letzten Wechsel in das Auswahl-Fenster vorgenommenen Änderungen bei den Markierungen rückgängig und wechselt zum vorherigen Fenster.

## Eingabefelder

In den Eingabefeldern gelten eigene Steuerbefehle; sie sind ab Seite 671 beschrieben.

Im oberen Eingabefeld des Auswahl-Fensters können Zeichenfolgen angegeben werden, von denen mindestens eine in einem Indexeintrag vorkommen muss, damit er ausgewählt wird.

Im unteren Eingabefeld können Zeichenfolgen angegeben werden, von denen keine in einem Indexeintrag vorkommen darf, damit er ausgewählt wird.

Die Zeichenfolgen müssen durch ein Leerzeichen getrennt werden.

Das Fragezeichen und der Stern haben eine spezielle Bedeutung. Ein Fragezeichen steht stellvertretend für ein beliebiges TUSTEP-Zeichen und ein Stern für null bis beliebig viele beliebige TUSTEP-Zeichen.

An Stelle eines einzelnen Zeichens einer Zeichenfolge sind folgende Angaben möglich:

- Eine in eckige Klammern eingeschlossene Gruppe von Zeichen.

Die Zeichen innerhalb der eckigen Klammern werden als eine Zeichengruppe mit den angegebenen Zeichen interpretiert.

- Ein Backslash und ein beliebiges Zeichen.

Ein Backslash bewirkt, dass das unmittelbar nachfolgende Zeichen seine eigentliche Bedeutung behält. Soll z. B. nach einem Fragezeichen gesucht werden, muss »\?« angegeben werden. Erforderlich ist eine Kennzeichnung mit dem Backslash für Fragezeichen, Stern, Backslash und sämtliche Klammern.

- Eine der folgenden Zeichengruppen-Kennungen:

{ \a }	alle Kleinbuchstaben des TUSTEP-Zeichensatzes
{ \A }	alle Großbuchstaben des TUSTEP-Zeichensatzes
{ \0 }	alle Ziffern des ASCII-Zeichensatzes
{ &0 }	alle Ziffern des TUSTEP-Zeichensatzes
{ &a }	alle Buchstaben des TUSTEP-Zeichensatzes
{ ! }	alle Zeichen des ASCII-Zeichensatzes
{ ; }	alle Zeichen des TUSTEP-Zeichensatzes, die nicht im ASCII-Zeichensatz enthalten sind
{ % }	alle Zeichen, die zur Akzent-Codierung nach % stehen können
{ @ }	alle Sonderzeichen des TUSTEP-Zeichensatzes einschließlich Leerzeichen

Enthält eine Zeichenfolge Buchstaben, so werden dafür die entsprechenden Groß- und Kleinbuchstaben gesucht. Soll zwischen Groß- und Kleinbuchstaben unterschieden werden, so muss dem jeweiligen Buchstaben ein Backslash vorangesetzt werden. Es bedeutet also »a« und »A« ein großes oder kleines a, »\a« ein kleines a und »\A« ein großes a.

Soll eine angegebene Zeichenfolge mit dem Anfang eines Indexeintrags übereinstimmen, so muss »{ [ ] « am Anfang der Zeichenfolge eingefügt werden. Entsprechend muss »{ } « am Ende der Zeichenfolge eingefügt werden, wenn die angegebene Zeichenfolge mit dem Ende eines Indexeintrags übereinstimmen soll. »{ [ ] abc« betrifft also alle Indexeinträge, die mit »abc« beginnen, und »xyz { } « alle Indexeinträge, die mit »xyz« enden (die Anführungszeichen dürfen nicht mit angegeben werden).

Die möglichen Angaben entsprechen denen einer Recherchier-Tabelle (Parameterart XII). Diese ist für die Parameter-Konvention »{ } « ab Seite 724 und für die Parameter-Konvention »<>« ab Seite 756 beschrieben. Nach welcher Konvention die Parameter interpretiert werden, kann mit dem Kommando #PARAMETER (siehe Seite 207) eingestellt werden.

## Index-Fenster

TUSTEP-Recherche: Beispieltext		Gehe zu
Index		O K
-	können . . . . .	3
-	kurzem . . . . .	1
-	macht . . . . .	1
-	Mann . . . . .	1
-	mehr . . . . .	1
-	mehrere . . . . .	3
-	mehrfach . . . . .	1
-	Mit . . . . .	6
p	Müller, Peter . . . . .	9
o	München . . . . .	9
-	muss . . . . .	6
-	müssen . . . . .	3
-	müsste . . . . .	1
-	nach . . . . .	1
-	nachfragen . . . . .	1
-	namens . . . . .	1
-	nicht . . . . .	9
-	nichts . . . . .	1
-	noch . . . . .	2

Die Indexeinträge werden im Index-Fenster (auch wenn sie aus verschiedenen Registern stammen) in einem einzigen Alphabet sortiert angezeigt.

Links vor dem Indexeintrag wird das Kennzeichen des Registers (vgl. Register-Fenster) angezeigt, aus dem der jeweilige Eintrag stammt. Falls es außer dem Wortformenregister keine weiteren Register gibt, wird das Kennzeichen nicht angezeigt. Hier im Beispiel kommt der Eintrag »Müller, Peter« aus dem Personenregister (Kennzeichen »p«) und »München« aus dem Ortsregister (Kennzeichen »o«). Die anderen Indexeinträge stammen alle aus dem Wortformenregister (Kennzeichen »-«).

Rechts hinter dem Indexeintrag wird die absolute Häufigkeit des jeweiligen Indexeintrags angegeben.

Beim Wechseln ins Index-Fenster wird geprüft, ob schon/noch Indexeinträge markiert sind. Wenn dies der Fall ist, wird nachgefragt, ob diese Markierungen beibehalten werden sollen:

Index	Alte Markierungen beibehalten ?	Nein	J a
-------	---------------------------------	------	-----

Auf diese Frage muss geachtet (sie wird erfahrungsgemäß leicht übersehen) und geantwortet werden; erst dann werden wieder andere Eingaben akzeptiert.

Sollen die Markierungen erhalten bleiben, kann »j« (für ja, beibehalten) eingegeben, die Enter-Taste gedrückt, oder das JA-Feld rechts oben im Fenster mit der linken Maustaste angeklickt werden.

Sollen die Markierungen gelöscht werden, kann »n« (für nein, nicht beibehalten) eingegeben, die Delete-Taste gedrückt oder das NEIN-Feld rechts oben im Fenster mit der linken Maustaste angeklickt werden.

## Steuerbefehle

SKP\_BEG / SKP\_END (Pos1 / End)

Zeigt den Anfang / das Ende an.

SHW\_DN / SHW\_UP (Bild runter, PgDn / Bild rauf, PgUp)

Blättert vorwärts / rückwärts (überlappend).

CUR\_DN / CUR\_UP (Pfeil nach unten / Pfeil nach oben)

Positioniert den Cursor um eine Zeile nach unten/oben; falls der Cursor unten/oben im Fenster steht, wird der Inhalt des Fensters um eine Zeile nach oben/unten verschoben.

CUR\_RI / CUR\_LE (Pfeil nach rechts / Pfeil nach links)

Sucht den (von der Cursor-Position aus gesehen) nachfolgenden/vorhergehenden markierten Indexeintrag und positioniert den Cursor auf diesen Eintrag; falls sich dieser Eintrag nicht innerhalb des Fensters befindet, wird er ins Fenster geholt.

CR (Return, Ctrl+M)

Markiert den Eintrag, der in derselben Zeile wie der Cursor steht.

Um einen Eintrag zu markieren, kann (unabhängig von der Cursor-Position) alternativ auch die entsprechende Zeile mit der linken Maustaste angeklickt werden.

DEL (Entf, Del, Shift+Backspace, Shift+Rücktaste)

Löscht die Markierung des Eintrags, der in derselben Zeile wie der Cursor steht.

Um die Markierung eines Eintrags zu löschen, kann (unabhängig von der Cursor-Position) alternativ auch die entsprechende Zeile mit der linken Maustaste angeklickt werden.

CLEAR (Plus-Stern)

Löscht alle Markierungen.

Buchstabe:

Sucht den ersten Indexeintrag, der mit dem eingegebenen Buchstaben beginnt, und positioniert den Cursor auf diesen Eintrag; falls sich dieser Eintrag nicht innerhalb des Fensters befindet, wird er ins Fenster geholt.

Werden unmittelbar danach weitere Buchstaben eingegeben, so werden sie mit dem ersten zu einer Zeichenfolge zusammengefügt, und der Cursor wird jeweils auf den ersten Eintrag positioniert, der mit dieser Zeichenfolge beginnt.



Beginnt kein Indexeintrag mit dem eingegebenen Buchstaben bzw. mit der eingegebenen Zeichenfolge, so wird auf den in der alphabetischen Ordnung folgenden Eintrag positioniert.

Umlaute und ß sind im Index wie die entsprechenden Grundbuchstaben a, o, u und s einsortiert. Deshalb kann statt der Umlaute und ß auch der jeweilige Grundbuchstabe eingegeben werden. Groß- und Kleinbuchstaben werden jeweils gleich behandelt.

BSP (Backspace, Rücktaste)

Entfernt den letzten Buchstaben von der eingegebenen Zeichenfolge und positioniert den Cursor auf den ersten Eintrag, der mit dieser verkürzten Zeichenfolge beginnt.

ENTER (Enter, Shift+Return, Ctrl+E)

Bestätigt die Markierungen und wechselt zum vorherigen Fenster.

Um die Markierungen zu bestätigen, kann alternativ auch das OK-Feld rechts oben im Fenster mit der linken Maustaste angeklickt werden.

F11

Zeigt den Index normal sortiert an.

F12

Zeigt den Index rückläufig sortiert an.

## Beispiel

Im folgenden Fenster wird der Index rückläufig sortiert angezeigt; standardmäßig wird er normal sortiert angezeigt.

TUSTEP-Recherche: Beispieltext		Gehe zu
Index		O K
3	. . . . . dürfen	-
1	. . . . . Fragen	-
1	. . . . . nachfragen	-
1	. . . . . eingetragen	-
4	. . . . . Sonntagen	-
1	. . . . . Feiertagen	-
1	. . . . . eckigen	-
1	. . . . . jeweiligen	-
4	. . . . . Tübingen	o
1	. . . . . gesprungen	-
1	. . . . . umgezogen	-
1	. . . . . gleichen	-
3	. . . . . Registerkennzeichen	-
1	. . . . . Gleichheitszeichen	-
3	. . . . . zusätzlichen	-
9	. . . . . München	o
4	. . . . . stehen	-
2	. . . . . Zeilen	-
1	. . . . . Textstellen	-

Wenn der Index rückläufig sortiert angezeigt wird, gelten die gleichen Steuerbefehle wie wenn er normal sortiert angezeigt wird. Werden Buchstaben eingegeben, so wird jedoch jeweils nach dem ersten Indexeintrag gesucht, der mit der eingegebenen Zeichenfolge endet (statt beginnt).



**Buchstabe:**

Sucht den ersten markierten Indexeintrag, der mit dem eingegebenen Buchstaben beginnt, und zeigt die dazugehörigen Textstellen an.

Werden unmittelbar danach weitere Buchstaben eingegeben, so werden sie mit dem ersten zu einer Zeichenfolge zusammengefügt, und es wird jeweils der erste markierte Indexeintrag gesucht, der mit dieser Zeichenfolge beginnt, und die dazugehörigen Textstellen werden angezeigt.

Beginnt kein markierter Indexeintrag mit dem eingegebenen Buchstaben bzw. mit der eingegebenen Zeichenfolge, so wird auf den in der alphabetischen Ordnung folgenden markierten Eintrag positioniert.

Umlaute und ß sind im Index wie die entsprechenden Grundbuchstaben a, o, u und s einsortiert. Deshalb kann statt der Umlaute und ß auch der jeweilige Grundbuchstabe eingegeben werden. Groß- und Kleinbuchstaben werden jeweils gleich behandelt.

**BSP (Backspace, Rücktaste)**

Entfernt den letzten Buchstaben von der eingegebenen Zeichenfolge, sucht den ersten markierten Indexeintrag, der mit dieser verkürzten Zeichenfolge beginnt, und zeigt die dazugehörigen Textstellen an.

**ENTER (Enter, Shift+Return, Ctrl+E)**

Wechselt zum vorherigen Fenster.

Um zum vorherigen Fenster zu wechseln, kann alternativ auch das Pfeil-Feld rechts oben im Fenster mit der linken Maustaste angeklickt werden.

**F11**

Blendet die Seiten- und Zeilennummern (Satznummern) ein bzw. aus.

**F12**

Blendet die Referenzen ein bzw. aus.

**Beispiel**

Im folgenden Fenster wurden sowohl die Seiten- und Zeilennummern als auch die Referenzen (hier im Beispiel dient jeweils eine Kurzform der Kapitelüberschrift als Referenz) eingeblendet; sie werden standardmäßig nicht angezeigt:



ferenzen mit der Funktionstaste F11 bzw. F12 eingeblendet worden; sie werden standardmäßig nicht angezeigt.

## Steuerbefehle und Eingaben

Im Umgebungs-Fenster sind außer den Steuerbefehlen/Eingaben, die beim KWIC-Fenster beschrieben sind, zusätzlich folgende Steuerbefehle, die beim Volltext-Fenster beschrieben sind, möglich: SKP\_WORD, CR, F13.

## Volltext-Fenster

TUSTEP-Recherche: Beispieltext		Gehe zu
Volltext		<--
<pre> 1.1  Beispieltext für das Kommando #SUCHE 1.2 1.3 2.1  &lt;*&gt;Einleitung 2.2 2.3  Dies ist ein einfacher Beispieltext. Er enthält 2.4  sowohl Personennamen (z.B. --&gt;Müller, Peter) 2.5  als auch Ortsnamen (z.B. --&gt;Tübingen und 2.6  --&gt;München). Außerdem kommen --&gt;Fußnoten und 2.7  --&gt;Verweise <b>auf</b> andere Textstellen vor. Alles 2.8  klar?[[1]] Wenn nicht, helfen vielleicht die 2.9  folgenden sechs Kapitel weiter. 2.10 3.1  Kapitel 1: &lt;*&gt;Satznummern 3.2 3.3  Die Sätze der Datei müssen so nummeriert sein, dass 3.4  - alle Satznummern aufsteigend sind,</pre>		

Im obigen Volltext-Fenster sind die Seiten- und Zeilennummern mit der Funktionstaste F11 eingeblendet worden; sie werden standardmäßig nicht angezeigt.

## Steuerbefehle und Eingaben

SKP\_BEG / SKP\_END (Pos1 / End)

Zeigt den Anfang / das Ende an.

SHW\_DN / SHW\_UP (Bild runter, PgDn / Bild rauf, PgUp)

Blättert vorwärts / rückwärts (überlappend).

CUR\_DN / CUR\_UP (Pfeil nach unten / Pfeil nach oben)

Positioniert den Cursor um eine Zeile nach unten/oben; falls der Cursor unten/oben im Fenster steht, wird der Inhalt des Fensters um eine Zeile nach oben/unten verschoben.

CUR\_RI / CUR\_LE (Pfeil nach rechts / Pfeil nach links)

Sucht die (von der Cursor-Position aus gesehen) nachfolgende/vorhergehende Zeile mit einem markierten Indexeintrag und positioniert den Cursor auf diese Zeile; falls sich diese Zeile nicht innerhalb des Fensters befindet, wird sie ins Fenster geholt.

Seite.Zeile CR

Zeigt die Daten mit der eingegebenen Seiten- und Zeilennummer an.

Wird die Seiten- oder die Zeilennummer nicht angegeben, wird dafür die aktuelle Seiten- bzw. Zeilennummer (von der Zeile, in der der Cursor steht) intern ergänzt. Der Punkt muss in jedem Fall angegeben werden.

BSP (Backspace, Rücktaste)

Entfernt das letzte Zeichen von der eingegebenen Seiten-Zeilen-Nummer.

SKP\_WORD (Minus)

Positioniert den Cursor in der aktuellen Zeile auf den nächsten Indexeintrag oder auf den nächsten Verweis und ermöglicht damit das »Anklicken« mit der Taste CR.

CR (Return, Ctrl+M)

»Klickt« den Indexeintrag bzw. den Verweis an, auf dem der Cursor steht.

Das Anklicken kann (unabhängig von der Cursor-Position) alternativ auch mit der linken Maustaste erfolgen.

Wird ein Indexeintrag angeklickt, so wird ins Index-Fenster gewechselt und der Cursor auf den entsprechenden Indexeintrag positioniert; falls sich dieser Eintrag nicht innerhalb des Fensters befindet, wird er ins Fenster geholt.

Wird ein Fußnotenverweis angeklickt, so wird die entsprechende Fußnote in einem Fußnoten-Fenster angezeigt.

Wird ein interner Verweis angeklickt, so wird die entsprechende Textstelle aufgesucht und (im gleichen Fenster) angezeigt.

Wird ein externer Verweis angeklickt, so wird die entsprechende Textstelle in einem Zusatz-Fenster angezeigt.

ENTER (Enter, Shift+Return, Ctrl+E)

Wechselt zum vorherigen Fenster.

Um zum vorherigen Fenster zu wechseln, kann alternativ auch das Pfeil-Feld rechts oben im Fenster mit der linken Maustaste angeklickt werden.

F11

Blendet die Seiten- und Zeilennummern (Satznummern) ein bzw. aus.

F12

Blendet die Referenzen ein bzw. aus.

F13 (nur wenn Fußnoten im Text stehen):

Schaltet die Anzeige von Fußnotentext ein bzw. aus.

## Beispiel

Im folgenden Fenster wurden sowohl die Seiten- und Zeilennummern eingeblendet als auch die Anzeige der im Text stehenden Fußnoten eingeschaltet.

TUSTEP-Recherche: Beispieltext		Gehe zu
Volltext		<--
1.3		
2.1	<*>Einleitung	
2.2		
2.3	Dies ist ein einfacher Beispieltext. Er enthält	
2.4	sowohl Personennamen (z.B. -->Müller, Peter)	
2.5	als auch Ortsnamen (z.B. -->Tübingen und	
2.6	-->München). Außerdem kommen -->Fußnoten und	
2.7	-->Verweise auf andere Textstellen vor. Alles	
2.8	klar?[[1]] Wenn nicht, helfen vielleicht die	
2.8	[] [[*1]] Es kann doch nicht alles klar sein,	
2.8	[] denn es wurde ja noch fast nichts erklärt!	
2.9	folgenden sechs Kapitel weiter.	
2.10		

Fußnotenzeilen sind mit »>[]« gekennzeichnet.

Die Satznummern der Sätze einer Fußnote unterscheiden sich von der Satznummer des vorangehenden Satzes mit normalem Text nur in der Unterscheidungsnummer; da diese nicht angezeigt wird, haben in der Anzeige alle Sätze einer Fußnote die gleiche Seiten- und Zeilennummer (hier im Beispiel 2.8) wie der vorangehende Satz, der normalen Text enthält.



## Editorfenster

TUSTEP-Recherche: Beispieltext		Gehe zu
Editor	Datei TUSTEP*BEISPIEL	<--
<pre> 1.1  (( Titel )) Beispieltext für das Kommando #SUCHE 1.2   1.3   2.1  (( Einleitung )) &lt;&lt;*1&gt;&gt;Einleitung 2.2   2.3  Dies ist ein einfacher Beispieltext. Er enthält 2.4  sowohl Personennamen (z.B. &lt;&lt;7&gt;&gt;((p Müller, Peter)) 2.5  als auch Ortsnamen (z.B. &lt;&lt;8&gt;&gt;((o Tübingen)) und 2.6  &lt;&lt;9&gt;&gt;((o München)). Außerdem kommen &lt;&lt;3&gt;&gt;Fußnoten und 2.7  &lt;&lt;4&gt;&gt;Verweise auf andere Textstellen vor. Alles 2.8  klar?[[1]] Wenn nicht, helfen vielleicht die 2.8/1 [[*1]] Es kann doch nicht alles klar sein, 2.8/2 denn es wurde ja noch fast nichts erklärt! 2.9  folgenden sechs Kapitel weiter. 2.10   3.1  (( Satznummern )) Kapitel 1: &lt;&lt;*2&gt;&gt;Satznummern </pre>		
<pre> &lt;*=1.1&gt; Gib Anweisung &gt; </pre>		
SPLIT		INSERT 16:46

Wird der Editor aufgerufen, so entspricht dies dem Kommando #EDIERE ohne weitere Spezifikationsangaben. Es wird also die zuletzt mit dem Editor edierte Datei im dabei verwendeten Modus ediert; handelt es sich um den ersten Aufruf des Editors nach dem Initialisieren von TUSTEP, so wird die Standard-Editor-Datei im Modus P ediert. Gegebenenfalls muss mit der Datei-Anweisung in die gewünschte Datei gewechselt werden.

Im Editor dürfen alle Editoranweisung mit Ausnahme der X-Anweisung, mit der sonst Kommandos ausgeführt werden können, verwendet werden.

Wird der Editor beendet oder das Pfeil-Feld rechts oben mit der linken Maustaste angeklickt, wird zum vorherigen Fenster gewechselt.

Das Einstellen von Farben, Zeilenlängen und Cursor-Modi ist nicht möglich. Diese müssen bei Bedarf mit dem Editor vor dem Aufruf des Kommandos #SUCHE eingestellt werden.

Während das Editorfenster aktiv ist, werden alle Steuerbefehle vom Editor interpretiert und ausgeführt. Die einzige Ausnahme bildet der Steuerbefehl `Ctrl+G` bzw. `Strg+G` zum Ändern der Fenstergröße und Verschieben des Fensters. Dieser wird an das SUCHE-Programm weitergegeben. Damit ist es möglich, das Editorfenster auch mit der Tastatur in seiner Größe zu verändern und zu verschieben.

## Fenstergröße ändern, Fenster verschieben

Daten-Fenster belegen standardmäßig (abgesehen von der obersten Zeile) das gesamte TUSTEP-Fenster. Die Größe dieser Daten-Fenster kann durch Verschieben der Ränder geändert werden. Jedoch kann dabei eine Minimalgröße nicht unterschritten werden. Sie liegt bei 5 Zeilen und 40 Spalten, beim Index-Fenster bei 20 Spalten, beim Editor bei 76 Spalten (jeweils ohne Kopfzeile und ohne Rahmen).

Menü-Fenster haben eine feste Größe und werden immer in der Mitte des TUSTEP-Fensters angezeigt. Eine Änderung ist nicht möglich.

### Ändern/Verschieben mit der Maus

Werden im TUSTEP-Fenster mehrere Daten-Fenster angezeigt, so muss zuvor in das Daten-Fenster gewechselt werden, das geändert bzw. verschoben werden soll.

Soll der Rand eines Fensters verschoben werden, muss der Mauszeiger auf diesen positioniert und die linke Maustaste gedrückt und festgehalten werden. Dann muss der Mauszeiger in die Richtung bewegt werden, in die der Rand verschoben werden soll. Ist die gewünschte Position erreicht, kann die linke Maustaste wieder losgelassen werden.

Soll das ganze Fenster verschoben werden, muss der Mauszeiger in das Namensfeld links oben im Fenster positioniert und die linke Maustaste gedrückt und festgehalten werden. Dann muss der Mauszeiger in die Richtung bewegt werden, in die das Fenster verschoben werden soll. Ist die gewünschte Position erreicht, kann die linke Maustaste wieder losgelassen werden.

### Ändern/Verschieben mit der Tastatur

Werden im TUSTEP-Fenster mehrere Daten-Fenster angezeigt, so muss zuvor in das Fenster gewechselt werden, das geändert bzw. verschoben werden soll.

Mit der Tastenkombination `Ctrl+G` bzw. `Strg+G` kann in jedem Daten-Fenster (nicht in Menü-Fenstern) das Ändern der Fenstergröße bzw. das Verschieben des Fensters eingeleitet werden. Es wird dann in der Kopfzeile des Fensters folgender Text angezeigt:

```
Welchen Rand verschieben ? (mit Pfeil-/Leertaste angeben)
```

Nun kann mit Pfeil nach oben/unten/links/rechts angegeben werden, dass der obere/untere/linke/rechte Rand des Fensters verschoben werden soll. Ist dies geschehen, wird z. B. nach »Pfeil nach oben« folgender Text angezeigt:

```
Oberen Rand mit Pfeiltaste verschieben (oder mit CR abschl.)
```

Jetzt kann mit Pfeil nach oben/unten (bzw. mit Pfeil nach links/rechts, wenn vorher der linke oder rechte Rand angegeben wurde) der obere (bzw. der vorher angegebene) Rand verschoben werden. Das Ende des Verschiebens muss mit CR angezeigt werden. Daraufhin wird wieder gefragt, welcher Rand verschoben werden soll.

Soll das ganze Fenster verschoben werden, so muss auf die Frage, welcher Rand verschoben werden soll, mit der Leertaste (statt mit einer Pfeiltaste) geantwortet werden. Es wird dann folgender Text angezeigt:

```
Fenster mit Pfeiltasten verschieben (oder mit CR abschl.)
```

Nun kann mit Pfeil nach oben/unten/links/rechts das ganze Fenster entsprechend der Pfeilrichtung verschoben werden. Das Ende des Verschiebens muss mit CR angezeigt werden. Daraufhin wird wieder gefragt, welcher Rand verschoben werden soll.

Zum Abschluss müssen die Änderungen mit der Enter-Taste bestätigt werden.

## Beispiel

Nach dem Verkleinern und Verschieben eines Volltext-Fensters, eines Index-Fensters und eines Fußnoten-Fensters kann das TUSTEP-Fenster so aussehen:

TUSTEP-Recherche: Beispieltext		Gehe zu
Volltext		<--
<pre> 1.3 2.1 &lt;*&gt;Einleitung 2.2 2.3 Dies ist ein einfacher Beispieltext. Er enthält 2.4 sowohl Personennamen (z.B. --&gt;Müller, Peter) 2.5 als auch Ortsnamen (z.B. --&gt;Tübingen und 2.6 --&gt;München). Außerdem kommen --&gt;Fußnoten und 2.7 --&gt;Verweise auf andere Textstellen vor. Alles 2.8 klar?[[1]] Wenn nicht, helfen vielleicht die 2.9 folgenden sechs Kapitel weiter.</pre>		
Index		Fußnoten
<pre> für . . . . . 3 * Fußnote . . . 4 [[*1]] Es kann nicht alles klar sein, * Fußnoten . . 7 denn es wurde ja noch nichts erklärt! Fußnotennummer 3 Fußnotenverweis 1</pre>		

## Wechseln in ein anderes Fenster

### Wechseln mit der Maus

Wenn das Daten-Fenster, in das gewechselt werden soll, im TUSTEP-Fenster (ganz oder teilweise) zu sehen ist, genügt es, mit der linken Maustaste eine beliebige Stelle dieses Daten-Fensters anzuklicken, um in dieses Fenster zu wechseln.

Soll in das vorherige Fenster zurückgekehrt werden, kann das Pfeil-Feld rechts oben im aktuellen Fenster mit der linken Maustaste angeklickt werden.

Wird die oberste Zeile des TUSTEP-Fensters mit der linken Maustaste angeklickt, so wird das Gehe-zu-Fenster angezeigt. In ihm kann dann das gewünschte Fenster ausgewählt werden.

### Wechseln mit der Tastatur

Soll in das vorherige Fenster zurückgekehrt werden, genügt es (außer im Gehe-zu-Fenster), die Enter-Taste zu drücken. Daraus ergibt sich, dass von einem Fenster, in das vom Gehe-zu-Fenster aus gewechselt wurde, mit der Enter-Taste wieder in das Gehe-zu-Fenster zurückgekehrt werden kann um dann das nächste Fenster auszuwählen.

Mit der Tastenkombination `ALT+x` und `PLUS-x`, wobei `x` der Anfangsbuchstabe des Fensternamens ist, kann direkt in das entsprechende Fenster gewechselt werden; mit `PLUS-i` kann also z. B. ins Index-Fenster gewechselt werden.

## Farbeinstellung ändern

### Ändern mit der Maus

Werden im TUSTEP-Fenster mehrere Daten-Fenster angezeigt, so muss zuvor in das Fenster gewechselt werden, in dem Farben geändert werden sollen.

Mit der Tastenkombination `Ctrl+F` bzw. `Strg+F` kann in jedem Fenster (mit Ausnahme des Farbwahl-Fensters) das Ändern der Farbeinstellung eingeleitet werden. Es wird dann in der obersten Zeile des TUSTEP-Fenster folgender Text angezeigt:

```
Welche Farbe ändern ?      (Pfeiltasten, dann CR)      Extras
```

Nun muss mit der linken Maustaste die Stelle im Fenster angeklickt werden, an der die Farbe geändert werden soll. Daraufhin wird das Farbwahl-Fenster angezeigt.

TUSTEP-Recherche: Beispieltext		Gehe zu													
Farbwahl	Auswählen der Farbe:	60	<input type="radio"/> <b>OK</b>												
00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
01	11	21	31	41	51	61	71	81	91	A1	B1	C1	D1	E1	F1
02	12	22	32	42	52	62	72	82	92	A2	B2	C2	D2	E2	F2
03	13	23	33	43	53	63	73	83	93	A3	B3	C3	D3	E3	F3
04	14	24	34	44	54	64	74	84	94	A4	B4	C4	D4	E4	F4
05	15	25	35	45	55	65	75	85	95	A5	B5	C5	D5	E5	F5
06	16	26	36	46	56	66	76	86	96	A6	B6	C6	D6	E6	F6
07	17	27	37	47	57	67	77	87	97	A7	B7	C7	D7	E7	F7
08	18	28	38	48	58	68	78	88	98	A8	B8	C8	D8	E8	F8
09	19	29	39	49	59	69	79	89	99	A9	B9	C9	D9	E9	F9
0A	1A	2A	3A	4A	5A	6A	7A	8A	9A	AA	BA	CA	DA	EA	FA
0B	1B	2B	3B	4B	5B	6B	7B	8B	9B	AB	BB	CB	DB	EB	FB
0C	1C	2C	3C	4C	5C	6C	7C	8C	9C	AC	BC	CC	DC	EC	FC
0D	1D	2D	3D	4D	5D	6D	7D	8D	9D	AD	BD	CD	DD	ED	FD
0E	1E	2E	3E	4E	5E	6E	7E	8E	9E	AE	BE	CE	DE	EE	FE
0F	1F	2F	3F	4F	5F	6F	7F	8F	9F	AF	BF	CF	DF	EF	FF

Im Farbwahl-Fenster sind 256 Zeichenpaare angegeben, die im günstigsten Fall alle in einer anderen Darstellung (Farbe und Helligkeit der Schrift und des Hintergrundes) erscheinen. Aus ihnen kann die gewünschte Darstellung ausgesucht und das entsprechende Zeichenpaar mit der linken Maustaste angeklickt werden. Dadurch wird dieses Zeichenpaar oben in die Kopfzeile eingetragen.

Zum Abschluss muss die Farbwahl durch Anklicken des OK-Feldes rechts oben im Fenster mit der linken Maustaste bestätigt werden.

Um das jeweils aktive Fenster gegenüber den anderen Fenstern (falls mehrere Daten-Fenster im TUSTEP-Fensters angezeigt werden) hervorzuheben, ändert sich die Farbe des Fensterrahmens automatisch, wenn das Fenster aktiviert (wenn in dieses gewechselt) bzw. deaktiviert (wenn in ein anderes gewechselt) wird. Die Farbe des Rahmens eines aktiven Fensters kann wie die jeder anderen Stelle innerhalb des Fensters geändert werden. Die Farbe, die der Rahmen haben soll, wenn das Fenster inaktiv ist, kann eingestellt werden, indem zunächst das Feld »Extras« rechts in der obersten Zeile des TUSTEP-Fensters mit der linken Maustaste angeklickt wird:

Welche Farbe ändern ?	(Pfeiltasten, dann CR)	Extras
-----------------------	------------------------	--------

Nach dem Anklicken des Feldes »Extras« ändert sich die Anzeige:

Welche Farbe ändern ?	Rahmen	Cursor	Schirm	Rand
-----------------------	--------	--------	--------	------

Nun muss das Wort »Rahmen« angeklickt werden. Es wird dann das Farbwahl-Fenster angezeigt, und die Farbe kann wie oben beschrieben eingestellt werden.

Neben den Farben, die zu einem Fenster gehören, gibt es noch einige, die keinem Daten-Fenster, sondern dem TUSTEP-Fenster zugeordnet sind: Die Farbe der obersten Zeile, des Cursors, des Hintergrundes (an den Stellen, an denen kein Daten-Fenster angezeigt wird) und des Randes (nur unter DOS einstellbar).

Soll die Farbe der obersten Zeile geändert werden, kann dies wie an Stellen innerhalb eines Daten-Fensters geschehen, indem die entsprechende Stelle in dieser Zeile angeklickt wird. Soll eine der anderen eben genannten Farbeinstellungen geändert werden, so muss zuerst das Feld »Extras« rechts in der obersten Zeile des TUSTEP-Fensters mit der linken Maustaste angeklickt werden und dann eines der Wörter »Cursor«, »Schirm« oder »Rand«. Es wird dann ebenfalls das Farbwahl-Fenster angezeigt, und die Farbe des Cursors, des Hintergrundes bzw. des Randes kann wie oben beschrieben eingestellt werden.

## Ändern mit der Tastatur

Das Ändern der Farbe mit der Tastatur geschieht in gleicher Weise wie mit der Maus, jedoch mit folgenden Ausnahmen:

Statt eine Stelle mit der linken Maustaste anzuklicken, muss der Cursor mit den Pfeiltasten auf diese Stelle positioniert und dann diese Stelle mit CR »angeklickt« werden.

Im Farbwahl-Fenster kann mit der Tastatur nicht »angeklickt« werden. Stattdessen ist das entsprechende Zeichenpaar mit der Tastatur einzugeben.

Zum Abschluss muss die Farbwahl mit der Enter-Taste bestätigt werden.

## Aufbereiten der Daten

Als »name« kann ein beliebiger, bis zu acht Zeichen langer Name verwendet werden, einzige Bedingung ist: Es darf keine Datei mit diesem Namen und auch keine Datei mit dem Namen name.xxx (xxx = beliebiger Namensteil) existieren.

Beispielaufrufe zur Datenaufbereitung:

```
#*SUCHE,name,,GENERIERE,TEXT=datei

#*SUCHE,name,,GENERIERE,TEXT=datei,FUSSNOTEN=+,REGISTER=*
o  Ortsnamen
p  Personennamen
-  sonstige Wörter
*EOF
```

Wenn zum Ausprobieren des Kommandos #SUCHE keine geeigneten Daten zur Verfügung stehen, kann der nachfolgende Beispieltext verwendet werden, indem das Makro wie folgt aufgerufen wird:

```
#*SUCHE,name,,GENERIERE,TEXT=+,FUSSNOTEN=+,REGISTER=+
```

Sollen alle Dateien, die mit dem Makro \*SUCHE eingerichtet wurden, wieder gelöscht werden, kann dazu folgender Aufruf verwendet werden:

```
#*SUCHE,name,,LOESCHE
```

## Beispieltext

1.1 ((|Titel|)) Beispieltext für das Kommando #SUCHE  
1.2  
1.3  
2.1 ((|Einleitung|)) <<\*1>>Einleitung  
2.2  
2.3 Dies ist ein einfacher Beispieltext. Er enthält  
2.4 sowohl Personennamen (z.B. <<7>>((p Müller, Peter)))  
2.5 als auch Ortsnamen (z.B. <<8>>((o Tübingen)) und  
2.6 <<9>>((o München))). Außerdem kommen <<3>>Fußnoten und  
2.7 <<4>>Verweise auf andere Textstellen vor. Alles  
2.8 klar?[[1]] Wenn nicht, helfen vielleicht die  
2.8/1 [[\*1]] Es kann doch nicht alles klar sein,  
2.8/2 denn es wurde ja noch fast nichts erklärt!  
2.9 folgenden sechs Kapitel weiter.  
2.10  
3.1 ((|Satznummern|)) Kapitel 1: <<\*2>>Satznummern  
3.2  
3.3 Die Sätze der Datei müssen so nummeriert sein, dass  
3.4 - alle Satznummern aufsteigend sind,  
3.5 - keine Satznummer mehrfach vorkommt.  
3.6  
3.7 Die Unterscheidungsnummer jeder Satznummer muss  
3.8 Null sein. Falls die Fußnoten jedoch nicht in  
3.9 einer eigenen Datei, sondern in derselben Datei  
3.10 wie der Text stehen, müssen die Unterscheidungs-  
3.11 nummern der Sätze, die Fußnoten enthalten,  
3.12 ungleich Null sein, damit Text und Fußnoten  
3.13 unterscheidbar sind.  
3.14  
4.1 ((|Stellenangaben|)) Kapitel 2: Stellenangaben  
4.2  
4.3 Mit den Daten können auch zwei verschiedene  
4.4 Arten von Stellenangaben angezeigt werden: die  
4.5 Seiten-Zeilen-Nummer und die Referenz. Die  
4.6 Seiten-Zeilen-Nummer wird aus der jeweiligen  
4.7 Satznummer genommen, als Referenz wird jeweils die  
4.8 im Text vorangehende Referenzangabe angezeigt.  
4.9  
4.10 Eine Referenzangabe muss im Text in "((|" und "|))"  
4.11 eingeschlossen werden; sie wird nicht als Text  
4.12 angezeigt. Hier im Beispieltext ist vor jeder  
4.13 Kapitelüberschrift eine entsprechende Referenz  
4.14 angegeben. So steht z. B. "((|Stellenangaben|))"  
4.15 oben vor "Kapitel 2" und "((|Register|))" unten  
4.16 vor "Kapitel 3".



- 4.17
- 5.1 ((|Register|)) Kapitel 3: Register
- 5.2
- 5.3 Sollen außer einem Wortformenregister auch noch
- 5.4 andere Register erstellt werden, so müssen die
- 5.5 entsprechenden Wörter in "(x " und ")")
- 5.6 eingeschlossen werden. Dabei ist für "x" jeweils
- 5.7 ein Buchstabe als Registerkennzeichen anzugeben.
- 5.8 Diese Kennzeichnung wird nicht mit angezeigt. Oben
- 5.9 in der <<1>>Einleitung steht z. B. in den Daten
- 5.10 "(p Müller, Peter)" damit "Müller, Peter" ins
- 5.11 Personenregister kommt, und "(o München) und
- 5.12 ((o Tübingen))", damit sowohl "München" als auch
- 5.13 "Tübingen" ins Ortsregister kommen.
- 5.14
- 5.15 Sollen Wörter nicht unverändert ins Register
- 5.16 übernommen werden, so kann dies mit der
- 5.17 Kennzeichnung "(textform ==x registerform)"
- 5.18 erreicht werden. Dabei wird "textform" im Text
- 5.19 angezeigt, "x" ist das Registerkennzeichen und
- 5.20 "Registerform" kommt in das angegebene Register.
- 5.21
- 5.22 Unten im <<6>>Schlusswort steht in den Daten z. B.
- 5.23 "(Münchner ==o München)" damit im Text
- 5.24 "Münchner" angezeigt wird, im Ortsregister aber
- 5.25 "München" eingetragen wird und weiter unten
- 5.26 "(Peter Müller ==o Müller, Peter)", damit
- 5.27 "Peter Müller" im Personenregister unter
- 5.28 "Müller, Peter" zu finden ist. Am Ende des
- 5.29 gleichen Absatzes steht als weiteres Beispiel
- 5.30 "(Sonn- ==- Sonntagen)"; dort ist hinter den
- 5.31 beiden Gleichheitszeichen ein "-" als
- 5.32 Registerkennzeichen angegeben, weil das Wort
- 5.33 "Sonntagen" ins Wortformenregister soll.
- 5.34
- 5.35 Sollen Wörter in verschiedenen Formen und/oder in
- 5.36 verschiedene Register übernommen werden, so kann
- 5.37 kann dies in der folgenden Form angegeben werden:
- 5.38 "(textform ==x form1 ==x form2 ==y form3)".
- 3.39
- 6.1 ((|Fußnoten|)) Kapitel 4: <<\*3>>Fußnoten
- 6.2
- 6.3 Oben in der <<1>>Einleitung steht hinter "Alles
- 6.4 klar?" ein Verweis auf eine Fußnote. Wird der
- 6.5 Verweis aktiviert, wird der Text der Fußnote in
- 6.6 einem Fußnoten-Fenster angezeigt.
- 6.7
- 6.8 Ein Fußnotenverweis ist in den Daten eine in zwei

- 6.9 eckigen Klammern eingeschlossene Fußnotennummer  
6.10 (hier [[1]]). Die dazugehörige Fußnote muss  
6.11 in gleicher Weise gekennzeichnet sein, jedoch mit  
6.12 einem zusätzlichen Stern unmittelbar vor der  
6.13 Fußnotennummer ([[\*1]]). Jede Fußnotennummer  
6.14 darf nur einmal vergeben werden; es dürfen aber  
6.15 mehrere Verweise auf die gleiche Fußnote vorkommen.  
6.16  
6.17 Fußnoten können in einer eigenen Datei oder in  
6.18 derselben Datei wie der Text stehen. Falls sie in  
6.19 derselben Datei stehen, muss auf die Sonderregelung  
6.20 für die <<2>>Satznummern geachtet werden.  
6.21  
7.1 ((|Interne Verweise|)) Kapitel 5: <<\*4>>Interne Verweise  
7.2  
7.3 Ein interner Verweis ist ein Verweis auf eine  
7.4 Textstelle innerhalb derselben Datei.  
7.5  
7.6 Ein interner Verweis steht oben in der  
7.7 <<1>>Einleitung vor "(p Müller, Peter)". Er wird als  
7.8 einfacher Pfeil (-->) angezeigt. Wird der Verweis  
7.9 aktiviert, wird an die Stelle im Text gesprungen,  
7.10 auf die der Verweis zeigt, und der Text dieser  
7.11 Stelle (hier das Schlusswort) im Volltext-Fenster  
7.12 angezeigt.  
7.13  
7.14 In den Daten steht an Stelle des Pfeils eine Zahl  
7.15 in doppelten spitzen Klammern eingeschlossen  
7.16 (hier <<7>>). Die Zahl gibt die Nummer der  
7.17 dazugehörigen Textstelle an. Diese muss in  
7.18 gleicher Weise gekennzeichnet sein, jedoch mit  
7.19 einem zusätzlichen Stern unmittelbar vor der  
7.20 Nummer (<<\*7>>). Jede interne Textstellenummer  
7.21 darf nur einmal vergeben werden; es dürfen aber  
7.22 mehrere Verweise auf die gleiche Textstelle  
7.23 vorkommen.  
7.24  
8.1 ((|Externe Verweise|)) Kapitel 6: <<\*5>>Externe Verweise  
8.2  
8.3 Ein externer Verweis ist ein Verweis auf eine  
8.4 Textstelle in einer anderen Datei.  
8.5  
8.6 Ein externer Verweis kommt in diesem Beispieltext  
8.7 nicht vor. Er würde als doppelter Pfeil (==>)  
8.8 angezeigt. Wird ein solcher Verweis aktiviert,  
8.9 wird der entsprechende Text in einem  
8.10 Zusatz-Fenster angezeigt.  
8.11

8.12 In den Daten müsste an Stelle des Pfeils eine Zahl  
8.13 in doppelten geschweiften Klammern stehen. Die  
8.14 Zahl gibt die Nummer der dazugehörenden Textstelle  
8.15 (in der anderen Datei) an. Diese muss in gleicher  
8.16 Weise gekennzeichnet sein, jedoch mit einem  
8.17 zusätzlichen Stern unmittelbar vor der Nummer.  
8.18 Jede externe Textstellenummer darf nur einmal  
8.19 vergeben werden; es dürfen aber mehrere Verweise  
8.20 auf die gleiche Textstelle vorkommen.  
8.21  
9.1 ((|Schlusswort|)) <<\*6>>Schlusswort  
9.2  
9.3 Wenn Sie nicht mehr weiter wissen, können Sie am  
9.4 ((Münchner ==o München)) Bahnhof nachfragen. Dort wohnt  
9.5 ein Mann namens <<\*7>>((Peter Müller ==o Müller, Peter)).  
9.6 Er hat keine Ahnung von TUSTEP, aber er kennt jemanden,  
9.7 der vor kurzem von <<\*8>>((o Tübingen)) nach  
9.8 <<\*9>>((o München)) umgezogen ist und fast alle Fragen  
9.9 zu TUSTEP beantworten kann. Er macht es immer gerne, nur  
9.10 nicht an ((Sonn- ==- Sonntagen)) und Feiertagen.



**{}-Parameter**

---

Allgemeines . . . . .	703
Codierungen mit Sonderfunktion . . . . .	704
Format der Parameter . . . . .	706
Fortsetzungszeilen bei Parametern . . . . .	706
Regelung für Groß- und Kleinbuchstaben . . . . .	707
I. Zahlenwerte . . . . .	707
II. Textteile . . . . .	707
III. Textteile-Vergleichs-Tabelle . . . . .	708
IV. Textteile-Austausch-Tabelle . . . . .	708
V. Zeichengruppen und Stringgruppen . . . . .	709
Vordefinierte Zeichengruppen . . . . .	709
Definition von Zeichengruppen . . . . .	710
Definition von Stringgruppen . . . . .	710
VI. Zeichen-Tabelle . . . . .	712
VII. Sortieralphabet-Tabelle . . . . .	712
VIII. Vergleichs-Tabelle . . . . .	713
Besonderheiten der Parameterart VIII-a . . . . .	714
Besonderheiten der Parameterart VIII-b . . . . .	714
IX. Such-Tabelle . . . . .	714
Zeichengruppen und Stringgruppen . . . . .	716
Zahlenwertbedingungen . . . . .	716
Häufigkeitsbedingungen . . . . .	717
Verweise (in Suchzeichenfolgen) . . . . .	719
Umgebungsbedingungen . . . . .	720
X. Austausch-Tabelle . . . . .	721
Elementbereiche (in Suchzeichenfolgen) . . . . .	722
Verweise (in Ersatzzeichenfolgen) . . . . .	722
XI. Sonstiges . . . . .	724
XII. Recherchier-Tabelle . . . . .	724
Besonderheiten der Parameterart XII-a . . . . .	726
Besonderheiten der Parameterart XII-b . . . . .	726
Auswählen und Eliminieren von Textteilen . . . . .	728

## Allgemeines

Mit Parametern werden die von den TUSTEP-Programmen gewünschten Leistungen genauer beschrieben. Ihre Funktion ist den einzelnen Programmbeschreibungen zu entnehmen.

Für die Parameter gibt es drei verschiedenen Konventionen. Nach welcher Konvention die Parameter interpretiert werden, kann mit dem Kommando `#PARAMETER,MODUS=...` (siehe Seite 207) eingestellt werden. Außerdem kann sie innerhalb von Parametern mit dem Parameter `PAR` festgelegt werden.

Die »alte« Konvention ist die ursprüngliche in TUSTEP. Sie wurde konzipiert, als die Lochkarte noch das Standardmedium für die Daten- und Programmeingabe war. Dabei musste berücksichtigt werden, dass auf Lochkarten üblicherweise Groß- und Kleinbuchstaben nicht unterschieden wurden und Umlaute nicht vorgesehen waren (`MODUS=ALT`).

Die »neue« Konvention ist auf neuere Eingabemedien abgestimmt, bei denen diese Einschränkungen nicht mehr bestehen. Zur Kennzeichnung von Zeichen- und Stringgruppen, Häufigkeitsangaben, Verweisen usw. werden dabei die spitzen Klammern verwendet (`MODUS=NEU`, gleichbedeutend mit `MODUS=<>`).

Die Mustererkennung (pattern matching) wurden nach und nach erweitert und verbessert. Dabei wurde bei den neu hinzukommenden Codierungen immer darauf geachtet, dass auch zuvor erstellte Parameter noch die gleiche Wirkung wie seither hatten. Das hat dazu geführt, dass die Codierungen unübersichtlich wurden und nicht mehr leicht zu merken und auch nicht mehr leicht zu lesen sind. Darüber hinaus ist die direkte Verwendung von spitzen Klammern nicht möglich, wenn Parameter mit einem XML-Editor geschrieben oder geändert werden sollen. Deshalb wurden die Angaben zur Mustererkennung neu konzipiert und damit eine dritte Konvention für die Parameter geschaffen, bei der die Codierungen leichter zu merken und auch leichter zu lesen sind. Zur Kennzeichnung von Zeichen- und Stringgruppen, Häufigkeitsangaben, Verweisen usw. werden dabei die geschweiften Klammern verwendet (`MODUS={ }`).

Für neue Projekte empfiehlt es sich, diese dritte Konvention zu verwenden. Für den Editor und die Programme mit Parametern ist diese dritte Konvention voreingestellt.

Um bei alten Projekten Aufwärtskompatibilität für Makros zu gewährleisten, ist für Makros (unabhängig von der Einstellung mit dem Kommando `#PARAMETER`) die zweite Konvention (`MODUS=<>`) voreingestellt. Mit der Makroanweisung `MODE` (siehe ab Seite 413) kann die Konvention, nach der Such- und Austausch-Tabellen interpretiert werden, explizit eingestellt werden.

Zur Erleichterung der Eingabe von Zeichen- und Stringgruppen, Häufigkeitsangaben, Verweisen usw. in Parametern nach der »dritten« Konvention gibt es im Editor eine Eingabehilfe (siehe Seite 269).

In diesem Kapitel sind die Parameter nach der »dritten« Konvention beschrieben. Nach der »alten« und nach der »neuen« Konvention sind sie im Kapitel »<>-Parameter« ab Seite 735 beschrieben.

## Codierungen mit Sonderfunktion

Bei einigen Parameterarten haben Fragezeichen, Stern, eckige und geschweifte Klammern eine Sonderfunktion und werden zur Codierung von Häufigkeitsangaben, Verweisen, Zahlenwerten sowie für Stellvertreter von Zeichen- und Stringgruppen verwendet.

Die folgende Liste zeigt eine Übersicht dieser Codierungen. Diese Liste kann im Editor mit der Tastenkombination `Ctrl+K-Leerzeichen` bzw. `Strg+K-Leerzeichen` angezeigt werden. Bei welchen Parameterarten diese Codierungen gelten, ist in der Beschreibung der jeweiligen Parameterarten angegeben.

### Einzelzeichen

<code>\?</code>	Fragezeichen
<code>\*</code>	Stern
<code>\[</code>	eckige Klammer auf
<code>\]</code>	eckige Klammer zu
<code>\{</code>	geschweifte Klammer auf
<code>\}</code>	geschweifte Klammer zu
<code>\a</code>	Kleinbuchstabe a
<code>\A</code>	Großbuchstabe A
<code>\\</code>	Backslash

### Zeichen- und Stringgruppen

<code>?</code>	ein beliebiges TUSTEP-Zeichen
<code>*</code>	null bis beliebig viele beliebige TUSTEP-Zeichen
<code>[ . . . ]</code>	lokale Zeichengruppe, z. B. <code>m[ae][iy]er</code>
<code>{Z:xy}</code>	Zeichengruppe xy
<code>{C:xy}</code>	Alternative Schreibweise zu <code>{Z:xy}</code>
<code>{S:xy}</code>	Stringgruppe xy
<code>{-}</code>	nachfolgende Zeichen aus der Zeichengruppe entfernen
<code>{+}</code>	nachfolgende Zeichen in die Zeichengruppe aufnehmen

### Vordefinierte Zeichengruppen

<code>{!}</code>	ASCII-Zeichen
<code>{;}</code>	TUSTEP-Zeichen außer ASCII-Zeichen
<code>{@}</code>	TUSTEP-Zeichen außer Buchstaben und Ziffern
<code>{%}</code>	Zeichen hinter % zur Akzent-Codierung
<code>{\a}</code>	Kleinbuchstaben
<code>{\A}</code>	Großbuchstaben
<code>{&amp;a}</code>	Kleinbuchstaben & Großbuchstaben
<code>{\0}</code>	normale Ziffern
<code>{&amp;0}</code>	normale Ziffern & hochgestellte Ziffern



## Häufigkeitsbedingungen in Suchzeichenfolgen

{n}	genau n Elemente
{n-m}	n bis m Elemente, möglichst wenige
{n--m}	n bis m Elemente, möglichst viele
{n-0}	n bis beliebig viele Elemente, möglichst wenige
{n--0}	n bis beliebig viele Elemente, möglichst viele
{0}	0 oder 1 Element, gleichbedeutend mit {0-1}
{00}	1 bis unendlich viele Elemente, gleichbed. mit {1-0}

## Zahlenwertbedingungen in Suchzeichenfolgen

{#}	Zahl mit beliebigem Wert
{#n}	Zahl mit dem Wert n
{!n}	Zahl mit einem Wert ungleich n
{#n-m}	Zahl mit einem Wert von n bis m
{#n-0}	Zahl mit einem Wert gleich oder größer n
{!n-m}	Zahl mit einem Wert kleiner n oder größer m
{!n-0}	Zahl mit einem Wert kleiner n

## Verweise in Suchzeichenfolgen

{+n=}	n-tes Element von links gezählt, a != A
{-n=}	n-tes Element von rechts gezählt, a != A
{+n:}	n-tes Element von links gezählt, a == A
{-n:}	n-tes Element von rechts gezählt, a == A

## Verweise in Ersatzzeichenfolgen

{+n=}	n-tes Element von links gezählt
{-n=}	n-tes Element von rechts gezählt
{=n=}	alle Elemente des n-ten Elementbereichs
{=0=}	alle Elemente der Kernzeichenfolge
{+n-m=}	n-tes bis m-tes Element von links gezählt
{+n-0=}	n-tes bis letztes Element von links gezählt
{-n-m=}	n-tes bis m-tes Element von rechts gezählt
{-0-m=}	erstes Element bis m-tes Element von rechts gezählt
{=n-m=}	alle Elemente des n-tes bis m-ten Elementbereichs
{...+}	... Kleinbuchstaben → Großbuchstaben
{...-}	... Großbuchstaben → Kleinbuchstaben
{...;}	... a, b, ..., \$, ... → ä, ^b, ..., ^\$, ...
{...!}	... ä, ^b, ..., ^\$, ... → a, b, ..., \$, ...

## Sonstiges in Suchzeichenfolgen

{[}	Wechsel vom linken Rand zur Kernzeichenfolge
{]}	Wechsel von der Kernzeichenfolge zum rechten Rand
{ }	Wechsel von einem Elementbereich in den nächsten

## Sonstiges in Sortieralphabeten

{ }	Wechsel von den niederen zu den hohen Sortierwerten
-----	---

## Format der Parameter

Spalte 1–3: Parameterkennung aus 1 bis 3 Zeichen, linksbündig. Parameter, die in den Spalten 1–3 Leerzeichen enthalten, gelten als Kommentar. Sie können an jeder beliebigen Stelle vorkommen.

4–5: leer: Parameter wird ausgewertet

»+n«: Parameter wird nur ausgewertet, wenn der Wahlschalter n (n = 1 bis 7) gesetzt ist

» n«: gleichbedeutend mit »+n«

»-n«: Parameter wird nur ausgewertet, wenn der Wahlschalter n (n = 1 bis 7) nicht gesetzt ist.

6–7: leer, oder rechtsbündig eine Zahl (vom jeweiligen Programm abhängig)

8: leer

9: leer Normalfall

»:« wenn das Informationsfeld in Spalte 11 mit Leerzeichen (z. B. bei Parameterart VII) beginnt

»=« wenn im Informationsfeld auf einen vorhergehenden Parameter gleicher Art verwiesen wird, dessen Informationsfeld übernommen werden soll. Spalte 11–13 muss die Parameterkennung und ggf. Spalte 16–17 die in Spalte 6–7 angegebene Nummer des Parameters, auf den verwiesen wird, enthalten.

10: leer

ab 11: Informationsfeld. Abschließende Leerzeichen werden ignoriert.

## Fortsetzungszeilen bei Parametern

Die Informationen für einen Parameter können auf mehrere Zeilen aufgeteilt werden. Bei allen Zeilen eines Parameters müssen die Spalten 1 bis 3 und 6 bis 7 identisch sein.

Bei Parametern mit Begrenzungszeichen muss in Fortsetzungszeilen das gleiche Begrenzungszeichen verwendet werden. Ein Begrenzungszeichen in Spalte 11 einer Fortsetzungszeile und ein Begrenzungszeichen am Ende der vorhergehenden Zeile gelten zusammen als nur ein Begrenzungszeichen.

Der besseren Lesbarkeit wegen empfiehlt es sich, diese Eigenschaft auszunutzen und die Zeileneinteilung so zu wählen, dass jedes Informationsfeld mit einem Begrenzungszeichen anfängt (Spalte 11) und mit einem Begrenzungszeichen endet (letzte verwendete Spalte).

Wird eine Zeichenfolge in der nächsten Zeile fortgesetzt, so darf jedoch weder am Ende der Zeile noch in Spalte 11 der nächsten Zeile ein Begrenzungszeichen stehen. In diesem Fall ist zu beachten, dass Leerzeichen am Zeilenende ignoriert werden.

## Regelung für Groß- und Kleinbuchstaben

a) bei den Parameterarten II bis IV:

Groß- und Kleinbuchstaben werden als solche interpretiert.

b) bei den Parameterarten V bis IX und XII:

Sind im Informationsfeld Buchstaben angegeben, so ist jeweils der entsprechende Groß- und Kleinbuchstabe gemeint (gleichgültig, ob der Buchstabe groß oder klein geschrieben ist). Soll zwischen Groß- und Kleinbuchstaben unterschieden werden, so ist der jeweilige Buchstabe mit einem vorangesetzten »\« zu kennzeichnen. Es bedeutet also »a« und »A« ein großes oder kleines a, »\a« ein kleines a und »\A« ein großes a. Soll das Zeichen »\« selbst dargestellt werden, so ist dafür »\\« zu schreiben.

c) bei der Parameterart X:

Für Suchzeichenfolgen gilt das Gleiche wie für Parameterart IX.

In Ersatzzeichenfolgen werden Groß- und Kleinbuchstaben als solche interpretiert, gleichgültig, ob sie mit einem vorangesetzten »\« gekennzeichnet sind oder nicht. Soll das Zeichen »\« selbst dargestellt werden, so ist dafür »\\« zu schreiben.

## Parameterart I: Zahlenwerte

Im Informationsfeld werden in arabischen Ziffern geschriebene Zahlenwerte angegeben. Unmittelbar hintereinander stehende Ziffern werden als eine Zahl interpretiert. Die einzelnen Zahlen können durch beliebige Zeichen (außer Ziffern) voneinander getrennt werden. Für die Zahlenwerte muss die in der jeweiligen Beschreibung angegebene Reihenfolge eingehalten werden; dabei darf kein Zahlenwert ausgelassen werden. Sind weniger Zahlen angegeben als vom jeweiligen Programm erwartet werden, so werden für die fehlenden Zahlen die vom jeweiligen Programm voreingestellten Zahlenwerte ergänzt.

Reicht für die Angaben eine Zeile nicht aus, so können Fortsetzungszeilen (mit der gleichen Parameterkennung) verwendet werden. An jedem Zeilenende wird ein Leerzeichen ergänzt.

Bei dieser Parameterart darf in Spalte 9 nur ein Leerzeichen stehen, d. h. es kann nicht auf einen vorhergehenden Parameter verwiesen werden.

## Parameterart II: Textteile

Im Informationsfeld werden Textteile angegeben, die durch ein frei wählbares Begrenzungszeichen voneinander getrennt sind. Das Begrenzungszeichen ist das erste Zeichen (Spalte 11) im Informationsfeld des Parameters. Der letzte Textteil muss durch ein Begrenzungszeichen abgeschlossen sein.

Sind weniger Textteile angegeben als vom jeweiligen Programm erwartet werden, so werden vom jeweiligen Programm voreingestellte Textteile ergänzt.

Reicht für die Angaben eine Zeile nicht aus, so können Fortsetzungszeilen (mit der gleichen Parameterkennung und dem gleichen Begrenzungszeichen; vgl. Seite 706) verwendet werden. Wird dabei ein Textteil in der nächsten Zeile fortgesetzt, so kann ein »-« als Silbentrennzeichen verwendet werden, falls es nicht als Begrenzungszeichen gewählt wurde. Ein »-« am Zeilenende gilt dann jedoch nur als Silbentrennzeichen, wenn das zweitletzte Zeichen ebenfalls ein »-« ist, oder wenn das zweitletzte Zeichen ein Buchstabe und das drittletzte Zeichen kein Steuerzeichen (\$, &, @, \, \_, #, %) ist.

### Parameterart III: Textteile-Vergleichs-Tabelle

Im Informationsfeld werden Textteile angegeben, die durch ein frei wählbares Begrenzungszeichen voneinander getrennt sind. Das Begrenzungszeichen ist das erste Zeichen (Spalte 11) im Informationsfeld des Parameters. Der letzte Textteil muss durch ein Begrenzungszeichen abgeschlossen sein.

Ob Groß- und Kleinschreibung für das Programm relevant ist, ist bei der jeweiligen Parameterbeschreibung angegeben; die Angabe von Zeichen- oder Stringgruppen (s. u.) ist nicht möglich.

Reicht für die Angaben eine Zeile nicht aus, so können Fortsetzungszeilen (mit der gleichen Parameterkennung und dem gleichen Begrenzungszeichen; vgl. Seite 706) verwendet werden.

### Parameterart IV: Textteile-Austausch-Tabelle

Im Informationsfeld werden Paare von Textteilen angegeben, deren jeweils erster Textteil zum Vergleich mit dem zu verarbeitenden Textteil dient, der bei Übereinstimmung durch den jeweils zweiten Textteil ersetzt werden soll. Vergleichs- und Ersatz-Textteil dürfen verschieden lang sein.

Die im Informationsfeld angegebenen Textteile werden durch ein frei wählbares Begrenzungszeichen voneinander getrennt. Das Begrenzungszeichen ist das erste Zeichen (Spalte 11) im Informationsfeld des Parameters. Der letzte Textteil muss durch ein Begrenzungszeichen abgeschlossen sein.

Ob in den Vergleichs-Textteilen Groß- und Kleinschreibung für das Programm relevant ist, ist bei der jeweiligen Parameterbeschreibung angegeben; die Angabe von Zeichen- oder Stringgruppen (s. u.) ist nicht möglich.

Reicht für die Angaben eine Zeile nicht aus, so können Fortsetzungszeilen (mit der gleichen Parameterkennung und dem gleichen Begrenzungszeichen; vgl. Seite 706) verwendet werden.

## Parameterart V: Zeichengruppen und Stringgruppen

(Gilt für »{}-Konvention«, für »<>-Konvention« siehe Seite 741.)

Eine Zeichengruppe ist eine Zusammenfassung von einzelnen Zeichen, auf die (nachdem sie definiert ist) im selben Programm in nachfolgenden Parametern der Parameterarten VI bis X und XII (in X nur in Suchzeichenfolgen) durch Angabe der dazugehörigen Gruppenkennung Bezug genommen werden kann. Die Zeichen der angegebenen Zeichengruppe werden dann so behandelt, als stünden sie alle gleichwertig an Stelle der Gruppenkennung.

Eine Stringgruppe ist eine Zusammenfassung von Zeichenfolgen (Strings), auf die (nachdem sie definiert ist) im selben Programm in nachfolgenden Parametern der Parameterarten IX und X (in X nur in Suchzeichenfolgen) durch Angabe der dazugehörigen Gruppenkennung Bezug genommen werden kann. Die Zeichenfolgen der angegebenen Stringgruppe werden dann so behandelt, als stünden sie alle an Stelle der Gruppenkennung.

Zur Definition von Zeichen- und Stringgruppen und zur Bezugnahme auf die so definierten Gruppen stehen für Zeichengruppen zwei gleichwertige Gruppenkennungen der Form »{z:xy}« und »{c:xy}« und für Stringgruppen eine Gruppenkennung der Form »{s:xy}« (jeweils ohne Anführungszeichen) zur Verfügung. Dabei ist »xy« ein aus zwei Zeichen bestehender Name, wobei »x« ein Buchstabe und »y« eine Buchstabe oder eine Ziffer sein muss. Groß- und Kleinschreibung wird nicht unterschieden.

Die Definition einer Gruppe gilt jeweils für alle nachfolgenden Parameter eines Kommandos, bis diese Gruppe neu definiert wird.

Bei dieser Parameterart darf in Spalte 9 nur ein Leerzeichen oder ein Doppelpunkt stehen, d. h. es kann nicht auf einen vorhergehenden Parameter verwiesen werden.

### Vordefinierte Zeichengruppen

Außer Zeichengruppen, die selbst definiert werden können, gibt es intern vordefinierte Zeichengruppen mit folgenden Zeichengruppen-Kennungen:

- {\a} alle Kleinbuchstaben des TUSTEP-Zeichensatzes
- {\A} alle Großbuchstaben des TUSTEP-Zeichensatzes
- {\0} alle Ziffern des ASCII-Zeichensatzes
- {&0} alle Ziffern des TUSTEP-Zeichensatzes
- {&a} alle Buchstaben des TUSTEP-Zeichensatzes
- {!} alle Zeichen des ASCII-Zeichensatzes
- {;} alle Zeichen des TUSTEP-Zeichensatzes, die nicht im ASCII-Zeichensatz enthalten sind
- {%} alle Zeichen, die zur Akzent-Codierung nach % stehen können
- {@} alle Sonderzeichen des TUSTEP-Zeichensatzes einschließlich Leerzeichen

## Definition von Zeichengruppen

Als Parameterkennung für die Definition einer Zeichengruppe ist statt der Gruppenkennungen »{z:xy}« bzw. »{c:xy}« in Spalte 1 bis 3 »>xy« zu schreiben.

Im Informationsfeld werden die Zeichen (ab Spalte 11, ohne Zwischenraum) angegeben, die zu der zu definierenden Gruppe gehören sollen.

Achtung: Das Fragezeichen hat bei dieser Parameterart eine spezielle Bedeutung. Es steht stellvertretend für ein beliebiges Zeichen. Ist das Zeichen »?« selbst gemeint, so ist dafür »\?« zu schreiben.

Außerdem dienen die geschweiften Klammern zur Kennzeichnung von Zeichengruppen. Deshalb müssen auch diese Klammern und der Backslash mit einem vorangestellten Backslash (z. B. »\\«) gekennzeichnet werden, wenn diese Zeichen selbst gemeint sind.

An Stelle eines Zeichens kann auch eine Kennung einer vordefinierten Zeichengruppe angegeben werden. Innerhalb von Parameterangaben (aber nicht in Editoranweisungen) kann an Stelle eines Zeichens auch eine Kennung einer selbst definierten Zeichengruppe angegeben werden.

Sind im Informationsfeld Buchstaben angegeben, so ist jeweils der entsprechende Groß- und Kleinbuchstabe gemeint (gleichgültig, ob der Buchstabe groß oder klein geschrieben ist). Soll zwischen Groß- und Kleinbuchstaben unterschieden werden, so ist der jeweilige Buchstabe mit einem vorangesetzten »\« zu kennzeichnen. Es bedeutet also »a« und »A« ein großes oder kleines a, »\a« ein kleines a, »\A« ein großes a.

Das Informationsfeld wird von links nach rechts abgearbeitet. Dabei werden die Zeichen in die (zu Anfang leere) Gruppe eingetragen. Die Zeichenfolge »{-}« bewirkt, dass die nachfolgenden Zeichen wieder aus der Gruppe gelöscht werden. Mit der Zeichenfolge »{+}« kann die Wirkung von »{-}« wieder aufgehoben werden, d. h. die nachfolgenden Zeichen werden wieder zur Gruppe hinzugefügt.

## Definition von Stringgruppen

Als Parameterkennung für die Definition einer Stringgruppe ist statt der Gruppenkennung »{s:xy}« in Spalte 1 bis 3 »<xy« zu schreiben.

Im Informationsfeld werden die Zeichenfolgen angegeben, die zu der zu definierenden Stringgruppe gehören sollen. Sie werden durch ein frei wählbares Begrenzungszeichen (außer Backslash und Klammern) voneinander getrennt. Das Begrenzungszeichen ist das erste Zeichen (Spalte 11) im Informationsfeld des Parameters. Die letzte Zeichenfolge muss durch ein Begrenzungszeichen abgeschlossen sein.

Achtung: Das Fragezeichen und der Stern haben bei dieser Parameterart eine spezielle Bedeutung. Ein Fragezeichen steht stellvertretend für ein beliebiges TUSTEP-Zeichen und ein Stern für null bis beliebig viele beliebige TUSTEP-Zeichen. Sind die Zeichen »?« und »\*« selbst gemeint, so ist dafür »\?« bzw. »\\*« zu schreiben.

Außerdem dienen die geschweiften Klammern zur Kennzeichnung von Zeichengruppen. Deshalb müssen auch diese Klammern und der Backslash mit einem vor-

angestellten Backslash (z. B. »\«) gekennzeichnet werden, wenn diese Zeichen selbst gemeint sind.

An Stelle eines einzelnen Zeichens einer Zeichenfolge kann die Kennung einer Zeichengruppe angegeben werden; Kennungen von Stringgruppen und in eckigen Klammern angegebene Zeichengruppen sind hier nicht vorgesehen.

Sind im Informationsfeld Buchstaben angegeben, so ist jeweils der entsprechende Groß- und Kleinbuchstabe gemeint (gleichgültig, ob der Buchstabe groß oder klein geschrieben ist). Soll zwischen Groß- und Kleinbuchstaben unterschieden werden, so ist der jeweilige Buchstabe mit einem vorangesetzten »\« zu kennzeichnen. Es bedeutet also »a« und »A« ein großes oder kleines a, »\a« ein kleines a, »\A« ein großes a.

Es können zwei Arten von Zeichenfolgen angegeben werden, nämlich »Suchzeichenfolgen« und »Ausnahmezeichenfolgen«. Sie werden durch zwei aufeinander folgende Begrenzungszeichen voneinander getrennt. Durch Angabe von zwei aufeinander folgenden Begrenzungszeichen kann beliebig oft zwischen Suchzeichenfolgen und Ausnahmezeichenfolgen gewechselt werden.

Entspricht eine im Text ab der jeweils beim Suchen erreichten Position stehende Zeichenfolge mehreren Such- und/oder Ausnahmezeichenfolgen, so hat die jeweils längere Zeichenfolge den Vorrang; bei gleich langen Zeichenfolgen entspricht ihre Rangfolge der Reihenfolge ihrer Angabe. Wird eine Ausnahmezeichenfolge gefunden, so hat dies die gleiche Wirkung, wie wenn keine der in der Stringgruppe enthaltenen Zeichenfolgen gefunden worden wäre.

Achtung: Auf Grund dieser Vorgehensweise sind in Stringgruppen nur solche Ausnahmezeichenfolgen sinnvoll, die im Text auf der gleichen Position beginnen können wie eine in der gleichen Stringgruppe angegebene Suchzeichenfolge (z. B. »/xy//xy!/«, aber nicht »/xy//!xy/«). Außerdem ist bei gleich langen Such- und Ausnahmezeichenfolgen auf die Reihenfolge zu achten. So ist z. B. die Angabe »/{\a}{\a}/xy/« nicht sinnvoll, weil die Zeichenfolgen »{\a}{\a}« und »xy« beide zwei Zeichen lang sind (»{\a}« gilt als ein Zeichen, weil es stellvertretend für einen Kleinbuchstaben steht), und bei der Überprüfung der im Text stehenden Zeichenfolge »xy« die Suchzeichenfolge »{\a}{\a}« zum Zuge kommt, weil sie vor der Ausnahmezeichenfolge »xy« angegeben ist. Damit in diesem Fall die Ausnahmezeichenfolge wirkt, muss »//xy//{\a}{\a}/« angegeben werden.

Reicht für die Angaben eine Zeile nicht aus, so können Fortsetzungszeilen (mit der gleichen Parameterkennung und dem gleichen Begrenzungszeichen; vgl. Seite 706) verwendet werden.

Wenn am Anfang eines Informationsfeldes eine Ausnahmezeichenfolge angegeben werden soll, kann mit zwei aufeinander folgenden Begrenzungszeichen am Anfang des Informationsfeldes von Suchzeichenfolge auf Ausnahmezeichenfolge umgeschaltet werden.

Um Fehler zu vermeiden, empfiehlt sich folgendes Verfahren:

Wenn am Ende einer Zeile eine Ausnahmezeichenfolge angegeben ist, sollte diese Zeile mit zwei aufeinander folgenden Begrenzungszeichen am Ende des Informati-

onsfeldes abgeschlossen werden und damit von Ausnahmezeichenfolgen auf Suchzeichenfolgen umgeschaltet werden.

## Parameterart VI: Zeichen-Tabelle

(Gilt für »{}-Konvention«, für »<>-Konvention« siehe Seite 744.)

Im Informationsfeld werden die Zeichen (ab Spalte 11, ohne Zwischenraum) angegeben, die in die Tabelle aufgenommen werden sollen. Wenn in der jeweiligen Beschreibung der einzelnen Parameter nichts anderes angegeben ist, ist die Reihenfolge der angegebenen Zeichen ohne Bedeutung.

Achtung: Das Fragezeichen hat bei dieser Parameterart eine spezielle Bedeutung. Es steht stellvertretend für ein beliebiges TUSTEP-Zeichen. Ist das Zeichen »?« selbst gemeint, so ist dafür »\?» zu schreiben.

Außerdem dienen die eckigen Klammern zur direkten Angabe von Zeichengruppen und die geschweiften Klammern zur Kennzeichnung von anderen Zeichengruppen. Deshalb müssen auch diese Klammern und der Backslash mit einem vorangestellten Backslash (z. B. »\«) gekennzeichnet werden, wenn diese Zeichen selbst gemeint sind.

Innerhalb von eckigen Klammern angegebene Zeichen werden jeweils als eine Zeichengruppe interpretiert, die diese Zeichen enthält. Es sind die gleichen Angaben wie bei der Definition von Zeichengruppen (siehe Parameterart V Seite 709 ff.) erlaubt.

An Stelle eines Zeichens kann auch die Kennung einer Zeichengruppe angegeben werden. Die Zeichen der angegebenen Zeichengruppe werden so behandelt, als stünden sie alle an Stelle der Zeichengruppen-Kennung.

Sind im Informationsfeld Buchstaben angegeben, so ist jeweils der entsprechende Groß- und Kleinbuchstabe gemeint (gleichgültig, ob der Buchstabe groß oder klein geschrieben ist). Soll zwischen Groß- und Kleinbuchstaben unterschieden werden, so ist der jeweilige Buchstabe mit einem vorangesetzten »\« zu kennzeichnen. Es bedeutet also »a« und »A« ein großes oder kleines a, »\a« ein kleines a, »\A« ein großes a.

## Parameterart VII: Sortieralphabet-Tabelle

(Gilt für »{}-Konvention«, für »<>-Konvention« siehe Seite 744.)

Das Informationsfeld wird wie bei einer Zeichen-Tabelle interpretiert, jedoch ist die Reihenfolge der einzelnen Zeichen immer von Bedeutung. Die Reihenfolge legt die Wertigkeit der Zeichen beim Sortieren bzw. beim Vergleichen fest; d. h. das erste Zeichen (bzw. alle Zeichen der auf erster Zeichenposition angegebenen Zeichengruppe) hat den niedrigsten Wert, dann folgt das zweite (bzw. die Zeichen der entsprechenden Gruppe) usw. Hinter der Zeichenkombination »{|}« können Zeichen (und Zeichengruppen) angegeben werden, die die höchsten Wertigkeiten erhalten sollen, also z. B. beim Sortieren ans Ende sortiert werden sollen. Werden nicht alle Zeichen des Zei-



chenvorrats angegeben, so werden die fehlenden Zeichen in der Reihenfolge der Standard-Sortierfolge (siehe Seite Seite 841) an der mit »{ | }« gekennzeichneten Stelle bzw. (falls »{ | }« fehlt) nach den angegebenen Zeichen logisch ergänzt.

## Parameterart VIII: Vergleichs-Tabelle

(Gilt für »{}-Konvention«, für »<>-Konvention« siehe Seite 744.)

Im Informationsfeld werden Zeichenfolgen angegeben, nach denen gesucht werden soll. Sie werden durch ein frei wählbares Begrenzungszeichen (außer Backslash und Klammern) voneinander getrennt. Das Begrenzungszeichen ist das erste Zeichen (Spalte 11) im Informationsfeld des Parameters. Die letzte Zeichenfolge muss durch ein Begrenzungszeichen abgeschlossen sein.

Achtung: Das Fragezeichen hat bei dieser Parameterart eine spezielle Bedeutung. Es steht stellvertretend für ein beliebiges TUSTEP-Zeichen. Ist das Zeichen »?« selbst gemeint, so ist dafür »\?« zu schreiben.

Außerdem dienen die eckigen Klammern zur direkten Angabe einer Zeichengruppe und die geschweiften Klammern zur Kennzeichnung von Verweisen auf eine Zeichengruppe. Deshalb müssen auch diese Klammern und der Backslash mit einem vorangestellten Backslash (z. B. »\«) gekennzeichnet werden, wenn diese Zeichen selbst gemeint sind.

Innerhalb von eckigen Klammern angegebene Zeichen werden jeweils als eine Zeichengruppe interpretiert, die diese Zeichen enthält. Es sind die gleichen Angaben wie bei der Definition von Zeichengruppen (siehe Parameterart V Seite 709 ff.) erlaubt.

An Stelle eines Zeichens kann auch die Kennung einer Zeichengruppe angegeben werden. Die Zeichen der angegebenen Zeichengruppe werden so behandelt, als stünden sie alle an Stelle der Zeichengruppen-Kennung.

Sind im Informationsfeld Buchstaben angegeben, so ist jeweils der entsprechende Groß- und Kleinbuchstabe gemeint (gleichgültig, ob der Buchstabe groß oder klein geschrieben ist). Soll zwischen Groß- und Kleinbuchstaben unterschieden werden, so ist der jeweilige Buchstabe mit einem vorangesetzten »\« zu kennzeichnen. Es bedeutet also »a« und »A« ein großes oder kleines a, »\a« ein kleines a, »\A« ein großes a.

Es können zwei Arten von Zeichenfolgen angegeben werden, nämlich »Suchzeichenfolgen« und »Ausnahmezeichenfolgen«. Sie werden durch zwei aufeinander folgende Begrenzungszeichen voneinander getrennt. Durch Angabe von zwei aufeinander folgenden Begrenzungszeichen kann beliebig oft zwischen Suchzeichenfolgen und Ausnahmezeichenfolgen gewechselt werden.

Achtung: Bei gleich langen Such- und Ausnahmezeichenfolgen ist auf die Reihenfolge zu achten. So ist z. B. die Angabe »/{\a}{\a} //xy/« nicht sinnvoll, weil die Zeichenfolgen »{\a}{\a}« und »xy« beide zwei Zeichen lang sind (»{\a}« gilt als ein Zeichen, weil es stellvertretend für einen Kleinbuchstaben steht), und bei der Überprüfung mit dem Textanfang »xy« die Suchzeichenfolge »{\a}{\a}« zum Zuge

kommt, weil sie vor der Ausnahmezeichenfolge »xy« angegeben ist. Damit in diesem Fall die Ausnahmezeichenfolge wirkt, muss »//xy//{\a}{\a}/« angegeben werden.

Reicht für die Angaben eine Zeile nicht aus, so können Fortsetzungszeilen (mit der gleichen Parameterkennung und dem gleichen Begrenzungszeichen; vgl. Seite 706) verwendet werden.

### Besonderheiten der Parameterart VIII-a

Die Texte werden nur daraufhin überprüft, ob sie mit einer der angegebenen Zeichenfolgen beginnen. Entspricht dabei der Textanfang mehreren Such- und/oder Ausnahmezeichenfolgen, so hat die jeweils längere Zeichenfolge den Vorrang; bei gleich langen Zeichenfolgen entspricht ihre Rangfolge der Reihenfolge ihrer Angabe. Wird (unter Berücksichtigung der genannten Rangfolge) eine Übereinstimmung mit einer Ausnahmezeichenfolge festgestellt, so hat dies die gleiche Wirkung, wie wenn keine der angegebenen Zeichenfolgen mit dem Textanfang übereinstimmen würde.

### Besonderheiten der Parameterart VIII-b

Die Texte werden nur daraufhin überprüft, ob sie mit einer der angegebenen Zeichenfolgen enden. Entspricht dabei das Textende mehreren Such- und/oder Ausnahmezeichenfolgen, so hat die jeweils längere Zeichenfolge den Vorrang; bei gleich langen Zeichenfolgen entspricht ihre Rangfolge der Reihenfolge ihrer Angabe. Wird (unter Berücksichtigung der genannten Rangfolge) eine Übereinstimmung mit einer Ausnahmezeichenfolge festgestellt, so hat dies die gleiche Wirkung, wie wenn keine der angegebenen Zeichenfolgen mit dem Textende übereinstimmen würde.

## Parameterart IX: Such-Tabelle

*(Gilt für »{}-Konvention«, für »<>-Konvention« siehe Seite 746.)*

Im Informationsfeld werden Zeichenfolgen angegeben, nach denen gesucht werden soll. Sie werden durch ein frei wählbares Begrenzungszeichen (außer Backslash und Klammern) voneinander getrennt. Das Begrenzungszeichen ist das erste Zeichen (Spalte 11) im Informationsfeld des Parameters. Die letzte Zeichenfolge muss durch ein Begrenzungszeichen abgeschlossen sein.

Achtung: Das Fragezeichen und der Stern haben bei dieser Parameterart eine spezielle Bedeutung. Ein Fragezeichen steht stellvertretend für ein beliebiges TUSTEP-Zeichen und ein Stern für null bis beliebig viele beliebige TUSTEP-Zeichen. Sind die Zeichen »?« und »\*« selbst gemeint, so ist dafür »\?« bzw. »\\*« zu schreiben.

Außerdem dienen die eckigen Klammern zur direkten Angabe einer Zeichengruppe und die geschweiften Klammern zur Kennzeichnung von Verweisen und Bedingungen. Deshalb müssen auch diese Klammern und der Backslash mit einem vorange-

stellten Backslash (z. B. »\\«) gekennzeichnet werden, wenn diese Zeichen selbst gemeint sind.

Innerhalb von eckigen Klammern angegebene Zeichen werden jeweils als eine Zeichengruppe interpretiert, die diese Zeichen enthält. Es sind die gleichen Angaben wie bei der Definition von Zeichengruppen (siehe Parameterart V Seite 709 ff.) erlaubt.

An Stelle eines einzelnen Zeichens einer Zeichenfolge kann eine in eckige Klammern eingeschlossene Zeichengruppe, die Kennung einer Zeichengruppe oder Stringgruppe, eine Zahlenwertbedingung oder ein Verweis auf andere Elemente (s. u.) der gleichen Zeichenfolge angegeben werden. Darüber hinaus können Zeichen, Zeichengruppen, Stringgruppen, Zahlenwertbedingungen sowie Verweise mit Häufigkeitsbedingungen versehen werden und können Umgebungsbedingungen für Zeichenfolgen angegeben werden.

Sind im Informationsfeld Buchstaben angegeben, so ist jeweils der entsprechende Groß- und Kleinbuchstabe gemeint (gleichgültig, ob der Buchstabe groß oder klein geschrieben ist). Soll zwischen Groß- und Kleinbuchstaben unterschieden werden, so ist der jeweilige Buchstabe mit einem vorangesetzten »\« zu kennzeichnen. Es bedeutet also »a« und »A« ein großes oder kleines a, »\a« ein kleines a, »\A« ein großes a.

Es können zwei Arten von Zeichenfolgen angegeben werden, nämlich »Suchzeichenfolgen« und »Ausnahmezeichenfolgen« (das sind Zeichenfolgen, die beim Durchsuchen des Textes übergangen werden sollen). Sie werden durch zwei aufeinander folgende Begrenzungszeichen voneinander getrennt. Durch Angabe von zwei aufeinander folgenden Begrenzungszeichen kann beliebig oft zwischen Suchzeichenfolgen und Ausnahmezeichenfolgen gewechselt werden.

Die Texte werden jeweils von links nach rechts durchsucht. Entspricht dabei eine im Text ab der jeweils beim Suchen erreichten Position stehende Zeichenfolge mehreren Such- und/oder Ausnahmezeichenfolgen, so hat die jeweils längere Zeichenfolge den Vorrang; bei gleich langen Zeichenfolgen entspricht ihre Rangfolge der Reihenfolge ihrer Angabe. Wird beim Durchsuchen (unter Berücksichtigung der genannten Rangfolge) eine Ausnahmezeichenfolge gefunden, so werden die dazugehörigen Zeichen im Text übersprungen; im Text wird dann ab der auf die Ausnahmezeichenfolge folgende Position weitergesucht. Für Ausnahmezeichenfolgen, die in Stringgruppen angegeben sind, gilt dies jedoch nicht; sie bewirken nur, dass andere Zeichenfolgen, die in der gleichen Stringgruppe angegeben sind, ggf. nicht zum Zuge kommen und somit ab der aktuellen Position im Text keine der in der Stringgruppe angegebene Suchzeichenfolge gefunden wird.

Achtung: Entsprechend dieser Vorgehensweise beim Durchsuchen des Textes nach angegebenen Zeichenfolgen wird z. B. beim Durchsuchen des Textes »...01234...« nach den Zeichenfolgen »/1234/01/« die Zeichenfolge »01« und nicht die Zeichenfolge »1234« gefunden. Grund: Die Zeichenfolge »1234« ist zwar länger als die Zeichenfolge »01«, aber entscheidend ist in diesem Fall, dass die Zeichenfolge »01« weiter links im Text steht und deshalb beim Durchsuchen von links nach rechts zuerst gefunden wird. Ebenso wird z. B. beim Durchsuchen des Textes »...01234...« nach den Zeichenfolgen »/1234//01/« zuerst die Ausnahmezei-

chenfolge »01« gefunden; dann wird im Text ab der Position, die auf die Zeichenfolge »01« folgt, weitergesucht und somit die Zeichenfolge »1234« nicht mehr gefunden. Außerdem ist bei gleich langen Such- und Ausnahmezeichenfolgen auf die Reihenfolge zu achten. So ist z. B. die Angabe »/{\a}{\a}/xy/« nicht sinnvoll, weil die Zeichenfolgen »{\a}{\a}« und »xy« beide zwei Zeichen lang sind (»{\a}« gilt als ein Zeichen, weil es stellvertretend für einen Kleinbuchstaben steht), und bei der Überprüfung der im Text stehenden Zeichenfolge »xy« die Suchzeichenfolge »{\a}{\a}« zum Zuge kommt, weil sie vor der Ausnahmezeichenfolge »xy« angegeben ist. Damit in diesem Fall die Ausnahmezeichenfolge wirkt, muss »/xy/{\a}{\a}/« angegeben werden.

Reicht für die Angaben eine Zeile nicht aus, so können Fortsetzungszeilen (mit der gleichen Parameterkennung und dem gleichen Begrenzungszeichen; vgl. Seite 706) verwendet werden.

Zeichen, Zeichengruppen-Kennung, Stringgruppen-Kennung, Zahlenwertbedingung sowie Verweis werden im folgenden »Element« der Suchzeichenfolge genannt. Eine vor einem Zeichen, einer Zeichengruppen-Kennung, einer Stringgruppen-Kennung sowie einem Verweis stehende Häufigkeitsbedingung gehört mit zum Element.

## Zeichengruppen und Stringgruppen

- { . . . } Vordefinierte Zeichengruppe { . . . } (siehe Seite 709).
- {z:xy} Selbst definierte Zeichengruppe xy.
- {c:xy} Alternative Schreibweise für {z:xy}.
- {s:xy} Selbst definierte Stringgruppe xy.
- [ . . . ] Lokale Zeichengruppe: Innerhalb von eckigen Klammern angegebene Zeichen werden jeweils als eine Zeichengruppe interpretiert, die diese Zeichen enthält. Es sind die gleichen Angaben wie bei der Definition von Zeichengruppen (siehe Parameterart V Seite 709 ff.) erlaubt.

## Zahlenwertbedingungen

Eine Zahlenwertbedingung steht stellvertretend für eine vorzeichenlose Zahl (aus arabischen Ziffern), die eine bestimmte Bedingung bezüglich ihres Wertes erfüllen muss. Im zu durchsuchenden Text darf weder vor noch nach einer solchen Zahl eine weitere Ziffer stehen.

- {#} Zahl mit beliebigem Wert.
- {#n} Zahl mit dem Wert n.
- {!n} Zahl mit einem Wert ungleich n.
- {#n-m} Zahl mit einem Wert von n bis m.
- {!n-m} Zahl mit einem Wert kleiner n oder größer m.

Wird für »m« eine Null angegeben, so ist dies gleichbedeutend mit dem maximal möglichen Wert 999999999.

Eine Zahlenwertbedingung darf nur mit der Häufigkeitsbedingung »{0}« bzw. »{0-1}« (d. h. darf fehlen oder einmal vorkommen) versehen werden, andere sind überflüssig oder sinnlos.

## Häufigkeitsbedingungen

{n} Enthält ein Element eine solche Häufigkeitsbedingung, so müssen im zu durchsuchenden Text die Zeichen, die dem im Element angegebenen Zeichen (der durch die Gruppenkennung angegebenen Gruppe, dem angegebenen Verweis) entsprechen, mindestens n-mal unmittelbar hintereinander vorkommen. Alle n Vorkommen dieser Zeichen werden diesem Element zugeordnet.

»n« ist eine ein- oder zweistellige Zahl und gibt die gewünschte Häufigkeit an. Wird für »n« eine Null angegeben, entspricht dies der Angabe »{0-1}« und bedeutet, dass die dem Element entsprechenden Zeichen im Text ganz fehlen oder einmal vorkommen dürfen. Werden für »n« zwei Nullen angegeben, entspricht dies der Angabe »{1-0}« und bedeutet, dass die dem Element entsprechenden Zeichen im Text mindestens einmal vorhanden sein müssen und beliebig oft vorkommen dürfen.

{n-m} Enthält ein Element eine solche Häufigkeitsbedingung, so müssen im zu durchsuchenden Text die Zeichen, die dem im Element angegebenen Zeichen (der durch die Gruppenkennung angegebenen Gruppe, dem angegebenen Verweis) entsprechen, mindestens n-mal unmittelbar hintereinander vorkommen. Alle n Vorkommen dieser Zeichen werden diesem Element zugeordnet. Kommen im zu durchsuchenden Text die Zeichen, die dem im Element angegebenen Zeichen (der durch die Gruppenkennung angegebenen Gruppe, dem angegebenen Verweis) entsprechen, mehr als n-mal unmittelbar hintereinander vor, so werden bis zu m Vorkommen dieser Zeichen diesem Element zugeordnet.

Achtung: Die Angabe »m« beinhaltet also nicht, dass im zu durchsuchenden Text die Zeichen, die dem im Element angegebenen Zeichen (der durch die Gruppenkennung angegebenen Gruppe, dem angegebenen Verweis) entsprechen, höchstens m-mal unmittelbar hintereinander vorkommen dürfen.

»n« und »m« sind ein- oder zweistellige Zahlen und geben die gewünschte Häufigkeiten an. Wird für »n« eine Null angegeben, so dürfen die dem Element entsprechenden Zeichen im Text ganz fehlen, wird für »m« eine Null angegeben, so dürfen sie beliebig oft vorkommen.

Beim Durchsuchen des Textes wird so früh wie möglich versucht, von dem mit der Häufigkeitsbedingung versehenen Element auf das nächste Element der Suchzeichenfolge überzugehen, um eine der Suchzeichenfolge im Text entsprechende Zeichenfolge zu finden, die

auch den unmittelbar nachfolgenden Elementen der Suchzeichenfolge (bis zu einem Element, bei dem sich die angegebenen Häufigkeiten »n« und »m« unterscheiden) entspricht.

{n--m} entspricht der Angabe »{n-m}«, jedoch wird beim Durchsuchen des Textes versucht, die Zeichen des Textes so lange als übereinstimmend mit dem mit einer solchen Häufigkeitsbedingung versehenen Element gelten zu lassen, bis die angegebene Häufigkeit »m« erreicht ist.

Wie oben beschrieben, hat beim Durchsuchen des Textes die längste einer Suchzeichenfolge entsprechende Zeichenfolge den Vorrang. Ist für ein Element die Häufigkeit »n« angegeben, so wird für die Berechnung der für die Rangfolge der Zeichenfolge maßgebenden Länge der Zeichenfolge dieses Element als aus n Zeichen bestehend gewertet.

Beispiele:

»/A{0}\\*B/« ist gleichbedeutend mit »/A{0-1}\\*B/« und definiert eine Suchzeichenfolge, die aus »a«, keinem oder einem »\*«, und einem »b« (in dieser Reihenfolge) besteht.

»/A{00}\\*B/« ist gleichbedeutend mit »/A{1-0}\\*B/« und definiert eine Suchzeichenfolge, die aus »a«, beliebig vielen (mindestens aber einem) »\*«, und einem »b« (in dieser Reihenfolge) besteht.

»/A{0-0}\\*B/« definiert eine Suchzeichenfolge, die aus »a«, keinem oder beliebig vielen »\*«, und einem »b« (in dieser Reihenfolge) besteht.

»:{2-4}{\0}:« definiert eine Suchzeichenfolge, die aus zwei bis vier Ziffern besteht. Mit dieser Suchzeichenfolge werden jedoch auch vier aufeinander folgende Ziffern gefunden, die Teil einer fünf- oder mehrstelligen Ziffernfolge sind. Soll dies ausgeschlossen werden, so muss entweder zusätzlich zur Suchzeichenfolge auch eine entsprechende Ausnahmezeichenfolge angegeben werden (also »:{2-4}{\0}::: {5-0}{\0}:«) oder durch Angabe der Zeichen, die links und rechts von der gesuchten Ziffernfolge stehen, die Anzahl der Ziffern begrenzt werden (z. B. »:({2-4}{\0}):«).

Im Text stehe die Zeichenfolge »1230000«; der Suchzeichenfolge »{1-5}{\0}0« entspräche dann die Textzeichenfolge »1230«, während der Suchzeichenfolge »{1--5}{\0}0« die Zeichenfolge »123000« entspräche.

Im Text stehe die Zeichenfolge »abxdxyzxyz.«; den Suchzeichenfolgen »{2-4}{&a}XYZ«, »{2-5}{&a}XYZ« usw. bis einschließlich »{2-9}{&a}XYZ« entspräche dann die Textzeichenfolge »abxdxyz«, während den beiden Suchzeichenfolgen »{2--9}{&a}XYZ« und »{2--8}{&a}XYZ« in diesem Text nichts, der Suchzeichenfolge »{2--7}{&a}XYZ« die Textzeichenfolge »abxdxyzxyz«, der Suchzeichenfolge »{2--6}{&a}XYZ« die Textzeichenfolge »bxdxyzxyz«, der Suchzeichenfolge »{2--5}{&a}XYZ« die Textzeichenfolge »xdxyzxyz«, der Suchzeichenfolge »{2--4}{&a}XYZ« die Textzeichenfolge »abxdxyz« entspräche.

## Verweise (in Suchzeichenfolgen)

- `{+n=}` Verweis auf das  $n$ -te Element der Suchzeichenfolge: die Zeichen, die im zu durchsuchenden Text dem  $n$ -ten Element der Suchzeichenfolge entsprechen, müssen auch an dieser Stelle im Text vorkommen; dabei werden Groß- und Kleinbuchstaben unterschieden. Es darf nur auf ein links vom Verweis stehendes Element verwiesen werden.  
Für » $n$ « ist eine ein- oder zweistellige Zahl einzusetzen.
- `{+n:}` Ist gleichbedeutend mit »`{+n=}`«, jedoch wird zwischen Groß- und Kleinbuchstaben nicht unterschieden.
- `{-n=}` Verweis auf das  $n$ -letzte Element der Suchzeichenfolge: die Zeichen, die im zu durchsuchenden Text dem vom Ende der Zeichenfolge her gezählten  $n$ -ten Element der Suchzeichenfolge entsprechen, müssen auch an dieser Stelle im Text vorkommen; dabei werden Groß- und Kleinbuchstaben unterschieden. Es darf nur auf ein links vom Verweis stehendes Element verwiesen werden.  
Für » $n$ « ist eine ein- oder zweistellige Zahl einzusetzen.
- `{-n:}` Ist gleichbedeutend mit »`{-n=}`«, jedoch wird zwischen Groß- und Kleinbuchstaben nicht unterschieden.

### Beispiele:

»`{&a}{\0}{+1=}` | « ist gleichwertig mit »`{&a}{\0}{-3=}`« und bezeichnet eine aus 3 Zeichen bestehende Suchzeichenfolge, die zwei identische Buchstaben rechts und links von einer Ziffer enthält (z. B. `a2a`, aber nicht `A2a` oder `a2A`); sie muss nämlich bestehen aus einem beliebigen Buchstaben (»`{&a}`«), einer Ziffer (»`{\0}`«) und einem weiteren Zeichen, das mit dem ersten (»`{+1=}`«) bzw. drittletzten (»`{-3=}`«) Zeichen der Zeichenfolge im Text identisch ist, also dem gleichen Großbuchstaben bzw. dem gleichen Kleinbuchstaben wie vor der Ziffer.

»`{00}{\0}:{+1=}` | « bezeichnet eine Suchzeichenfolge, die rechts von einem Doppelpunkt die gleiche Ziffernfolge wie links von diesem Doppelpunkt enthält (z. B. `123:123`, aber auch `23:23` von `123:234`); die Zeichenfolge besteht aus beliebig vielen (»`{00}`«) Ziffern (»`{\0}`«), einem Doppelpunkt (»`:`«) und der gleichen Ziffernfolge wie vor dem Doppelpunkt (»`{+1=}`«), d. h. aus den gleichen Zeichen, die im durchsuchten Text dem ersten Element der Suchzeichenfolge entsprechen.

»`{#}:{+1=}` | « bezeichnet eine Suchzeichenfolge, die rechts von einem Doppelpunkt die gleiche Ziffernfolge wie links von diesem Doppelpunkt enthält (z. B. `123:123`, aber auch `23:23` von `23:234`, jedoch nicht `123:123` von `0123:123`); die Zeichenfolge besteht aus einer beliebigen Zahl (»`{#}`«), einem Doppelpunkt (»`:`«) und der gleichen Ziffernfolge wie vor dem Doppelpunkt (»`{+1=}`«), genauer: aus den gleichen Zeichen, die im durchsuchten Text dem ersten Element der Suchzeichenfolge entsprechen.

## Umgebungsbedingungen

Mit Hilfe von Umgebungsbedingungen kann festgelegt werden, in welcher Umgebung eine Zeichenfolge im zu durchsuchenden Text stehen muss. Eine Suchzeichenfolge, zu der Umgebungsbedingungen angegeben sind, besteht aus einer »Kernzeichenfolge« (die einer Suchzeichenfolge ohne Umgebungsbedingungen entspricht) und einer linken und/oder einer rechten »Randzeichenfolge«. Die der linken Randzeichenfolge entsprechende Zeichenfolge muss im zu durchsuchenden Text unmittelbar vor der Kernzeichenfolge entsprechenden Textzeichenfolge stehen, die der rechten Randzeichenfolge entsprechende Zeichenfolge unmittelbar nach der Kernzeichenfolge entsprechenden Textzeichenfolge.

Die linke Randzeichenfolge wird vor der Kernzeichenfolge angegeben und von dieser durch das unten beschriebene »Begrenzungszeichen für linken Rand« abgetrennt; die rechte Randzeichenfolge wird hinter der Kernzeichenfolge angegeben und von dieser durch das unten beschriebene »Begrenzungszeichen für rechten Rand« abgetrennt.

Für die Angabe der Randzeichenfolgen gelten die gleichen Regeln wie für die Angabe von Suchzeichenfolgen ohne Umgebungsbedingungen; bei der Angabe von Verweisen ist jedoch zu beachten, dass beim Zählen der Elemente die Elemente der Randzeichenfolgen mitgezählt werden, dass jedoch nur auf Elemente der Kernzeichenfolge verwiesen werden darf.

{ []            Begrenzungszeichen für linken Rand der Kernzeichenfolge

{ }            Begrenzungszeichen für rechten Rand der Kernzeichenfolge

Beispiele:

»: : {\0} { [] 123 : 123 {} } {\0} : : 123 : « definiert eine Such-Tabelle, mit der nach der Zahl 123 gesucht wird, wobei also z. B. die Ziffernfolge 123 in der Zahl 1234 nicht gefunden werden soll. Für die Angabe »beliebige Ziffer« wird die vordefinierte Zeichengruppe mit der Zeichengruppen-Kennung »{\0}« verwendet. Am Anfang wird mit zwei aufeinander folgenden Begrenzungszeichen »: : « auf Ausnahmezeichenfolgen umgeschaltet. Die erste Ausnahmezeichenfolge ist »{\0} { [] 123 «; sie besagt, dass 123 nicht gefunden werden soll, wenn unmittelbar links davon eine Ziffer steht. Nach einem Begrenzungszeichen folgt die zweite Ausnahmezeichenfolge »123 {} } {\0} «; sie besagt, dass 123 nicht gefunden werden soll, wenn unmittelbar rechts davon ein Ziffer steht. Danach wird mit zwei aufeinander folgenden Begrenzungszeichen von Ausnahmezeichenfolgen auf Suchzeichenfolgen umgeschaltet. Jetzt folgt die zu suchende Zeichenfolge »123« und noch das abschließende Begrenzungszeichen. Würden die Ausnahmezeichenfolgen erst nach der Suchzeichenfolge angegeben (also »: : 123 : : {\0} { [] 123 : 123 {} } {\0} : «), so würden alle Ziffernfolgen 123 gefunden, da die Ausnahmezeichenfolgen wirkungslos blieben, weil sie gleich lang sind (die Zeichen der Randzeichenfolge zählen bei der Längenberechnung nicht mit) wie die Suchzeichenfolge und bei gleich langen Zeichenfolgen die Reihenfolge maßgebend ist, in der sie angegeben sind.

Anmerkung: Die gleiche Wirkung hätten die Such-Tabellen »: : 123 : : { 4-0 } {\0} : : « und »: : { #123 } : : «.

»: : { 0 } { [] UND {} } { 0 } : : « definiert eine Suchzeichenfolge, mit der nach allen Zei-



chenfolgen »und« gesucht wird, die hinter einem Leerzeichen oder am linken Rand des abzusuchenden Textes und außerdem vor einem Leerzeichen oder am rechten Rand des abzusuchenden Textes stehen. Die Angabe ist wie folgt zu lesen: »:« = freigesähltes Begrenzungszeichen für die Such-Tabelle, »{0}« = ein oder kein Leerzeichen, »{ }« = Begrenzungszeichen für linken Rand der Kernzeichenfolge (zusammen mit der davor stehenden Angabe bedeutet dies also: Unmittelbar links vor der Kernzeichenfolge muss ein Leerzeichen stehen, oder es darf, wenn es fehlt, nichts stehen; der linke Rand wäre im letzteren Fall gleichzeitig der Anfang des zu durchsuchenden Textes), »UND« = gesuchte Zeichenfolge (Kernzeichenfolge), »{ }« = Begrenzungszeichen für »rechter Rand der Kernzeichenfolge«, »{0}« = ein oder kein Leerzeichen (zusammen mit der davor stehenden Angabe bedeutet dies also: Unmittelbar rechts nach der gesuchten Zeichenfolge muss ein Leerzeichen stehen, oder es darf, wenn es fehlt, nichts stehen; der rechte Rand wäre im letzteren Fall gleichzeitig das Ende des zu durchsuchenden Textes).

## Parameterart X: Austausch-Tabelle

(Gilt für »{}-Konvention«, für »<>-Konvention« siehe Seite 752.)

Im Informationsfeld werden Zeichenfolgenpaare angegeben, deren jeweils erste Zeichenfolge die Suchzeichenfolge ist, die durch die jeweils zweite Zeichenfolge, die Ersatzzeichenfolge, ersetzt werden soll. Such- und Ersatzzeichenfolgen dürfen verschieden lang sein. Sie werden durch ein frei wählbares Begrenzungszeichen (außer Backslash und Klammern) voneinander getrennt. Das Begrenzungszeichen ist das erste Zeichen (Spalte 11) im Informationsfeld des Parameters. Die letzte Zeichenfolge muss durch ein Begrenzungszeichen abgeschlossen werden.

Außer Zeichenfolgenpaaren können Ausnahmezeichenfolgen (das sind Zeichenfolgen, die beim Austauschen übergangen werden sollen und für die deshalb keine Ersatzzeichenfolgen existieren) angegeben werden. Das Umschalten von Zeichenfolgenpaaren auf Ausnahmezeichenfolgen geschieht dadurch, dass an einer Stelle, an der eine Suchzeichenfolge erwartet wird, nochmals ein Begrenzungszeichen steht. Daran anschließend können eine oder mehrere Ausnahmezeichenfolgen angegeben werden. Das Umschalten von Ausnahmezeichenfolgen auf Zeichenfolgenpaare geschieht dadurch, dass an einer Stelle, an der eine Ausnahmezeichenfolge erwartet wird, nochmals ein Begrenzungszeichen steht. Daran anschließend können wieder Zeichenfolgenpaare angegeben werden. Auf diese Weise kann beliebig oft zwischen der Angabe von Zeichenfolgenpaaren und Ausnahmezeichenfolgen gewechselt werden.

Für die Angabe der Suchzeichenfolgen und der Ausnahmezeichenfolgen gelten die gleichen Regeln wie bei der Such-Tabelle (Parameterart IX). Zusätzlich können in Suchzeichenfolgen noch Begrenzungszeichenfolgen für Elementbereiche (s. u.) angegeben werden.

Reicht für die Angaben eine Zeile nicht aus, so können Fortsetzungszeilen (mit der gleichen Parameterkennung und dem gleichen Begrenzungszeichen; vgl. Seite 706) verwendet werden.

## Elementbereiche (in Suchzeichenfolgen)

In den Ersatzzeichenfolgen können an Stelle einzelner Zeichen Verweise auf Zeichen in der dazugehörenden Suchzeichenfolge (einschließlich evtl. angegebener Randzeichenfolgen) angegeben werden, wenn die Zeichen, die im Text einem Element der Suchzeichenfolge entsprechen, beim Austauschen an dieser Stelle übernommen werden sollen. Ein solcher Verweis kann sich auf ein einzelnes Element oder auf mehrere aufeinander folgende Elemente in der Suchzeichenfolge beziehen. Um bei längeren Suchzeichenfolgen ein mühsames Abzählen der Elemente zu vermeiden, kann die Suchzeichenfolge in Bereiche unterteilt und dann in der Ersatzzeichenfolge auf solche Bereiche verwiesen werden:

{ | }            Begrenzungszeichen zwischen zwei Elementbereichen.

## Verweise (in Ersatzzeichenfolgen)

In den Ersatzzeichenfolgen können an Stelle einzelner Zeichen Verweise auf Elemente und Elementbereiche der dazugehörenden Suchzeichenfolge (einschließlich evtl. angegebener Randzeichenfolgen) angegeben werden, wenn die Zeichen, die im Text einem Element der Suchzeichenfolge entsprechen, beim Austauschen an dieser Stelle übernommen werden sollen:

{+n=}            Ist gleichbedeutend mit »{+n-n=}«

{+n-m=}        Verweis auf das n-te bis m-te Element der Suchzeichenfolge: die Zeichen, die im zu durchsuchenden Text dem n-ten bis m-ten Element der Suchzeichenfolge entsprechen, werden beim Austauschen an dieser Stelle unverändert in den Text eingesetzt. Der Verweis darf auf Elemente der Kernzeichenfolge und der Randzeichenfolgen verweisen. Für »n« und »m« ist eine ein- oder zweistellige Zahl einzusetzen. Wird für »n« bzw. »m« der Wert Null angegeben, so ist damit das erste bzw. letzte Element der Kernzeichenfolge gemeint.

{-n=}            Ist gleichbedeutend mit »{-n-n=}«

{-n-m=}        Verweis auf das n-letzte bis m-letzte Element der Suchzeichenfolge: die Zeichen, die im zu durchsuchenden Text dem n-ten bis m-ten Element, jeweils vom Ende her gezählt, entsprechen, werden beim Austauschen an dieser Stelle unverändert in den Text eingesetzt. Der Verweis darf auf Elemente der Kernzeichenfolge und der Randzeichenfolgen verweisen. Für »n« und »m« ist eine ein- oder zweistellige Zahl einzusetzen. Wird für »n« bzw. »m« der Wert Null angegeben, so ist damit das erste bzw. letzte Element der Kernzeichenfolge gemeint.

{=n=}            Ist gleichbedeutend mit »{=n-n=}«

{=n-m=}        Verweis auf den n-te bis m-ten Elementbereich der Suchzeichenfolge: die Zeichen, die im zu durchsuchenden Text dem n-ten bis m-ten Elementbereich der Suchzeichenfolge entsprechen, werden beim Austauschen an dieser Stelle unverändert in den Text eingesetzt. Für »n« und »m« ist eine ein- oder zweistellige Zahl einzusetzen. Wird für »n« und »m« der Wert Null angegeben, so ist damit die Kernzeichenfolge gemeint.

{+n+}	Ist gleichbedeutend mit »{+n-n+}«
{+n-m+}	Wie »{+n-m=}«, jedoch werden beim Austauschen alle den angegebenen Elementen der Suchzeichenfolge entsprechenden Kleinbuchstaben in Großbuchstaben umgewandelt.
{-n+}	Ist gleichbedeutend mit »{-n-n+}«
{-n-m+}	Wie »{-n-m=}«, jedoch werden beim Austauschen alle den angegebenen Elementen der Suchzeichenfolge entsprechenden Kleinbuchstaben in Großbuchstaben umgewandelt.
{=n+}	Ist gleichbedeutend mit »{=n-n+}«
{=n-m+}	Wie »{=n-m=}«, jedoch werden beim Austauschen alle den angegebenen Elementbereichen der Suchzeichenfolge entsprechenden Kleinbuchstaben in Großbuchstaben umgewandelt.
{+n-}	Ist gleichbedeutend mit »{+n-n-}«
{+n-m-}	Wie »{+n-m=}«, jedoch werden beim Austauschen alle den angegebenen Elementen der Suchzeichenfolge entsprechenden Großbuchstaben in Kleinbuchstaben umgewandelt.
{-n-}	Ist gleichbedeutend mit »{-n-n-}«
{-n-m-}	Wie »{-n-m=}«, jedoch werden beim Austauschen alle den angegebenen Elementen der Suchzeichenfolge entsprechenden Großbuchstaben in Kleinbuchstaben umgewandelt.
{=n-}	Ist gleichbedeutend mit »{=n-n-}«
{=n-m-}	Wie »{=n-m=}«, jedoch werden beim Austauschen alle den angegebenen Elementbereichen der Suchzeichenfolge entsprechenden Großbuchstaben in Kleinbuchstaben umgewandelt.
{+n;}	Ist gleichbedeutend mit »{+n-n;}«
{+n-m;}	Wie »{+n-m=}«, jedoch werden beim Austauschen alle den angegebenen Elementen der Suchzeichenfolge entsprechenden Zeichen aus dem ASCII-Zeichensatz in die entsprechenden mit »^« codierten Zeichen des TUSTEP-Zeichensatz umgewandelt (vgl. Tabellen Seite 763 und Seite 815).
{-n;}	Ist gleichbedeutend mit »{-n-n;}«
{-n-m;}	Wie »{-n-m;}«, jedoch werden beim Austauschen alle den angegebenen Elementen der Suchzeichenfolge entsprechenden Zeichen aus dem ASCII-Zeichensatz in die entsprechenden mit »^« codierten Zeichen des TUSTEP-Zeichensatz umgewandelt (vgl. Tabellen Seite 763 und Seite 815).
{=n;}	Ist gleichbedeutend mit »{=n-n;}«
{=n-m;}	Wie »{=n-m=}«, jedoch werden beim Austauschen alle den angegebenen Elementbereichen der Suchzeichenfolge entsprechenden Zeichen aus dem ASCII-Zeichensatz in die entsprechenden mit »^« co-

	dierten Zeichen des TUSTEP-Zeichensatz umgewandelt (vgl. Tabellen Seite 763 und Seite 815).
{+n! }	Ist gleichbedeutend mit »{+n-n! }«
{+n-m! }	Wie »{+n-m=}«, jedoch werden beim Austauschen alle den angegebenen Elementen der Suchzeichenfolge entsprechenden mit »^« codierten Zeichen aus dem TUSTEP-Zeichensatz in die entsprechenden ohne »^« codierten Zeichen aus dem ASCII-Zeichensatz umgewandelt (vgl. Tabellen Seite 763 und Seite 815).
{-n! }	Ist gleichbedeutend mit »{-n-n! }«
{-n-m! }	Wie »{-n-m=}«, jedoch werden beim Austauschen alle den angegebenen Elementen der Suchzeichenfolge entsprechenden mit »^« codierten Zeichen aus dem TUSTEP-Zeichensatz in die entsprechenden ohne »^« codierten Zeichen aus dem ASCII-Zeichensatz umgewandelt (vgl. Tabellen Seite 763 und Seite 815).
{=n! }	Ist gleichbedeutend mit »{=n-n! }«
{=n-m! }	Wie »{=n-m=}«, jedoch werden beim Austauschen alle den angegebenen Elementbereichen der Suchzeichenfolge entsprechenden mit »^« codierten Zeichen aus dem TUSTEP-Zeichensatz in die entsprechenden ohne »^« codierten Zeichen aus dem ASCII-Zeichensatz umgewandelt (vgl. Tabellen Seite 763 und Seite 815).

## Parameterart XI: Sonstiges

Das Informationsfeld enthält Angaben in einem bei der jeweiligen Parameterbeschreibung genannten Format.

Reicht für die Angaben eine Zeile nicht aus, so können Fortsetzungszeilen (mit der gleichen Parameterkennung) verwendet werden. An jedem Zeilenende wird ein Leerzeichen ergänzt.

## Parameterart XII: Recherchier-Tabelle

*(Gilt für »{}-Konvention«, für »<>-Konvention« siehe Seite 756.)*

Im Informationsfeld werden Zeichenfolgen angegeben, nach denen gesucht werden soll. Sie werden durch ein frei wählbares Begrenzungszeichen (außer Backslash und Klammern) voneinander getrennt. Das Begrenzungszeichen ist das erste Zeichen (Spalte 11) im Informationsfeld des Parameters. Die letzte Zeichenfolge muss durch ein Begrenzungszeichen abgeschlossen sein.

Achtung: Das Fragezeichen und der Stern haben bei dieser Parameterart eine spezielle Bedeutung. Ein Fragezeichen steht stellvertretend für ein beliebiges TUSTEP-Zeichen und ein Stern für null bis beliebig viele beliebige TUSTEP-Zeichen.

An Stelle eines einzelnen Zeichens einer Zeichenfolge sind folgende Angaben möglich:

- Eine in eckige Klammern eingeschlossene Gruppe von Zeichen.

Die Zeichen innerhalb der eckigen Klammern werden als eine Zeichengruppe mit den angegebenen Zeichen interpretiert. Es sind die gleichen Angaben wie bei der Definition von Zeichengruppen (siehe Parameterart V Seite 709 ff.) erlaubt.

- Ein in geschweifte Klammern eingeschlossene Folge von Zeichen.

Innerhalb der geschweiften Klammern sind keine Unterschiede (Ersetzungen, Auslassungen und Einfügungen) erlaubt.

Achtung: Um eine Verwechslung mit einer Zeichengruppen-Kennung in jedem Fall auszuschließen, muss unmittelbar nach der öffnenden geschweiften Klammer ein Gleichheitszeichen eingefügt werden; es gehört nicht zur eingeschlossenen Zeichenfolge.

- Eine in runde Klammern eingeschlossene Folge von Zeichen.

Die in runde Klammern eingeschlossenen Zeichen gelten als eine einzige Zeichenfolge; zwischen den Klammern vorkommende Begrenzungszeichen werden nicht als solche gewertet.

- Ein Backslash und ein beliebiges Zeichen.

Ein Backslash bewirkt, dass das unmittelbar nachfolgende Zeichen seine eigentliche Bedeutung behält. Soll z. B. nach einem Fragezeichen gesucht werden, muss »\ ?« angegeben werden. Erforderlich ist eine Kennzeichnung mit dem Backslash für Fragezeichen, Stern, Backslash und sämtliche Klammern. Wird ein Buchstabe mit einem Backslash gekennzeichnet, so ist nur der jeweilige Groß- bzw. Kleinbuchstabe gemeint; bei nicht gekennzeichneten Buchstaben ist jeweils der entsprechende Groß- und Kleinbuchstabe gemeint (gleichgültig, ob der Buchstabe groß oder klein geschrieben ist).

- Eine Kennung einer Zeichengruppe.

Soll eine zu suchende Zeichenfolge nur dann gefunden werden, wenn sie am Anfang bzw. am Ende der zu durchsuchenden Zeichenfolge steht, kann die Zeichenfolge am Anfang (unmittelbar nach dem Begrenzungszeichen) mit »{ [ ]« bzw. am Ende (unmittelbar vor dem Begrenzungszeichen) mit »} { }« gekennzeichnet werden.

Für jede Recherchier-Tabelle kann zusätzlich festgelegt werden,

- ob Groß- und Kleinschreibung bei nicht mit einem Backslash gekennzeichneten Buchstaben relevant ist,
- wieviele Unterschiede (Ersetzungen, Auslassungen und Einfügungen) je Zeichenfolge erlaubt sind,
- ob die gesuchte Zeichenfolge an einer Wortgrenze (= jedes Sonderzeichen) oder an der Textgrenze (= Anfang und Ende der zu durchsuchenden Zeichenfolge) beginnen und enden muss,

In welcher Weise dies festgelegt werden kann, ist bei der jeweiligen Parameterbeschreibung angegeben.

Reicht für die Angaben eine Zeile nicht aus, so können Fortsetzungszeilen (mit der gleichen Parameterkennung und dem gleichen Begrenzungszeichen; vgl. Seite 706) verwendet werden.

### **Besonderheiten der Parameterart XII-a**

Als Begrenzungszeichen kann ein beliebiges Zeichen (außer Backslash und Klammern) verwendet werden.

Es kann zusätzlich festgelegt werden, ob alle oder nur eine der angegebenen Zeichenfolgen gefunden werden muss.

In welcher Weise dies festgelegt werden kann, ist bei der jeweiligen Parameterbeschreibung angegeben.

### **Besonderheiten der Parameterart XII-b**

Als Begrenzungszeichen muss ein Leerzeichen verwendet werden.

Mehrere unmittelbar aufeinander folgende Leerzeichen werden als ein einziges Begrenzungszeichen gewertet. Eine leere Zeichenfolge kann mit zwei aufeinander folgenden Anführungszeichen (""") angegeben werden.

Anführungszeichen (") dürfen nur paarweise vorkommen. Jedes erste eines Paares wird als »runde Klammer auf«, jede zweite als »runde Klammer zu« gewertet. Die so eingeschlossenen Zeichen gelten jeweils als eine einzige Zeichenfolge; zwischen den Klammern vorkommende Leerzeichen werden nicht als Begrenzungszeichen gewertet.

Die (zwischen Leerzeichen stehenden) Zeichenfolgen AND, OR, NOT, AND NOT und OR NOT werden als logische Operatoren interpretiert. Auf diese Weise kann festgelegt werden, welche der angegebenen Zeichenfolgen im zu durchsuchenden Text gefunden werden müssen.

Ist zwischen zwei Zeichenfolgen (die keine logische Operatoren sind) kein logischer Operator angegeben, so wird ein logisches ODER angenommen; dieses logische ODER hat Vorrang vor allen angegebenen logischen Operatoren. Es kann jedoch festgelegt werden, dass stattdessen ein logisches UND oder ein logisches ODER ohne Vorrang angenommen wird. In welcher Weise dies festgelegt werden kann, ist bei der jeweiligen Parameterbeschreibung angegeben.

Im Übrigen gelten die für logischen Operatoren üblichen Regeln. Eine Klammerung zur Angabe der Priorität der logischen Operatoren ist nicht möglich.

**Beispiele:**

Die Anführungszeichen grenzen die Tabellen gegenüber dem umgebenden Text ab und sind nicht Bestandteil der Tabellen.

- » eins zwei «: »eins« oder »zwei« oder beide
- » eins OR zwei «: »eins« oder »zwei« oder beide
- » eins AND zwei «: Sowohl »eins« als auch »zwei«
- » eins AND NOT zwei «: »eins« aber nicht »zwei«
- » eins AND zwei drei «: »eins« und ( »zwei« oder »drei« oder beide )
- » NOT eins zwei «: Weder »eins« noch »zwei«

An den Stellen, an denen zwischen zwei Zeichenfolgen kein logischer Operator angegeben ist, wird ein logisches ODER mit Vorrang angenommen.

## Auswählen und Eliminieren von Textteilen

Einer der häufigsten Verarbeitungsschritte in TUSTEP ist das Auswählen bzw. Eliminieren bestimmter Teile aus einer Texteinheit (= zusammengehörender Text, bis zu 256000 Zeichen lang). Dies geschieht anhand von eindeutigen Kennzeichen, die im Text enthalten sind. Diese Kennzeichen können als »Anfangskennung« den Anfang und als »Endekennung« das Ende des auszuwählenden bzw. des zu eliminierenden Textteils kennzeichnen, oder sie können als öffnende bzw. schließende Klammern interpretiert werden, wobei dann jeweils der eingeklammerte Teil ausgewählt bzw. eliminiert wird.

Das Auswählen bzw. Eliminieren von Textteilen mit Anfangs- und Endekennungen und mit Klammern kann auch kombiniert werden. In diesem Fall werden zuerst die Textteile auf Grund der Anfangs- und Endekennungen ausgewählt und dann erfolgt für jeden ausgewählten Textteil einzeln die Auswahl auf Grund der Klammern.

Das Auswählen bzw. Eliminieren von Textteilen kann bei manchen Programmen auch mehrfach auf die gleiche Texteinheit angewandt und die jeweils ausgewählten Textteile zu einer neuen Texteinheit zusammengefügt werden. Auf diese Weise können die einzelnen Textteile umgestellt und in der gewünschten Reihenfolge angeordnet werden.

### Auswählen bzw. Eliminieren mit Anfangs- und Endekennungen

Die Kennzeichen, die den Anfang und das Ende des auszuwählenden bzw. zu eliminierenden Textteils kennzeichnen, werden mit je einem eigenen Parameter angegeben. Im Folgenden wird der Parameter für die Anfangskennungen **A** und der Parameter für die Endekennungen **E** genannt. Bei jedem der beiden Parametern können mehrere Zeichenfolgen als Kennzeichen angegeben werden; sie gelten dann jeweils als gleichwertig.

Über einen Index, der mit einem weiteren Parameter angegeben werden kann, wird bestimmt, ob der gekennzeichnete Textteil ausgewählt oder eliminiert werden soll, und ob die Kennzeichen selber mit ausgewählt bzw. mit eliminiert werden sollen. Der Parameter für den Index wird im Folgenden **ÆEI** genannt. Der Index ist eine Zahlenangabe, die wie folgt definiert ist:

1 = Es wird der erste mit Zeichenfolgen der Parameter **A** und **E** gekennzeichnete Textteil (er beginnt mit einer Anfangskennung und endet vor der nachfolgenden Endekennung bzw. am Ende der Texteinheit) ausgewählt. Der Rest wird eliminiert.

Ist nur der Parameter **A** angegeben, so endet der ausgewählte Textteil am Ende der Texteinheit; ist nur der Parameter **E** angegeben, so beginnt der ausgewählte Textteil am Anfang der Texteinheit.

0 = Es wird der Teil der Texteinheit eliminiert, der bei 1 ausgewählt würde.

3 = Wie 1, jedoch wird nicht nur der erste, sondern es werden alle mit Zeichenfolgen der Parameter **A** und **E** gekennzeichneten Textteile (der zweite beginnt mit



der Anfangskennung, die dem Ende des ersten gekennzeichneten Textteils folgt) ausgewählt. Der Rest wird eliminiert.

Ist nur der Parameter **A** angegeben, so beginnt der ausgewählte Textteil mit der letzten in der Texteinheit vorkommenden Anfangskennung und endet am Ende der Texteinheit; ist nur der Parameter **E** angegeben, so beginnt der ausgewählte Textteil am Anfang der Texteinheit und endet vor der letzten in der Texteinheit vorkommenden Endekennung.

2 = Es wird der Teil der Texteinheit eliminiert, der bei 3 ausgewählt würde.

Bei den Werten 0 bis 3 wird die Anfangskennung jeweils zum nachfolgenden Text gerechnet, während die Endekennung nicht mehr zum davor stehenden Text gerechnet wird. Durch Aufaddieren von 10 und/oder 20 kann diese Regelung für die Anfangs- und/oder Endekennung umgekehrt werden. Wird zu den Werten 10 aufaddiert (also 10 bis 13 angegeben), so wird die Anfangskennung jeweils nicht zum nachfolgenden Text gerechnet; wird 20 addiert, so wird die Endekennung jeweils noch zum davor stehenden Text gerechnet; wird 30 addiert, so wird die Anfangskennung nicht zum nachfolgenden Text und die Endekennung zum davor stehenden Text gerechnet.

Bei den Werten 2 und 3 wird die nächste Anfangskennung ab der ersten Position der zuletzt gefundenen Endekennung gesucht, da sie nicht mehr zum davor stehenden Textteil gehört. Anfangs- und Endekennung können sich also im Text überschneiden. Wurde auf diese Werte 20 oder 30 aufaddiert, so wird die nächste Anfangskennung erst ab der Position gesucht, die auf die letzte Position der zuletzt gefundenen Endekennung folgt, da diese noch zum davor stehenden Textteil gehört.

In den folgenden Tabellen wird die Wirkung des Index schematisch dargestellt. Die oberste und unterste Zeile stellt jeweils den Text dar, aus dem Teile ausgewählt bzw. eliminiert werden sollen. Dabei steht »[A]« jeweils für eine Anfangskennung, die mit dem Parameter **A** angegeben ist, und »[E]« jeweils für eine Endekennung, die mit dem Parameter **E** angegeben ist. Für jeweils zwei möglichen Werte des Index, der mit dem Parameter **AEI** angegeben werden kann, folgt danach eine weitere Zeile. Ein Wert steht jeweils in der linken und einer in der rechten Spalte. Ist mit dem Parameter **AEI** der links stehende Index angegeben, so werden die Teile ausgewählt, die in der mittleren Spalte mit »=« gekennzeichnet sind, und die mit »·« gekennzeichneten eliminiert; ist der rechts stehende Index angegeben, so werden die mit »=« gekennzeichneten eliminiert und die mit »·« gekennzeichneten ausgewählt.

1. Wenn Parameter A und Parameter E angegeben sind:

	xxxx [A] xxxx [E] xxxx [A] xxxx [E] xxxx [A] xxxx [E] xxxx	
1	.....	0
3	.....	2
11	.....	10
13	.....	12
21	.....	20
23	.....	22
31	.....	30
33	.....	32
	xxxx [A] xxxx [E] xxxx [A] xxxx [E] xxxx [A] xxxx [E] xxxx	

	xxxx [E] xxxx [A] xxxx [A] xxxx [E] xxxx [E] xxxx [A] xxxx	
1	.....	0
3	.....	2
11	.....	10
13	.....	12
21	.....	20
23	.....	22
31	.....	30
33	.....	32
	xxxx [E] xxxx [A] xxxx [A] xxxx [E] xxxx [E] xxxx [A] xxxx	

2. Wenn nur Parameter A angegeben ist:

	xxxx [A] xxxx [E] xxxx [A] xxxx [E] xxxx [A] xxxx [E] xxxx	
1	.....	0
3	.....	2
11	.....	10
13	.....	12
21	.....	20
23	.....	22
31	.....	30
33	.....	32
	xxxx [A] xxxx [E] xxxx [A] xxxx [E] xxxx [A] xxxx [E] xxxx	

	xxxx [E] xxxx [A] xxxx [A] xxxx [E] xxxx [E] xxxx [A] xxxx	
1	.....=====	0
3	.....=====	2
11	.....=====	10
13	.....=====	12
21	.....=====	20
23	.....=====	22
31	.....=====	30
33	.....=====	32
	xxxx [E] xxxx [A] xxxx [A] xxxx [E] xxxx [E] xxxx [A] xxxx	

3. Wenn nur Parameter E angegeben ist:

	xxxx [A] xxxx [E] xxxx [A] xxxx [E] xxxx [A] xxxx [E] xxxx	
1	=====.....	0
3	=====.....	2
11	=====.....	10
13	=====.....	12
21	=====.....	20
23	=====.....	22
31	=====.....	30
33	=====.....	32
	xxxx [A] xxxx [E] xxxx [A] xxxx [E] xxxx [A] xxxx [E] xxxx	

	xxxx [E] xxxx [A] xxxx [A] xxxx [E] xxxx [E] xxxx [A] xxxx	
1	=====.....	0
3	=====.....	2
11	=====.....	10
13	=====.....	12
21	=====.....	20
23	=====.....	22
31	=====.....	30
33	=====.....	32
	xxxx [E] xxxx [A] xxxx [A] xxxx [E] xxxx [E] xxxx [A] xxxx	

## Auswählen bzw. Eliminieren mit Klammern

Die Kennzeichen, die als öffnende und als schließende Klammern dienen, werden mit je einem eigenen Parameter angegeben. Im Folgenden wird der Parameter für die öffnenden Klammern mit `KLA` und der Parameter für die schließenden Klammern mit `KLZ` bezeichnet. Bei jedem der beiden Parametern können mehrere Zeichenfolgen als »Klammern« angegeben werden; sie gelten dann jeweils als gleichwertig.

Über einen Index, der mit einem weiteren Parameter angegeben werden kann, wird bestimmt, ob der »eingeklammerte« Textteil ausgewählt oder eliminiert werden soll, und ob die Kennzeichen selber mit ausgewählt bzw. mit eliminiert werden sollen. Der Parameter für den Index wird im Folgenden `KLI` genannt. Der Index ist eine Zahlenangabe, die wie folgt definiert ist:

- 0 = Es werden die Teile (einschließlich der Klammern) eliminiert, die eingeklammert sind. Fehlende Klammern werden am Anfang bzw. am Ende der Texteinheit logisch ergänzt.
- 1 = Es werden die Teile ausgewählt, die bei 0 eliminiert würden.
- 2 = Wie 0, jedoch werden fehlende Klammern nicht logisch ergänzt, sondern die unpaarigen Klammern werden ignoriert.
- 3 = Es werden die Teile ausgewählt, die bei 2 eliminiert würden.

Bei den Werten 0 bis 3 werden die Klammern jeweils zum eingeklammerten Text gerechnet und werden mit ihm eliminiert bzw. bleiben mit ihm erhalten. Durch Aufaddieren von 10 und/oder 20 kann diese Regelung für die öffnenden und/oder schließenden Klammern umgekehrt werden. Wird auf diese Werte 10 aufaddiert (also 10 bis 13 angegeben), so werden die öffnenden Klammern nicht zum eingeklammerten Text gerechnet; wird 20 addiert, so werden die schließenden Klammern nicht zum eingeklammerten Text gerechnet; wird 30 addiert, so werden beide Klammern nicht zum eingeklammerten Text gerechnet.

In den folgenden Tabellen wird die Wirkung des Index schematisch dargestellt. Die oberste und unterste Zeile stellt jeweils den Text dar, aus dem Teile ausgewählt bzw. eliminiert werden sollen. Dabei steht » ( ( « jeweils für eine öffnende Klammer, die mit dem Parameter `KLA` angegeben ist, und » ) ) « jeweils für eine schließende Klammer, die mit dem Parameter `KLZ` angegeben ist. Für jeweils zwei möglichen Werte des Index, der mit dem Parameter `KLI` angegeben werden kann, folgt danach eine weitere Zeile. Ein Wert steht jeweils in der linken und einer in der rechten Spalte. Ist mit dem Parameter `KLI` der links stehende Index angegeben, so werden die Teile ausgewählt, die in der mittleren Spalte mit »=« gekennzeichnet sind, und die mit »·« gekennzeichneten eliminiert; ist der rechts stehende Index angegeben, so werden die mit »=« gekennzeichneten eliminiert und die mit »·« gekennzeichneten ausgewählt.

	xxxx ((xxxx)) xxxx ((xxxx)) xxxx ((xxxx)) xxxx	
0	=====	1
2	=====	3
10	=====	11
12	=====	13
20	=====	21
22	=====	23
30	=====	31
32	=====	33
	xxxx ((xxxx)) xxxx ((xxxx)) xxxx ((xxxx)) xxxx	

	xxxx )) xxxx ((xxxx ((xxxx)) xxxx)) xxxx ((xxxx	
0	.....=====	1
2	=====	3
10	.....=====	11
12	=====	13
20	.....=====	21
22	=====	23
30	.....=====	31
32	=====	33
	xxxx )) xxxx ((xxxx ((xxxx)) xxxx)) xxxx ((xxxx	



**<>-Parameter**

---

Allgemeines . . . . .	737
Format der Parameter . . . . .	738
Fortsetzungszeilen bei Parametern . . . . .	738
Regelung für Groß- und Kleinbuchstaben . . . . .	739
Regelung für das Zeichen Zirkumflex . . . . .	739
I. Zahlenwerte . . . . .	740
II. Textteile . . . . .	740
III. Textteile-Vergleichs-Tabelle . . . . .	740
IV. Textteile-Austausch-Tabelle . . . . .	741
V. Zeichengruppen und Stringgruppen . . . . .	741
Vorrang bei Zeichen- und Stringgruppen . . . . .	742
Vordefinierte Zeichengruppen . . . . .	742
Definition von Zeichengruppen . . . . .	742
Definition von Stringgruppen . . . . .	743
VI. Zeichen-Tabelle . . . . .	744
VII. Sortieralphabet-Tabelle . . . . .	744
VIII. Vergleichs-Tabelle . . . . .	744
Besonderheiten der Parameterart VIII-a . . . . .	745
Besonderheiten der Parameterart VIII-b . . . . .	745
IX. Such-Tabelle . . . . .	746
Zahlenwertbedingungen . . . . .	747
Häufigkeitsbedingungen . . . . .	747
Verweise (in Suchzeichenfolgen) . . . . .	750
Umgebungsbedingungen . . . . .	751
X. Austausch-Tabelle . . . . .	752
Verweise (in Ersatzzeichenfolgen) . . . . .	753
XI. Sonstiges . . . . .	755
XII. Recherchier-Tabelle . . . . .	756
Besonderheiten der Parameterart XII-a . . . . .	757
Besonderheiten der Parameterart XII-b . . . . .	757



## Allgemeines

Mit Parametern werden die von den TUSTEP-Programmen gewünschten Leistungen genauer beschrieben. Ihre Funktion ist den einzelnen Programmbeschreibungen zu entnehmen.

Für die Parameter gibt es drei verschiedenen Konventionen. Nach welcher Konvention die Parameter interpretiert werden, kann mit dem Kommando `#PARAMETER,MODUS=...` (siehe Seite 207) eingestellt werden. Außerdem kann sie innerhalb von Parametern mit dem Parameter `PAR` festgelegt werden.

Die "alte" Konvention ist die ursprüngliche in TUSTEP. Sie wurde konzipiert, als die Lochkarte noch das Standardmedium für die Daten- und Programmeingabe war. Dabei musste berücksichtigt werden, dass auf Lochkarten üblicherweise Groß- und Kleinbuchstaben nicht unterschieden wurden und Umlaute nicht vorgesehen waren (`MODUS=ALT`).

Die "neue" Konvention ist auf neuere Eingabemedien abgestimmt, bei denen diese Einschränkungen nicht mehr bestehen. Zur Kennzeichnung von Zeichen- und Stringgruppen, Häufigkeitsangaben, Verweisen usw. werden dabei die spitzen Klammern verwendet (`MODUS=NEU`, gleichbedeutend mit `MODUS=<>`).

Die Mustererkennung (pattern matching) wurden nach und nach erweitert und verbessert. Dabei wurde bei den neu hinzukommenden Codierungen immer darauf geachtet, dass auch zuvor erstellte Parameter noch die gleiche Wirkung wie seither hatten. Das hat dazu geführt, dass die Codierungen unübersichtlich wurden und nicht mehr leicht zu merken und auch nicht mehr leicht zu lesen sind. Darüber hinaus ist die direkte Verwendung von spitzen Klammern nicht möglich, wenn Parameter mit einem XML-Editor geschrieben oder geändert werden sollen. Deshalb wurden die Angaben zur Mustererkennung neu konzipiert und damit eine dritte Konvention für die Parameter geschaffen, bei der die Codierungen leichter zu merken und auch leichter zu lesen sind. Zur Kennzeichnung von Zeichen- und Stringgruppen, Häufigkeitsangaben, Verweisen usw. werden dabei die geschweiften Klammern verwendet (`MODUS={ }`).

Für neue Projekte empfiehlt es sich, diese dritte Konvention zu verwenden. Parameter nach der dritten Konvention sind im Kapitel "`{ }`-Parameter" ab Seite 701 beschreiben.

Für den Editor und die TUSTEP-Programme mit Parametern ist diese dritte Konvention voreingestellt.

Um bei alten Projekten Aufwärtskompatibilität für Makros zu gewährleisten, ist für Makros (unabhängig von der Einstellung mit dem Kommando `#PARAMETER`) die zweite Konvention (`MODUS=<>`) voreingestellt. Mit der Makroanweisung `MODE` (siehe ab Seite 413) kann die Konvention, nach der Such- und Austausch-Tabellen in Makros interpretiert werden, explizit eingestellt werden.

In diesem Kapitel sind die Parameter nach der "alten" und nach der "neuen" Konvention beschrieben. Die Stellen, die nur für die alte bzw. nur für die neue Konvention gelten, sind mit `ALT` bzw. `NEU` gekennzeichnet.

## Format der Parameter

- Spalte 1–3: Parameterkennung aus 1 bis 3 Zeichen, linksbündig. Parameter, die in den Spalten 1–3 Leerzeichen enthalten, gelten als Kommentar. Sie können an jeder beliebigen Stelle vorkommen.
- 4–5: leer: Parameter wird ausgewertet
- " +n ": Parameter wird nur ausgewertet, wenn der Wahlschalter n (n = 1 bis 7) gesetzt ist
- " n ": gleichbedeutend mit "+n"
- " -n ": Parameter wird nur ausgewertet, wenn der Wahlschalter n (n = 1 bis 7) nicht gesetzt ist.
- 6–7: leer, oder rechtsbündig eine Zahl (vom jeweiligen Programm abhängig)
- 8: leer
- 9: leer Normalfall
- " :« wenn das Informationsfeld in Spalte 11 mit Leerzeichen (z. B. bei Parameterart VII) beginnt
- " =« wenn im Informationsfeld auf einen vorhergehenden Parameter gleicher Art verwiesen wird, dessen Informationsfeld übernommen werden soll. Spalte 11–13 muss die Parameterkennung und ggf. Spalte 16–17 die in Spalte 6–7 angegebene Nummer des Parameters, auf den verwiesen wird, enthalten.
- 10: leer
- ab 11: Informationsfeld. Abschließende Leerzeichen werden ignoriert.

## Fortsetzungszeilen bei Parametern

Die Informationen für einen Parameter können auf mehrere Zeilen aufgeteilt werden. Bei allen Zeilen eines Parameters müssen die Spalten 1 bis 3 und 6 bis 7 identisch sein.

Bei Parametern mit Begrenzungszeichen muss in Fortsetzungszeilen das gleiche Begrenzungszeichen verwendet werden. Ein Begrenzungszeichen in Spalte 11 einer Fortsetzungszeile und ein Begrenzungszeichen am Ende der vorhergehenden Zeile gelten zusammen als nur ein Begrenzungszeichen.

Der besseren Lesbarkeit wegen empfiehlt es sich, diese Eigenschaft auszunutzen und die Zeileneinteilung so zu wählen, dass jedes Informationsfeld mit einem Begrenzungszeichen anfängt (Spalte 11) und mit einem Begrenzungszeichen endet (letzte verwendete Spalte).

Wird eine Zeichenfolge in der nächsten Zeile fortgesetzt, so darf jedoch weder am Ende der Zeile noch in Spalte 11 der nächsten Zeile ein Begrenzungszeichen stehen. In diesem Fall ist zu beachten, dass Leerzeichen am Zeilenende ignoriert werden.

## Regelung für Groß- und Kleinbuchstaben

a) bei den Parameterarten II bis IV:

NEU: Groß- und Kleinbuchstaben werden als solche interpretiert.

ALT: Die Zeichen "<" bzw. ">" werden als Bereichsumschaltzeichen für Groß- und Kleinschreibung interpretiert. Die jeweilige Umschaltung gilt bis zum folgenden Umschaltzeichen bzw. bis zum Ende des Parameters. Es ist ohne Bedeutung, ob die Buchstaben groß oder klein geschrieben sind. Steht am Anfang kein Umschaltzeichen, so werden die Buchstaben bis zum ersten Umschaltzeichen in Kleinbuchstaben umgewandelt. Nach "<" werden die Buchstaben in Großbuchstaben umgewandelt und nach ">" werden die Buchstaben in Kleinbuchstaben umgewandelt. Sollen die Zeichen "<" und ">" selbst dargestellt werden, so ist dafür "<<" bzw. ">>" zu schreiben.

b) bei den Parameterarten V bis IX und XII:

Sind im Informationsfeld Buchstaben angegeben, so ist jeweils der entsprechende Groß- und Kleinbuchstabe gemeint (gleichgültig, ob der Buchstabe groß oder klein geschrieben ist). Soll zwischen Groß- und Kleinbuchstaben unterschieden werden, so ist der jeweilige Buchstabe mit einem vorangesetzten "<" bzw. ">" zu kennzeichnen. Es bedeutet also "a" und "A" ein großes oder kleines a, ">a" und ">A" ein kleines a, "<a" und "<A" ein großes a. Sollen die Zeichen "<" und ">" selbst dargestellt werden, so ist dafür "<<" bzw. ">>" zu schreiben.

c) bei der Parameterart X:

Für Suchzeichenfolgen gilt das Gleiche wie für Parameterart IX.

NEU: In Ersatzzeichenfolgen werden Groß- und Kleinbuchstaben als solche interpretiert, es sei denn, dass durch ein vor dem Buchstaben stehendes Zeichen ">" bzw. "<" ausdrücklich der entsprechende Klein- bzw. Großbuchstabe verlangt wird.

ALT: In Ersatzzeichenfolgen werden Groß- und Kleinbuchstaben als Kleinbuchstaben interpretiert, es sei denn, dass sie durch ein vorangesetztes "<" einzeln als Großbuchstaben gekennzeichnet sind. Eine Kennzeichnung von Kleinbuchstaben durch ">" ist möglich, aber wirkungslos.

## Regelung für das Zeichen Zirkumflex

NEU: Ein Zirkumflex "^" (eingegeben als "^") wird als normales Zeichen interpretiert.

ALT: Ein Zirkumflex "^" (eingegeben als "^") wirkt als Steuerzeichen und wird zusammen mit dem unmittelbar nachfolgenden Zeichen als ein einziges Zeichen interpretiert (vgl. die Tabellen Seite 763 und Seite 815).

Bei der Kennzeichnung eines mit "^" eingegebenen Buchstabens mit "<" oder ">" als Groß- oder Kleinbuchstabe ist die Reihenfolge zu beachten: z. B. ">^a" (nicht ">a") für ein kleines "ä".

## Parameterart I: Zahlenwerte

Im Informationsfeld werden in arabischen Ziffern geschriebene Zahlenwerte angegeben. Unmittelbar hintereinander stehende Ziffern werden als eine Zahl interpretiert. Die einzelnen Zahlen können durch beliebige Zeichen (außer Ziffern) voneinander getrennt werden. Für die Zahlenwerte muss die in der jeweiligen Beschreibung angegebene Reihenfolge eingehalten werden; dabei darf kein Zahlenwert ausgelassen werden. Sind weniger Zahlen angegeben als vom jeweiligen Programm erwartet werden, so werden für die fehlenden Zahlen die vom jeweiligen Programm voreingestellten Zahlenwerte ergänzt.

Reicht für die Angaben eine Zeile nicht aus, so können Fortsetzungszeilen (mit der gleichen Parameterkennung) verwendet werden. An jedem Zeilenende wird ein Leerzeichen ergänzt.

Bei dieser Parameterart darf in Spalte 9 nur ein Leerzeichen stehen, d. h. es kann nicht auf einen vorhergehenden Parameter verwiesen werden.

## Parameterart II: Textteile

Im Informationsfeld werden Textteile angegeben, die durch ein frei wählbares Begrenzungszeichen voneinander getrennt sind. Das Begrenzungszeichen ist das erste Zeichen (Spalte 11) im Informationsfeld des Parameters. Der letzte Textteil muss durch ein Begrenzungszeichen abgeschlossen sein.

Sind weniger Textteile angegeben als vom jeweiligen Programm erwartet werden, so werden vom jeweiligen Programm voreingestellte Textteile ergänzt.

Reicht für die Angaben eine Zeile nicht aus, so können Fortsetzungszeilen (mit der gleichen Parameterkennung und dem gleichen Begrenzungszeichen; vgl. Seite 738) verwendet werden. Wird dabei ein Textteil in der nächsten Zeile fortgesetzt, so kann ein "-" als Silbentrennzeichen verwendet werden, falls es nicht als Begrenzungszeichen gewählt wurde. Ein "-" am Zeilenende gilt dann jedoch nur als Silbentrennzeichen, wenn das zweitletzte Zeichen ebenfalls ein "-" ist, oder wenn das zweitletzte Zeichen ein Buchstabe und das drittletzte Zeichen kein Steuerzeichen (\$, &, @, \, \_, #, %) ist.

## Parameterart III: Textteile-Vergleichs-Tabelle

Im Informationsfeld werden Textteile angegeben, die durch ein frei wählbares Begrenzungszeichen voneinander getrennt sind. Das Begrenzungszeichen ist das erste Zeichen (Spalte 11) im Informationsfeld des Parameters. Der letzte Textteil muss durch ein Begrenzungszeichen abgeschlossen sein.

Ob Groß- und Kleinschreibung für das Programm relevant ist, ist bei der jeweiligen Parameterbeschreibung angegeben; die Angabe von Zeichen- oder Stringgruppen (s. u.) ist nicht möglich.

Reicht für die Angaben eine Zeile nicht aus, so können Fortsetzungszeilen (mit der gleichen Parameterkennung und dem gleichen Begrenzungszeichen; vgl. Seite 738) verwendet werden.

## Parameterart IV: Textteile-Austausch-Tabelle

Im Informationsfeld werden Paare von Textteilen angegeben, deren jeweils erster Textteil zum Vergleich mit dem zu verarbeitenden Textteil dient, der bei Übereinstimmung durch den jeweils zweiten Textteil ersetzt werden soll. Vergleichs- und Ersatz-Textteil dürfen verschieden lang sein.

Die im Informationsfeld angegebenen Textteile werden durch ein frei wählbares Begrenzungszeichen voneinander getrennt. Das Begrenzungszeichen ist das erste Zeichen (Spalte 11) im Informationsfeld des Parameters. Der letzte Textteil muss durch ein Begrenzungszeichen abgeschlossen sein.

Ob in den Vergleichs-Textteilen Groß- und Kleinschreibung für das Programm relevant ist, ist bei der jeweiligen Parameterbeschreibung angegeben; die Angabe von Zeichen- oder Stringgruppen (s. u.) ist nicht möglich.

Reicht für die Angaben eine Zeile nicht aus, so können Fortsetzungszeilen (mit der gleichen Parameterkennung und dem gleichen Begrenzungszeichen; vgl. Seite 738) verwendet werden.

## Parameterart V: Zeichengruppen und Stringgruppen

*(Gilt für "<>-Konvention", für "{}-Konvention" siehe Seite 709.)*

Eine Zeichengruppe ist eine Zusammenfassung von einzelnen Zeichen, auf die (nachdem sie definiert ist) im selben Programm in nachfolgenden Parametern der Parameterarten VI bis X und XII (in X nur in Suchzeichenfolgen) durch Angabe der dazugehörigen Gruppenkennung Bezug genommen werden kann. Die Zeichen der angegebenen Zeichengruppe werden dann so behandelt, als stünden sie alle gleichwertig an Stelle der Gruppenkennung.

Eine Stringgruppe ist eine Zusammenfassung von Zeichenfolgen (Strings), auf die (nachdem sie definiert ist) im selben Programm in nachfolgenden Parametern der Parameterarten IX und X (in X nur in Suchzeichenfolgen) durch Angabe der dazugehörigen Gruppenkennung Bezug genommen werden kann. Die Zeichenfolgen der angegebenen Stringgruppe werden dann so behandelt, als stünden sie alle an Stelle der Gruppenkennung.

Zur Definition von Zeichen- und Stringgruppen und zur Bezugnahme auf die so definierten Gruppen stehen für Zeichengruppen Gruppenkennungen der Form ">[xy]" und für Stringgruppen Gruppenkennungen der Form "<[xy]" (jeweils ohne Anführungszeichen) zur Verfügung. Dabei ist xy ein aus zwei Zeichen bestehender Name, wobei x ein Buchstabe und y eine Buchstabe oder eine Ziffer sein muss. Groß- und Kleinschreibung wird nicht unterschieden.

Daneben sind noch Gruppenkennungen für Zeichen- und Stringgruppen der Form ">n" und "<n" (ohne Anführungszeichen) möglich, wobei n jeweils durch eine Ziffer zu ersetzen ist. Es können auf diese Weise also bis zu 20 Zeichengruppen und bis zu 20 Stringgruppen zusätzlich definiert werden.

Die Definition einer Gruppe gilt jeweils für alle nachfolgenden Parameter eines Kommandos, bis diese Gruppe neu definiert wird.

Bei dieser Parameterart darf in Spalte 9 nur ein Leerzeichen oder ein Doppelpunkt stehen, d. h. es kann nicht auf einen vorhergehenden Parameter verwiesen werden.

### Vorrang bei Zeichen- und Stringgruppen

Kommt in der Definition einer Zeichen- oder Stringgruppe eine Gruppenkennung der Form ">n" oder "<n" vor, so wird diese immer als Kennung für die entsprechende Zeichengruppe interpretiert (auch wenn zur gleichen Kennung eine Stringgruppe definiert ist). Dies gilt auch für die Parameterarten VI bis VIII und XII. Bei den Parameterarten IX und X wird eine Gruppenkennung der Form ">n" oder "<n" als Kennung der entsprechenden Stringgruppe interpretiert, falls diese definiert ist, andernfalls als Kennung der entsprechenden Zeichengruppe.

### Vordefinierte Zeichengruppen

Außer Zeichengruppen, die selbst definiert werden können, gibt es acht intern vordefinierte Zeichengruppen mit folgenden Zeichengruppen-Kennungen:

- >\* alle Kleinbuchstaben des TUSTEP-Zeichensatzes
- <\* alle Großbuchstaben des TUSTEP-Zeichensatzes
- >/ alle Ziffern des TUSTEP-Zeichensatzes
- </ alle Buchstaben des TUSTEP-Zeichensatzes
- >% alle Zeichen des ASCII-Zeichensatzes
- <% alle Zeichen des TUSTEP-Zeichensatzes
- >@ alle Zeichen, die zur Akzent-Codierung nach % stehen können
- <@ alle Sonderzeichen des TUSTEP-Zeichensatzes einschließlich Leerzeichen

### Definition von Zeichengruppen

Als Parameterkennung für die Definition einer Zeichengruppe ist bei Gruppenkennungen der Form "> [xy]" in Spalte 1 bis 3 die Gruppenkennung ohne die eckigen Klammern zu schreiben; bei Gruppenkennungen der Form ">n" und "<n" ist in Spalte 1 und 2 die Gruppenkennung und in Spalte 3 ein "z" zu schreiben.

Im Informationsfeld werden die Zeichen (ab Spalte 11, ohne Zwischenraum) angegeben, die zu der zu definierenden Gruppe gehören sollen. An Stelle eines Zeichens kann auch eine Kennung einer vordefinierten Zeichengruppe angegeben werden. Innerhalb von Parameterangaben (aber nicht in Editoranweisungen) kann an Stelle eines Zeichens auch eine Kennung einer zuvor selbst definierten Zeichengruppe angegeben werden.

Das Informationsfeld wird von links nach rechts abgearbeitet. Dabei werden die

Zeichen in die (zu Anfang leere) Gruppe eingetragen. Die Zeichenfolge "><" bewirkt, dass die nachfolgenden Zeichen wieder aus der Gruppe gelöscht werden. Mit der Zeichenfolge "<>" kann die Wirkung von "><" wieder aufgehoben werden, d. h. die nachfolgenden Zeichen werden wieder zur Gruppe hinzugefügt.

## Definition von Stringgruppen

Als Parameterkennung für die Definition einer Stringgruppe ist bei Gruppenkennungen der Form "< [xy]" in Spalte 1 bis 3 die Gruppenkennung ohne die eckigen Klammern zu schreiben; bei Gruppenkennungen der Form ">n" und "<n" ist in Spalte 1 und 2 die Gruppenkennung und in Spalte 3 ein "s" zu schreiben.

Im Informationsfeld werden die Zeichenfolgen angegeben, die zu der zu definierenden Stringgruppe gehören sollen. An Stelle eines einzelnen Zeichens einer Zeichenfolge kann die Kennung einer zuvor definierten Zeichengruppe angegeben werden; Kennungen von Stringgruppen sind hier nicht vorgesehen.

Die im Informationsfeld angegebenen Zeichenfolgen werden durch ein frei wählbares Begrenzungszeichen voneinander getrennt. Das Begrenzungszeichen ist das erste Zeichen (Spalte 11) im Informationsfeld des Parameters. Die letzte Zeichenfolge muss durch ein Begrenzungszeichen abgeschlossen sein.

Es können zwei Arten von Zeichenfolgen angegeben werden, nämlich "Suchzeichenfolgen" und "Ausnahmezeichenfolgen". Sie werden durch zwei aufeinander folgende Begrenzungszeichen voneinander getrennt. Durch Angabe von zwei aufeinander folgenden Begrenzungszeichen kann beliebig oft zwischen Suchzeichenfolgen und Ausnahmezeichenfolgen gewechselt werden.

Entspricht eine im Text ab der jeweils beim Suchen erreichten Position stehende Zeichenfolge mehreren Such- und/oder Ausnahmezeichenfolgen, so hat die jeweils längere Zeichenfolge den Vorrang; bei gleich langen Zeichenfolgen entspricht ihre Rangfolge der Reihenfolge ihrer Angabe. Wird eine Ausnahmezeichenfolge gefunden, so hat dies die gleiche Wirkung, wie wenn keine der in der Stringgruppe enthaltenen Zeichenfolgen gefunden worden wäre.

Achtung: Auf Grund dieser Vorgehensweise sind in Stringgruppen nur solche Ausnahmezeichenfolgen sinnvoll, die im Text auf der gleichen Position beginnen können wie eine in der gleichen Stringgruppe angegebene Suchzeichenfolge (z. B. "/xy/xy\*/", aber nicht "/xy/\*\*xy/"). Außerdem ist bei gleich langen Such- und Ausnahmezeichenfolgen auf die Reihenfolge zu achten. So ist z. B. die Angabe "/>\*>\*/xy/" nicht sinnvoll, weil die Zeichenfolgen ">\*>\*" und "xy" beide zwei Zeichen lang sind (">\*" gilt als ein Zeichen, weil es stellvertretend für einen Kleinbuchstaben steht), und bei der Überprüfung der im Text stehenden Zeichenfolge "xy" die Suchzeichenfolge ">\*>\*" zum Zuge kommt, weil sie vor der Ausnahmezeichenfolge "xy" angegeben ist. Damit in diesem Fall die Ausnahmezeichenfolge wirkt, muss "/xy/ />\*>\*/" angegeben werden.

Reicht für die Angaben eine Zeile nicht aus, so können Fortsetzungszeilen (mit der gleichen Parameterkennung und dem gleichen Begrenzungszeichen; vgl. Seite 738) verwendet werden.

Wenn am Anfang eines Informationsfeldes eine Ausnahmezeichenfolge angegeben

werden soll, kann mit zwei aufeinander folgenden Begrenzungszeichen am Anfang des Informationsfeldes von Suchzeichenfolge auf Ausnahmezeichenfolge umgeschaltet werden.

Um Fehler zu vermeiden, empfiehlt sich folgendes Verfahren:

Wenn am Ende einer Zeile eine Ausnahmezeichenfolge angegeben ist, sollte diese Zeile mit zwei aufeinander folgenden Begrenzungszeichen am Ende des Informationsfeldes abgeschlossen werden und damit von Ausnahmezeichenfolgen auf Suchzeichenfolgen umgeschaltet werden.

## Parameterart VI: Zeichen-Tabelle

*(Gilt für "<>-Konvention", für "{}-Konvention" siehe Seite 712.)*

Im Informationsfeld werden die Zeichen (ab Spalte 11, ohne Zwischenraum) angegeben, die in die Tabelle aufgenommen werden sollen. Wenn in der jeweiligen Beschreibung der einzelnen Parameter nichts anderes angegeben ist, ist die Reihenfolge der angegebenen Zeichen ohne Bedeutung.

An Stelle eines Zeichens kann auch eine Kennung einer zuvor definierten Zeichengruppe angegeben werden. Die Zeichen der angegebenen Zeichengruppe werden so behandelt, als stünden sie alle an Stelle der Zeichengruppen-Kennung.

## Parameterart VII: Sortieralphabet-Tabelle

*(Gilt für "<>-Konvention", für "{}-Konvention" siehe Seite 712.)*

Das Informationsfeld wird wie bei einer Zeichen-Tabelle interpretiert, jedoch ist die Reihenfolge der einzelnen Zeichen immer von Bedeutung. Die Reihenfolge legt die Wertigkeit der Zeichen beim Sortieren bzw. beim Vergleichen fest; d. h. das erste Zeichen (bzw. alle Zeichen der auf erster Zeichenposition angegebenen Zeichengruppe) hat den niedrigsten Wert, dann folgt das zweite (bzw. die Zeichen der entsprechenden Gruppe) usw. Hinter der Zeichenkombination "><" können Zeichen (und Zeichengruppen) angegeben werden, die die höchsten Wertigkeiten erhalten sollen, also z. B. beim Sortieren ans Ende sortiert werden sollen. Werden nicht alle Zeichen des Zeichenvorrats angegeben, so werden die fehlenden Zeichen in der Reihenfolge der Standard-Sortierfolge (siehe Seite Seite 841) an der mit "><" gekennzeichneten Stelle bzw. (falls "><" fehlt) nach den angegebenen Zeichen logisch ergänzt.

## Parameterart VIII: Vergleichs-Tabelle

*(Gilt für "<>-Konvention", für "{}-Konvention" siehe Seite 713.)*

Im Informationsfeld werden Zeichenfolgen angegeben, nach denen gesucht werden soll. An Stelle eines einzelnen Zeichens einer Zeichenfolge kann die Kennung einer vordefinierten oder einer zuvor selbst definierten Zeichengruppe angegeben werden.



Eine Kennung der Form ">n" oder "<n" wird bei dieser Parameterart immer als Kennung einer Zeichengruppe gewertet, auch wenn eine Stringgruppe mit der gleichen Kennung definiert ist.

Die im Informationsfeld angegebenen Zeichenfolgen werden durch ein frei wählbares Begrenzungszeichen voneinander getrennt. Das Begrenzungszeichen ist das erste Zeichen (Spalte 11) im Informationsfeld des Parameters. Die letzte Zeichenfolge muss durch ein Begrenzungszeichen abgeschlossen sein.

Es können zwei Arten von Zeichenfolgen angegeben werden, nämlich "Suchzeichenfolgen" und "Ausnahmezeichenfolgen". Sie werden durch zwei aufeinander folgende Begrenzungszeichen voneinander getrennt. Durch Angabe von zwei aufeinander folgenden Begrenzungszeichen kann beliebig oft zwischen Suchzeichenfolgen und Ausnahmezeichenfolgen gewechselt werden.

Achtung: Bei gleich langen Such- und Ausnahmezeichenfolgen ist auf die Reihenfolge zu achten. So ist z. B. die Angabe "/>\*>\*/xy/" nicht sinnvoll, weil die Zeichenfolgen ">\*>\*" und "xy" beide zwei Zeichen lang sind (">\*" gilt als ein Zeichen, weil es stellvertretend für einen Kleinbuchstaben steht), und bei der Überprüfung mit dem Textanfang "xy" die Suchzeichenfolge ">\*>\*" zum Zuge kommt, weil sie vor der Ausnahmezeichenfolge "xy" angegeben ist. Damit in diesem Fall die Ausnahmezeichenfolge wirkt, muss "//xy//>\*>\*" angegeben werden.

Reicht für die Angaben eine Zeile nicht aus, so können Fortsetzungszeilen (mit der gleichen Parameterkennung und dem gleichen Begrenzungszeichen; vgl. Seite 738) verwendet werden.

### Besonderheiten der Parameterart VIII-a

Die Texte werden nur daraufhin überprüft, ob sie mit einer der angegebenen Zeichenfolgen beginnen. Entspricht dabei der Textanfang mehreren Such- und/oder Ausnahmezeichenfolgen, so hat die jeweils längere Zeichenfolge den Vorrang; bei gleich langen Zeichenfolgen entspricht ihre Rangfolge der Reihenfolge ihrer Angabe. Wird (unter Berücksichtigung der genannten Rangfolge) eine Übereinstimmung mit einer Ausnahmezeichenfolge festgestellt, so hat dies die gleiche Wirkung, wie wenn keine der angegebenen Zeichenfolgen mit dem Textanfang übereinstimmen würde.

### Besonderheiten der Parameterart VIII-b

Die Texte werden nur daraufhin überprüft, ob sie mit einer der angegebenen Zeichenfolgen enden. Entspricht dabei das Textende mehreren Such- und/oder Ausnahmezeichenfolgen, so hat die jeweils längere Zeichenfolge den Vorrang; bei gleich langen Zeichenfolgen entspricht ihre Rangfolge der Reihenfolge ihrer Angabe. Wird (unter Berücksichtigung der genannten Rangfolge) eine Übereinstimmung mit einer Ausnahmezeichenfolge festgestellt, so hat dies die gleiche Wirkung, wie wenn keine der angegebenen Zeichenfolgen mit dem Textende übereinstimmen würde.

## Parameterart IX: Such-Tabelle

*(Gilt für "<>-Konvention", für "{}-Konvention" siehe Seite 714.)*

Im Informationsfeld werden Zeichenfolgen angegeben, nach denen gesucht werden soll. An Stelle eines einzelnen Zeichens einer Zeichenfolge kann die Kennung einer zuvor definierten Zeichen- bzw. Stringgruppe, eine Zahlenwertbedingung oder ein Verweis auf andere Elemente (s. u.) der gleichen Zeichenfolge angegeben werden. Darüber hinaus können Zeichen, Zeichengruppen, Stringgruppen, Zahlenwertbedingungen sowie Verweise mit Häufigkeitsbedingungen versehen werden und können Umgebungsbedingungen für Zeichenfolgen angegeben werden.

Die im Informationsfeld angegebenen Zeichenfolgen werden durch ein frei wählbares Begrenzungszeichen voneinander getrennt. Das Begrenzungszeichen ist das erste Zeichen (Spalte 11) im Informationsfeld des Parameters. Die letzte Zeichenfolge muss durch ein Begrenzungszeichen abgeschlossen sein.

Es können zwei Arten von Zeichenfolgen angegeben werden, nämlich "Suchzeichenfolgen" und "Ausnahmezeichenfolgen" (das sind Zeichenfolgen, die beim Durchsuchen des Textes übergangen werden sollen). Sie werden durch zwei aufeinander folgende Begrenzungszeichen voneinander getrennt. Durch Angabe von zwei aufeinander folgenden Begrenzungszeichen kann beliebig oft zwischen Suchzeichenfolgen und Ausnahmezeichenfolgen gewechselt werden.

Die Texte werden jeweils von links nach rechts durchsucht. Entspricht dabei eine im Text ab der jeweils beim Suchen erreichten Position stehende Zeichenfolge mehreren Such- und/oder Ausnahmezeichenfolgen, so hat die jeweils längere Zeichenfolge den Vorrang; bei gleich langen Zeichenfolgen entspricht ihre Rangfolge der Reihenfolge ihrer Angabe. Wird beim Durchsuchen (unter Berücksichtigung der genannten Rangfolge) eine Ausnahmezeichenfolge gefunden, so werden die dazugehörigen Zeichen im Text übersprungen; im Text wird dann ab der auf die Ausnahmezeichenfolge folgende Position weitergesucht. Für Ausnahmezeichenfolgen, die in Stringgruppen angegeben sind, gilt dies jedoch nicht; sie bewirken nur, dass andere Zeichenfolgen, die in der gleichen Stringgruppe angegeben sind, ggf. nicht zum Zuge kommen und somit ab der aktuellen Position im Text keine der in der Stringgruppe angegebene Suchzeichenfolge gefunden wird.

Achtung: Entsprechend dieser Vorgehensweise beim Durchsuchen des Textes nach angegebenen Zeichenfolgen wird z. B. beim Durchsuchen des Textes "...01234..." nach den Zeichenfolgen "/1234/01/" die Zeichenfolge "01" und nicht die Zeichenfolge "1234" gefunden. Grund: Die Zeichenfolge "1234" ist zwar länger als die Zeichenfolge "01", aber entscheidend ist in diesem Fall, dass die Zeichenfolge "01" weiter links im Text steht und deshalb beim Durchsuchen von links nach rechts zuerst gefunden wird. Ebenso wird z. B. beim Durchsuchen des Textes "...01234..." nach den Zeichenfolgen "/1234//01/" zuerst die Ausnahmezeichenfolge "01" gefunden; dann wird im Text ab der Position, die auf die Zeichenfolge "01" folgt, weitergesucht und somit die Zeichenfolge "1234" nicht mehr gefunden. Außerdem ist bei gleich langen Such- und Ausnahmezeichenfolgen auf die Reihenfolge zu achten. So ist z. B. die Angabe "/>\*>\*/xy/" nicht sinnvoll, weil die Zeichenfolgen ">\*>\*" und "xy" beide zwei Zeichen lang sind (">\*" gilt als ein

Zeichen, weil es stellvertretend für einen Kleinbuchstaben steht), und bei der Überprüfung der im Text stehenden Zeichenfolge "xy" die Suchzeichenfolge ">\*>\*" zum Zuge kommt, weil sie vor der Ausnahmezeichenfolge "xy" angegeben ist. Damit in diesem Fall die Ausnahmezeichenfolge wirkt, muss "//xy//>\*>\*/" angegeben werden.

Reicht für die Angaben eine Zeile nicht aus, so können Fortsetzungszeilen (mit der gleichen Parameterkennung und dem gleichen Begrenzungszeichen; vgl. Seite 738) verwendet werden.

Zeichen, Zeichengruppen-Kennung, Stringgruppen-Kennung, Zahlenwertbedingung sowie Verweis werden im folgenden "Element" der Suchzeichenfolge genannt. Eine vor einem Zeichen, einer Zeichengruppen-Kennung, einer Stringgruppen-Kennung sowie einem Verweis stehende Häufigkeitsbedingung gehört mit zum Element.

## Zahlenwertbedingungen

Eine Zahlenwertbedingung steht stellvertretend für eine vorzeichenlose Zahl (aus arabischen Ziffern), die eine bestimmte Bedingung bezüglich ihres Wertes erfüllen muss. Im zu durchsuchenden Text darf weder vor noch nach einer solchen Zahl eine weitere Ziffer stehen.

<{n}	Zahl kleiner als n.
<{n-m}	Zahl kleiner als n oder größer als m.
<={n}	Zahl kleiner gleich n.
<={n-m}	Zahl kleiner gleich n oder größer gleich m.
>{n}	Zahl größer als n.
>{n-m}	Zahl größer als n und kleiner als m.
>={n}	Zahl größer gleich n.
>={n-m}	Zahl größer gleich n und kleiner gleich m.

Eine Zahlenwertbedingung darf nur mit der Häufigkeitsbedingung ><0 versehen werden, andere sind überflüssig oder sinnlos.

## Häufigkeitsbedingungen

><n	Mindesthäufigkeit: Enthält ein Element die Häufigkeitsbedingung "><n", so müssen im zu durchsuchenden Text die Zeichen, die dem im Element angegebenen Zeichen (der durch die Gruppenkennung angegebenen Gruppe, dem angegebenen Verweis) entsprechen, mindestens n-mal unmittelbar hintereinander vorkommen. Alle n Vorkommen dieser Zeichen werden diesem Element zugeordnet.
-----	---

"n" ist eine einstellige Zahl und gibt die gewünschte Mindesthäufigkeit an. Soll angegeben werden, dass die dem Element entsprechenden Zeichen im Text ganz fehlen dürfen, so muss n = 0 sein. Diese

Ziffer "0" kann weggelassen werden, wenn dadurch keine Verwechslung entstehen kann (d. h. wenn nicht unmittelbar auf diese Häufigkeitsangabe eine Ziffer folgt).

Fehlt diese Angabe für die Mindesthäufigkeit, so wird 1 als Mindesthäufigkeit angenommen.

Sollen einem mit der Mindesthäufigkeit  $n$  versehenen Element mehr als  $n$  Vorkommen (mehr als 1 Vorkommen, falls für  $n$  0 angegeben ist) der dem Element entsprechenden Zeichen zugeordnet werden, so muss zu diesem Element auch eine Maximalhäufigkeit angegeben werden.

<> $n$ 

Maximalhäufigkeit: Enthält ein Element die Häufigkeitsbedingung "<> $n$ " und kommen im zu durchsuchenden Text die Zeichen, die dem im Element angegebenen Zeichen (der durch die Gruppenkennung angegebenen Gruppe, dem angegebenen Verweis) entsprechen, mehrmals unmittelbar hintereinander vor, so werden bis zu  $n$  Vorkommen dieser Zeichen diesem Element zugeordnet.

Achtung: Die Angabe der Maximalhäufigkeit <> $n$  beinhaltet also nicht, dass im zu durchsuchenden Text die Zeichen, die dem im Element angegebenen Zeichen (der durch die Gruppenkennung angegebenen Gruppe, dem angegebenen Verweis) entsprechen, höchstens  $n$ -mal unmittelbar hintereinander vorkommen dürfen.

" $n$ " ist eine einstellige Zahl und gibt die gewünschte maximale Häufigkeit an. Soll angegeben werden, dass die im Element angegebenen Zeichen im Text beliebig oft unmittelbar hintereinander vorkommen dürfen, so muss  $n$  Null sein. Diese Ziffer "0" kann weggelassen werden, wenn dadurch keine Verwechslung entstehen kann (d. h. wenn nicht unmittelbar auf diese Häufigkeitsangabe eine Ziffer folgt).

Fehlt diese Angabe der maximalen Häufigkeit, so ist die maximale Häufigkeit gleich der mit ">< $n$ " angegebenen Mindesthäufigkeit bzw. ist 1, wenn als Mindesthäufigkeit "><0" oder wenn keine Mindesthäufigkeit angegeben ist.

Es darf nur jeweils eine Angabe für Mindesthäufigkeit und eine Angabe für maximale Häufigkeit vor einem Zeichen, einer Gruppenkennung oder einem Verweis stehen. Die Häufigkeitsbedingungen >< $n$  und <> $n$  können jedoch kombiniert werden; so bedeutet z. B. "><<>" soviel wie: kein Leerzeichen oder beliebig viele Leerzeichen. Soll als Mindesthäufigkeit und als maximale Häufigkeit der gleiche, von 0 verschiedene Wert  $n$  angegeben werden, so genügt die Angabe der Mindesthäufigkeit. Werden für  $n$  verschiedene Werte angegeben, so hat die Reihenfolge der Angabe folgende Bedeutung:

Ist die Bedingung für die Mindesthäufigkeit vor der Bedingung für die maximale Häufigkeit angegeben (oder ist nur die Bedingung für die maximale Häufigkeit angegeben), so wird beim Durchsuchen des Textes so früh wie möglich versucht, von dem mit den Häufigkeitsbedingungen versehenen Element auf das nächste Element der Suchzeichenfolge überzugehen, um eine der Suchzeichenfolge im Text entsprechen-

de Zeichenfolge zu finden, die auch den unmittelbar nachfolgenden Elementen der Suchzeichenfolge (bis zu einem Element, bei dem sich die angegebene Mindesthäufigkeit von der angegebenen maximalen Häufigkeit unterscheidet) entspricht.

> (n-m)      Entspricht der Angabe ><n<>m, jedoch kann für n und m auch eine zweistellige Zahl angegeben werden.

> (n)          Entspricht der Angabe > (n-n) .

Ist die Bedingung für die maximale Häufigkeit vor der Bedingung für die Mindesthäufigkeit angegeben, so wird beim Durchsuchen des Textes versucht, die Zeichen des Textes so lange als übereinstimmend mit dem mit solchen Häufigkeitsbedingungen versehenen Element gelten zu lassen, bis die angegebene maximale Häufigkeit erreicht ist.

< (n-m)      Entspricht der Angabe <>m<n, jedoch kann für n und m auch eine zweistellige Zahl angegeben werden.

< (n)          Entspricht der Angabe < (n-n) .

Wie oben beschrieben, hat beim Durchsuchen des Textes die längste einer Suchzeichenfolge entsprechende Zeichenfolge den Vorrang. Ist für ein Element eine Mindesthäufigkeit "><n" angegeben, so wird für die Berechnung der für die Rangfolge der Zeichenfolge maßgebenden Länge der Zeichenfolge dieses Element als aus n Zeichen bestehend gewertet.

Beispiele:

"/A<>\*B/" ist gleichbedeutend mit "/A<>0\*B/" und definiert eine Suchzeichenfolge, die aus "a", beliebig vielen (mindestens aber einem) "\*", und einem "b" (in dieser Reihenfolge) besteht.

"/A<<>\*B/" ist gleichbedeutend mit "/A<<0<>0\*B/" und definiert eine Suchzeichenfolge, die aus "a", keinem oder beliebig vielen "\*", und einem "b" (in dieser Reihenfolge) besteht.

":><2<>4>/:" definiert eine Suchzeichenfolge, die aus zwei bis vier Ziffern besteht. Mit dieser Suchzeichenfolge werden jedoch auch vier aufeinander folgende Ziffern gefunden, die Teil einer fünf- oder mehrstelligen Ziffernfolge sind. Soll dies ausgeschlossen werden, so muss entweder zusätzlich zur Suchzeichenfolge auch eine entsprechende Ausnahmezeichenfolge angegeben werden (also ":><2<>4>/ :><5<>>/:" ) oder durch Angabe der Zeichen, die links und rechts von der gesuchten Ziffernfolge stehen, die Anzahl der Ziffern begrenzt werden (z. B. ":[><2<>4>/] :").

Im Text stehe die Zeichenfolge "1230000"; der Suchzeichenfolge "><1<>5>/0" (und der damit gleichwertigen Suchzeichenfolge "<>5>/0") entspräche dann die Textzeichenfolge "1230", während der Suchzeichenfolge "<>5><1>/0" die Zeichenfolge "123000" entspräche.

Im Text stehe die Zeichenfolge "abxdxyzxyz."; den Suchzeichenfolgen "><2<>4</XYZ", "><2<>5</XYZ" usw. bis einschließlich "><2<>9</XYZ" entspräche dann die Textzeichenfolge "abxdxyz", während den Suchzeichenfolgen "<>9><2</XYZ" und "<>8><2</XYZ" in diesem Text nichts, der Suchzeichenfolge

"<>7<2</XYZ" die Textzeichenfolge "abxdxyzxyz", der Suchzeichenfolge  
 "<>6<2</XYZ" die Textzeichenfolge "bxdxyzxyz", der Suchzeichenfolge  
 "<>5<2</XYZ" die Textzeichenfolge "xdxyzxyz", der Suchzeichenfolge  
 "<>4<2</XYZ" die Textzeichenfolge "abxdxyz" entspräche.

### Verweise (in Suchzeichenfolgen)

>=nn Verweis auf das nn-te Element der Suchzeichenfolge: die Zeichen, die im zu durchsuchenden Text dem nn-ten Element der Suchzeichenfolge entsprechen, müssen auch an dieser Stelle im Text vorkommen; dabei werden Groß- und Kleinbuchstaben unterschieden. Es darf nur auf ein links vom Verweis stehendes Element verwiesen werden.

Für "nn" ist eine zweistellige Zahl (bei Zahlen unter 10 also mit einer führenden "0") einzusetzen.

>:nn Ist gleichbedeutend mit ">=nn", jedoch wird zwischen Groß- und Kleinbuchstaben nicht unterschieden.

<=nn Verweis auf das nn-letzte Element der Suchzeichenfolge: die Zeichen, die im zu durchsuchenden Text dem vom Ende der Zeichenfolge her gezählten nn-ten Element der Suchzeichenfolge entsprechen, müssen auch an dieser Stelle im Text vorkommen; dabei werden Groß- und Kleinbuchstaben unterschieden. Es darf nur auf ein links vom Verweis stehendes Element verwiesen werden.

Für "nn" ist eine zweistellige Zahl (bei Zahlen unter 10 also mit einer führenden "0") einzusetzen.

<:nn Ist gleichbedeutend mit "<=nn", jedoch wird zwischen Groß- und Kleinbuchstaben nicht unterschieden.

### Beispiele:

"|</>/>=01|" ist gleichwertig mit "|</>/<=03|" und bezeichnet eine aus 3 Zeichen bestehende Suchzeichenfolge, die zwei identische Buchstaben rechts und links von einer Ziffer enthält (z. B. a2a, aber nicht A2a oder a2A); sie muss nämlich bestehen aus einem beliebigen Buchstaben ("</"), einer Ziffer (">/") und einem weiteren Zeichen, das mit dem ersten (">=01") bzw. drittletzten ("<=03") Zeichen der Zeichenfolge im Text identisch ist, also dem gleichen Großbuchstaben bzw. dem gleichen Kleinbuchstaben wie vor der Ziffer.

"|<>>/:>=01|" bezeichnet eine Suchzeichenfolge, die rechts von einem Doppelpunkt die gleiche Ziffernfolge wie links von diesem Doppelpunkt enthält (z. B. 123:123, aber auch 23:23 von 123:234); die Zeichenfolge besteht aus beliebig vielen ("<>") Ziffern (">/"), einem Doppelpunkt (":") und der gleichen Ziffernfolge wie vor dem Doppelpunkt (">=01"), d. h. aus den gleichen Zeichen, die im durchsuchten Text dem ersten Element der Suchzeichenfolge entsprechen.

"|>={0}:>=01|" bezeichnet eine Suchzeichenfolge, die rechts von einem Doppelpunkt die gleiche Zahl wie links von diesem Doppelpunkt enthält (z. B. 123:123, aber auch 23:23 von 23:234, jedoch nicht 123:123 von 0123:123); die Zeichenfolge besteht aus einer Zahl größer gleich Null (">={0}"), einem Doppelpunkt (":")

und der gleichen Ziffernfolge wie vor dem Doppelpunkt (" $\geq 01$ "), genauer: aus den gleichen Zeichen, die im durchsuchten Text dem ersten Element der Suchzeichenfolge entsprechen.

## Umgebungsbedingungen

Mit Hilfe von Umgebungsbedingungen kann festgelegt werden, in welcher Umgebung eine Zeichenfolge im zu durchsuchenden Text stehen muss. Eine Suchzeichenfolge, zu der Umgebungsbedingungen angegeben sind, besteht aus einer "Kernzeichenfolge" (die einer Suchzeichenfolge ohne Umgebungsbedingungen entspricht) und einer linken und/oder einer rechten "Randzeichenfolge". Die der linken Randzeichenfolge entsprechende Zeichenfolge muss im zu durchsuchenden Text unmittelbar vor der der Kernzeichenfolge entsprechenden Textzeichenfolge stehen, die der rechten Randzeichenfolge entsprechende Zeichenfolge unmittelbar nach der der Kernzeichenfolge entsprechenden Textzeichenfolge.

Die linke Randzeichenfolge wird vor der Kernzeichenfolge angegeben und von dieser durch das unten beschriebene "Begrenzungszeichen für linken Rand" abgetrennt; die rechte Randzeichenfolge wird hinter der Kernzeichenfolge angegeben und von dieser durch das unten beschriebene "Begrenzungszeichen für rechten Rand" abgetrennt.

Für die Angabe der Randzeichenfolgen gelten die gleichen Regeln wie für die Angabe von Suchzeichenfolgen ohne Umgebungsbedingungen; bei der Angabe von Verweisen ist jedoch zu beachten, dass beim Zählen der Elemente die Elemente der Randzeichenfolgen mitgezählt werden, dass jedoch nur auf Elemente der Kernzeichenfolge verwiesen werden darf.

<|            Begrenzungszeichen für linken Rand der Kernzeichenfolge

>|            Begrenzungszeichen für rechten Rand der Kernzeichenfolge

Gleichbedeutend mit den zuvor aufgeführten Begrenzungszeichen können auch (noch) folgende verwendet werden:

><x          Begrenzungszeichen für linken Rand der Kernzeichenfolge

<>x          Begrenzungszeichen für rechten Rand der Kernzeichenfolge

Für "x" ist hier das für die ganze Tabelle gewählte (= das auf Spalte 11 der Parameterzeile stehende) Begrenzungszeichen einzusetzen.

Beispiele:

" : : > / < | 123 : 123 > | > / : : 123 : " definiert eine Such-Tabelle, mit der nach der Zahl 123 gesucht wird, wobei also z. B. die Ziffernfolge 123 in der Zahl 1234 nicht gefunden werden soll. Für die Angabe "beliebige Ziffer" wird die vordefinierte Zeichengruppe mit der Zeichengruppen-Kennung ">/" verwendet. Am Anfang wird mit zwei aufeinander folgenden Begrenzungszeichen " : : " auf Ausnahmezeichenfolgen umgeschaltet. Die erste Ausnahmezeichenfolge ist "> / < | 123 "; sie besagt, dass 123 nicht gefunden werden soll, wenn unmittelbar links davon eine Ziffer steht. Nach einem Begrenzungszeichen folgt die zweite Ausnahmezeichenfolge "123 > | > / "; sie besagt, dass 123 nicht gefunden werden soll, wenn unmittelbar rechts davon ein

Ziffer steht. Danach wird mit zwei aufeinander folgenden Begrenzungszeichen von Ausnahmezeichenfolgen auf Suchzeichenfolgen umgeschaltet. Jetzt folgt die zu suchende Zeichenfolge "123" und noch das abschließende Begrenzungszeichen. Würden die Ausnahmezeichenfolgen erst nach der Suchzeichenfolge angegeben (also ":123::>/<|123:123>|>/:"), so würden alle Ziffernfolgen 123 gefunden, da die Ausnahmezeichenfolgen wirkungslos blieben, weil sie gleich lang sind (die Zeichen der Randzeichenfolge zählen bei der Längenberechnung nicht mit) wie die Suchzeichenfolge und bei gleich langen Zeichenfolgen die Reihenfolge maßgebend ist, in der sie angegeben sind. Wird als Begrenzungszeichen für den linken bzw. den rechten Rand der Kernzeichenfolge die Form <>x bzw. ><x gewählt, so ergibt sich für die am Anfang dieses Abschnitts angeführte Suchzeichenfolge die Schreibweise ":>/><:123:123<>:>/:123:"

Anmerkung: Die gleiche Wirkung hätte die Such-Tabelle ":123::><4<>/:".

":>< <|UND>|>< : " bzw. ":>< ><:UND<>:>< : " definiert eine Suchzeichenfolge, mit der nach allen Zeichenfolgen "und" gesucht wird, die hinter einem Leerzeichen oder am linken Rand des abzusuchenden Textes und außerdem vor einem Leerzeichen oder am rechten Rand des abzusuchenden Textes stehen. Die Angabe ist wie folgt zu lesen: ":" = frei gewähltes Begrenzungszeichen für die Such-Tabelle, ">< " = ein oder kein Leerzeichen, "<|" bzw. "><:" = Begrenzungszeichen für linken Rand der Kernzeichenfolge (zusammen mit der davor stehenden Angabe bedeutet dies also: Unmittelbar links vor der Kernzeichenfolge muss ein Leerzeichen stehen, oder es darf, wenn es fehlt, nichts stehen; der linke Rand wäre im letzteren Fall gleichzeitig der Anfang des zu durchsuchenden Textes), "UND" = gesuchte Zeichenfolge (Kernzeichenfolge), ">|" bzw. "<>:" = Begrenzungszeichen für rechten Rand der Kernzeichenfolge, ">< " = ein oder kein Leerzeichen (zusammen mit der davor stehenden Angabe bedeutet dies also: Unmittelbar rechts nach der gesuchten Zeichenfolge muss ein Leerzeichen stehen, oder es darf, wenn es fehlt, nichts stehen; der rechte Rand wäre im letzteren Fall gleichzeitig das Ende des zu durchsuchenden Textes).

## Parameterart X: Austausch-Tabelle

(Gilt für "<>-Konvention", für "{}-Konvention" siehe Seite 721.)

Im Informationsfeld werden Zeichenfolgenpaare angegeben, deren jeweils erste Zeichenfolge die Suchzeichenfolge ist, die durch die jeweils zweite Zeichenfolge, die Ersatzzeichenfolge, ersetzt werden soll. Such- und Ersatzzeichenfolgen dürfen verschieden lang sein. Sie werden durch ein frei wählbares Begrenzungszeichen voneinander getrennt. Das Begrenzungszeichen ist das erste Zeichen (Spalte 11) im Informationsfeld des Parameters. Die letzte Zeichenfolge muss durch ein Begrenzungszeichen abgeschlossen werden.

Außer Zeichenfolgenpaaren können Ausnahmezeichenfolgen (das sind Zeichenfolgen, die beim Austauschen übergangen werden sollen und für die deshalb keine Ersatzzeichenfolgen existieren) angegeben werden. Das Umschalten von Zeichenfolgenpaaren auf Ausnahmezeichenfolgen geschieht dadurch, dass an einer Stelle,



an der eine Suchzeichenfolge erwartet wird, nochmals ein Begrenzungszeichen steht. Daran anschließend können eine oder mehrere Ausnahmezeichenfolgen angegeben werden. Das Umschalten von Ausnahmezeichenfolgen auf Zeichenfolgenpaare geschieht dadurch, dass an einer Stelle, an der eine Ausnahmezeichenfolge erwartet wird, nochmals ein Begrenzungszeichen steht. Daran anschließend können wieder Zeichenfolgenpaare angegeben werden. Auf diese Weise kann beliebig oft zwischen der Angabe von Zeichenfolgenpaaren und Ausnahmezeichenfolgen gewechselt werden.

Reicht für die Angaben eine Zeile nicht aus, so können Fortsetzungszeilen Leerzeichen (mit der gleichen Parameterkennung und dem gleichen Begrenzungszeichen; vgl. Seite 738) verwendet werden.

Für die Angabe der Suchzeichenfolgen und der Ausnahmezeichenfolgen gelten die gleichen Regeln wie bei der Such-Tabelle (Parameterart IX).

NEU: In den Ersatzzeichenfolgen werden Groß- und Kleinbuchstaben als solche interpretiert.

ALT: In den Ersatzzeichenfolgen werden die Buchstaben als Kleinbuchstaben interpretiert, wenn sie nicht durch ein vorangesetztes "<" einzeln als Großbuchstaben gekennzeichnet sind. Eine Kennzeichnung von Kleinbuchstaben durch ">" ist möglich, aber wirkungslos.

### Verweise (in Ersatzzeichenfolgen)

In den Ersatzzeichenfolgen können an Stelle einzelner Zeichen Verweise auf Elemente der dazugehörenden Suchzeichenfolge (einschließlich evtl. angegebener Randzeichenfolgen) angegeben werden, wenn die Zeichen, die im Text einem Element der Suchzeichenfolge entsprechen, beim Austauschen an dieser Stelle übernommen werden sollen:

- >=nn      Verweis auf das nn-te Element der Suchzeichenfolge: die Zeichen, die im zu durchsuchenden Text dem nn-ten Element der Suchzeichenfolge entsprechen, werden beim Austauschen an dieser Stelle unverändert in den Text eingesetzt. Der Verweis darf auf ein Element der Kernzeichenfolge oder ein Element einer Randzeichenfolge verweisen. Für "nn" ist eine zweistellige Zahl (bei Zahlen unter 10 also mit einer führenden "0") einzusetzen. Werden für "nn" zwei Nullen angegeben, so ist damit die gesamte Kernzeichenfolge gemeint.
- >=(nn-mm)      Verweis auf das nn-te bis mm-te Element der Suchzeichenfolge: Bedeutung analog zu >=nn; bei Zahlen unter 10 kann die führende "0" weggelassen werden. nn muss kleiner als mm sein. Wird für "nn" bzw. "mm" der Wert Null angegeben, so ist damit das erste bzw. letzte Element der Kernzeichenfolge gemeint.
- <=nn      Verweis auf das nn-letzte Element der Suchzeichenfolge: die Zeichen, die im zu durchsuchenden Text dem nn-ten Element, vom Ende her gezählt, entsprechen, werden beim Austauschen an dieser Stelle unverändert in den Text eingesetzt. Der Verweis darf auf ein Element der Kernzeichenfolge oder ein Element einer Randzeichenfolge verwei-

sen. Für "nn" ist eine zweistellige Zahl (bei Zahlen unter 10 also mit einer führenden "0") einzusetzen. Werden für "nn" zwei Nullen angegeben, so ist damit die gesamte Kernzeichenfolge gemeint.

- <= (nn-mm) Verweis auf das nn-letzte bis mm-letzte Element der Suchzeichenfolge: Bedeutung analog zu <=nn; bei Zahlen unter 10 kann die führende "0" weggelassen werden. nn muss größer als mm sein. Wird für "nn" bzw. "mm" der Wert Null angegeben, so ist damit das erste bzw. letzte Element der Kernzeichenfolge gemeint.
- >+nn Wie >=nn, jedoch werden beim Austauschen alle dem angegebenen Element der Suchzeichenfolge entsprechenden Kleinbuchstaben in Großbuchstaben umgewandelt.
- >+ (nn-mm) Wie >= (nn-mm), jedoch werden beim Austauschen alle den angegebenen Elementen der Suchzeichenfolge entsprechenden Kleinbuchstaben in Großbuchstaben umgewandelt.
- <+nn Wie <=nn, jedoch werden beim Austauschen alle dem angegebenen Element der Suchzeichenfolge entsprechenden Kleinbuchstaben in Großbuchstaben umgewandelt.
- <+ (nn-mm) Wie <= (nn-mm), jedoch werden beim Austauschen alle den angegebenen Elementen der Suchzeichenfolge entsprechenden Kleinbuchstaben in Großbuchstaben umgewandelt.
- >-nn Wie >=nn, jedoch werden beim Austauschen alle dem angegebenen Element der Suchzeichenfolge entsprechenden Großbuchstaben in Kleinbuchstaben umgewandelt.
- >- (nn-mm) Wie >= (nn-mm), jedoch werden beim Austauschen alle den angegebenen Elementen der Suchzeichenfolge entsprechenden Großbuchstaben in Kleinbuchstaben umgewandelt.
- <-nn Wie <=nn, jedoch werden beim Austauschen alle dem angegebenen Element der Suchzeichenfolge entsprechenden Großbuchstaben in Kleinbuchstaben umgewandelt.
- <- (nn-mm) Wie <= (nn-mm), jedoch werden beim Austauschen alle den angegebenen Elementen der Suchzeichenfolge entsprechenden Großbuchstaben in Kleinbuchstaben umgewandelt.
- >:nn Wie >=nn, jedoch werden beim Austauschen alle dem angegebenen Element der Suchzeichenfolge entsprechenden Zeichen aus dem ASCII-Zeichensatz in die entsprechenden mit "^" codierten Zeichen aus dem TUSTEP-Zeichensatz umgewandelt (vgl. Tabellen Seite 763 und Seite 815).
- >: (nn-mm) Wie >= (nn-mm), jedoch werden beim Austauschen alle den angegebenen Elementen der Suchzeichenfolge entsprechenden Zeichen aus dem ASCII-Zeichensatz in die entsprechenden mit "^" codierten Zeichen aus dem TUSTEP-Zeichensatz umgewandelt (vgl. Tabellen Seite 763 und Seite 815).

- <:nn Wie <=nn, jedoch werden beim Austauschen alle dem angegebenen Element der Suchzeichenfolge entsprechenden Zeichen aus dem ASCII-Zeichensatz in die entsprechenden mit "^" codierten Zeichen aus dem TUSTEP-Zeichensatz umgewandelt (vgl. Tabellen Seite 763 und Seite 815).
- <:(nn-mm) Wie <=(nn-mm), jedoch werden beim Austauschen alle den angegebenen Elementen der Suchzeichenfolge entsprechenden Zeichen aus dem ASCII-Zeichensatz in die entsprechenden mit "^" codierten Zeichen aus dem TUSTEP-Zeichensatz umgewandelt (vgl. Tabellen Seite 763 und Seite 815).
- >;nn Wie >=nn, jedoch werden beim Austauschen alle dem angegebenen Element der Suchzeichenfolge entsprechenden mit "^" codierten Zeichen aus dem TUSTEP-Zeichensatz in die entsprechenden ohne "^" codierten Zeichen aus dem ASCII-Zeichensatz umgewandelt (vgl. Tabellen Seite 763 und Seite 815).
- >:(nn-mm) Wie >=(nn-mm), jedoch werden beim Austauschen alle den angegebenen Elementen der Suchzeichenfolge entsprechenden mit "^" codierten Zeichen aus dem TUSTEP-Zeichensatz in die entsprechenden ohne "^" codierten Zeichen aus dem ASCII-Zeichensatz umgewandelt (vgl. Tabellen Seite 763 und Seite 815).
- <;nn Wie <=nn, jedoch werden beim Austauschen alle dem angegebenen Element der Suchzeichenfolge entsprechenden mit "^" codierten Zeichen aus dem TUSTEP-Zeichensatz in die entsprechenden ohne "^" codierten Zeichen aus dem ASCII-Zeichensatz umgewandelt (vgl. Tabellen Seite 763 und Seite 815).
- <:(nn-mm) Wie <=(nn-mm), jedoch werden beim Austauschen alle den angegebenen Elementen der Suchzeichenfolge entsprechenden mit "^" codierten Zeichen aus dem TUSTEP-Zeichensatz in die entsprechenden ohne "^" codierten Zeichen aus dem ASCII-Zeichensatz umgewandelt (vgl. Tabellen Seite 763 und Seite 815).

## Parameterart XI: Sonstiges

Das Informationsfeld enthält Angaben in einem bei der jeweiligen Parameterbeschreibung genannten Format.

Reicht für die Angaben eine Zeile nicht aus, so können Fortsetzungszeilen (mit der gleichen Parameterkennung) verwendet werden. An jedem Zeilenende wird ein Leerzeichen ergänzt.

## Parameterart XII: Recherchier-Tabelle

(Gilt für "<>-Konvention", für "{}-Konvention" siehe Seite 724.)

Im Informationsfeld werden Zeichenfolgen angegeben, nach denen gesucht werden soll. Sie werden durch ein Begrenzungszeichen voneinander getrennt. Das Begrenzungszeichen ist das erste Zeichen (Spalte 11) im Informationsfeld des Parameters. Die letzte Zeichenfolge muss durch ein Begrenzungszeichen abgeschlossen sein.

Das Fragezeichen und der Stern haben bei dieser Parameterart eine spezielle Bedeutung. Ein Fragezeichen steht stellvertretend für ein beliebiges TUSTEP-Zeichen und ein Stern für null bis beliebig viele beliebige TUSTEP-Zeichen.

An Stelle eines einzelnen Zeichens einer Zeichenfolge sind folgende Angaben möglich:

- Eine in eckige Klammern eingeschlossene Gruppe von Zeichen.

Die Zeichen innerhalb der eckigen Klammern werden als eine Zeichengruppe mit den angegebenen Zeichen interpretiert. Es sind die gleichen Angaben wie bei der Definition von Zeichengruppen (siehe Parameterart V Seite 741 ff.) erlaubt.

- Ein in geschweifte Klammern eingeschlossene Folge von Zeichen.

Innerhalb der geschweiften Klammern sind keine Unterschiede (Ersetzungen, Auslassungen und Einfügungen) erlaubt.

- Eine in runde Klammern eingeschlossene Folge von Zeichen.

Die in runde Klammern eingeschlossenen Zeichen gelten als eine einzige Zeichenfolge; zwischen den Klammern vorkommende Begrenzungszeichen werden nicht als solche gewertet.

- Ein Backslash und ein beliebiges Zeichen.

Ein Backslash bewirkt, dass das unmittelbar nachfolgende Zeichen seine eigentliche Bedeutung behält. Soll z. B. nach einem Fragezeichen gesucht werden, muss "\?" angegeben werden. Erforderlich ist eine Kennzeichnung mit dem Backslash für Fragezeichen, Stern, Backslash und sämtliche Klammern.

- Eine Kennung einer zuvor definierten Zeichengruppe.

Eine Kennung der Form ">n" oder "<n" wird bei dieser Parameterart immer als Kennung einer Zeichengruppe gewertet, auch wenn eine Stringgruppe mit der gleichen Kennung definiert ist.

Soll eine zu suchende Zeichenfolge nur dann gefunden werden, wenn sie am Anfang bzw. am Ende der zu durchsuchenden Zeichenfolge steht, kann die Zeichenfolge am Anfang (unmittelbar nach dem Begrenzungszeichen) mit "<|" bzw. am Ende (unmittelbar vor dem Begrenzungszeichen) mit ">|" gekennzeichnet werden.

Für jede Recherchier-Tabelle kann zusätzlich festgelegt werden,

- ob Groß- und Kleinschreibung relevant ist,
- wieviele Unterschiede (Ersetzungen, Auslassungen und Einfügungen) je Zeichenfolge erlaubt sind,

- ob die gesuchte Zeichenfolge an einer Wortgrenze (= jedes Sonderzeichen) oder an der Textgrenze (= Anfang und Ende der zu durchsuchenden Zeichenfolge) beginnen und enden muss,

In welcher Weise dies festgelegt werden kann, ist bei der jeweiligen Parameterbeschreibung angegeben.

Reicht für die Angaben eine Zeile nicht aus, so können Fortsetzungszeilen (mit der gleichen Parameterkennung und dem gleichen Begrenzungszeichen; vgl. Seite 738) verwendet werden.

### **Besonderheiten der Parameterart XII-a**

Als Begrenzungszeichen kann ein beliebiges Zeichen verwendet werden.

Es kann zusätzlich festgelegt werden, ob alle oder nur eine der angegebenen Zeichenfolgen gefunden werden muss.

In welcher Weise dies festgelegt werden kann, ist bei der jeweiligen Parameterbeschreibung angegeben.

### **Besonderheiten der Parameterart XII-b**

Als Begrenzungszeichen muss ein Leerzeichen verwendet werden.

Mehrere unmittelbar aufeinander folgende Leerzeichen werden als ein einziges Begrenzungszeichen gewertet. Eine leere Zeichenfolge kann mit zwei aufeinander folgenden Anführungszeichen (""") angegeben werden.

Anführungszeichen (") dürfen nur paarweise vorkommen. Jedes erste eines Paares wird als "runde Klammer auf", jede zweite als "runde Klammer zu" gewertet. Die so eingeschlossenen Zeichen gelten jeweils als eine einzige Zeichenfolge; zwischen den Klammern vorkommende Leerzeichen werden nicht als Begrenzungszeichen gewertet.

Die (zwischen Leerzeichen stehenden) Zeichenfolgen AND, OR, NOT, AND NOT und OR NOT werden als logische Operatoren interpretiert. Auf diese Weise kann festgelegt werden, welche der angegebenen Zeichenfolgen im zu durchsuchenden Text gefunden werden müssen.

Ist zwischen zwei Zeichenfolgen (die keine logische Operatoren sind) kein logischer Operator angegeben, so wird ein logisches ODER angenommen; dieses logische ODER hat Vorrang vor allen angegebenen logischen Operatoren. Es kann jedoch festgelegt werden, dass stattdessen ein logisches UND oder ein logisches ODER ohne Vorrang angenommen wird. In welcher Weise dies festgelegt werden kann, ist bei der jeweiligen Parameterbeschreibung angegeben.

Im Übrigen gelten die für logischen Operatoren üblichen Regeln. Eine Klammerung zur Angabe der Priorität der logischen Operatoren ist nicht möglich.

Beispiele:

Die Anführungszeichen grenzen die Tabellen gegenüber dem umgebenden Text ab und sind nicht Bestandteil der Tabellen.

" eins zwei ": ( "eins" oder "zwei" oder beide )

" eins OR zwei ": "eins" oder "zwei" oder beide

" eins AND zwei ": Sowohl "eins" als auch "zwei"

" eins AND NOT zwei ": "eins" aber nicht "zwei"

" eins AND zwei drei ": "eins" und ( "zwei" oder "drei" oder beide )

" NOT eins zwei ": ( Weder "eins" noch "zwei" )

An den Stellen, an denen zwischen zwei Zeichenfolgen kein logischer Operator angegeben ist, wird ein logisches ODER mit Vorrang angenommen (Voreinstellung).

**Zeichenvorrat**

---

Vorbemerkungen . . . . .	761
Besonderheiten für das Satzprogramm . . . . .	762
ASCII-Zeichen und mit "^" codierte Zeichen . . . . .	763
Mit Steuerzeichen "#" codierte Sonderzeichen . . . . .	765
Mit Steuerzeichen "#" codierte Sonderbuchstaben . . . . .	766
Mit Steuerzeichen "# (name)" codierte Sonderzeichen . . . . .	767
Akzente und diakritische Zeichen . . . . .	773
Griechische Schrift . . . . .	775
Koptische Schrift . . . . .	777
Hebräische Schrift . . . . .	779
Syrische Schrift . . . . .	781
Arabische Schrift . . . . .	782
Russische (kyrillische) Schrift . . . . .	786
Cyrillische (alt-kirchenslavische) Schrift . . . . .	789
Phonetische Zeichen . . . . .	791
Auszeichnungen, Schriftumschaltungen, Druckeffekte . . . . .	798
Hoch- und Tiefstellungen . . . . .	798
Alphabetische Liste der Sonderzeichen . . . . .	799



## Vorbemerkungen

In den folgenden Tabellen ist links jeweils die Eingabe-Codierung angegeben. Falls auf einer Tastatur die Umlaute und das scharfe s vorhanden sind, können diese alternativ statt der entsprechenden mit "^" gekennzeichneten Grundbuchstaben verwendet werden.

Umlaute und das scharfe s können bei der Eingabe jedoch nur verwendet werden, wenn mit dem Kommando #DEFINIERE ein entsprechender Code eingestellt worden ist. Darüber hinaus können evtl. auch Akzentbuchstaben direkt eingegeben werden. Ist z. B. der Code (die Code Page) CP437 oder CP850 eingestellt, so kann für ein e mit Gravis auch "%\e" (Gravis+e) eingegeben werden. In diesem Fall erscheint bei der Eingabe je nach Betriebssystem "%\e" oder "è" im Editorfenster. In die Datei wird unabhängig davon, wie das e mit Gravis eingegeben wurde, "%\e" eingetragen.

Die Zeichen in der ersten Tabelle werden als ein Zeichen in die Datei eingetragen und belegen jeweils ein Byte. Die Zeichen der darauf folgenden Tabellen werden als Zeichenkombinationen, die aus Zeichen der ersten beiden Tabellen zusammengesetzt sind, eingetragen und belegen mehrere Bytes. So belegt z. B. die Ligatur ae, für die die Eingabe-Codierung "#.^a" angegeben ist, drei Bytes mit der Zeichenkombination "#.ä" ("#" und "." sind Zeichen aus der ersten Tabelle und "^a" ist ein Zeichen aus der zweiten Tabelle).

Jedem Buchstaben können bis zu drei Akzente zugeordnet werden: zwei über dem Buchstaben und einer unter dem Buchstaben. Die Akzent-Codierungen müssen vor dem jeweiligen Buchstaben stehen (bei mehr als einem Akzent in der Reihenfolge von oben nach unten). Frei stehende Akzente müssen vor einem festen Abschluss (z. B. "%<\_" für frei stehenden Zirkumflex) codiert werden.

Für die Darstellung nicht-lateinischer Schriften ist zusätzlich eine Auszeichnung (siehe Seite 798) am Anfang und Ende erforderlich, die hier bei der Eingabe-Codierung nicht eigens angegeben ist.

"(STZ)" hinter dem Namen eines Zeichens heißt, dass dieses Zeichen in TUSTEP-Programmen beim Aufbereiten zum Drucken als Steuerzeichen verwendet wird. Soll ein so gekennzeichnetes Zeichen nicht als Steuerzeichen, sondern als zu druckendes Zeichen verwendet werden, so ist bei der Eingabe davor ein "^" zu schreiben.

## Besonderheiten für das Satzprogramm

Der Zeichenvorrat des Satzprogramms (Kommando #SATZ) weicht in einigen Punkten vom hier beschriebenen Zeichenvorrat ab. Er ist in der Beschreibung des Satzprogramms ab Kapitel 12 angegeben. Die wichtigsten Abweichungen für die lateinischen Schriften sind:

- Die Zeichen { und } sind Steuerzeichen und müssen mit #. { und #. } codiert werden, wenn sie gedruckt werden sollen.
- Die Zeichenpaare """, ++, !!, << und >> sowie das Zeichen ^ (Eingabe-Codierung ^^) gelten als Steuerzeichen.
- Von den mit Steuerzeichen # (name) codierten Zeichen stehen folgende nicht zur Verfügung: "Box: ...", "Menge der ... Zahlen" und "... (groß)".

## ASCII-Zeichen und mit "^" codierte Zeichen

		Leerzeichen, Blank
!	!	Ausrufezeichen
^!	¡	spanisches Ausrufezeichen
"	"	Anführungszeichen
^"	„	Schmierzeichen
#	#	Nummernzeichen (STZ für Sonderzeichen)
^#	#	Nummernzeichen
\$	\$	Dollarzeichen (STZ zur Druckaufbereitung)
^\$	\$	Dollarzeichen
%	%	Prozentzeichen (STZ für Akzente)
^%	%	Prozentzeichen
&	&	Et-Zeichen (STZ zur Druckaufbereitung)
^&	&	Et-Zeichen
'	'	Apostroph
^'	‘	umgekehrter Apostroph
(	(	runde Klammer auf
)	)	runde Klammer zu
*	*	Stern
^*	×	Malkreuz
+	+	Pluszeichen
^+	†	Sterbekreuz
,	,	Komma
-	-	Minuszeichen
^-	—	langes Minuszeichen
.	.	Punkt
^.	·	Malpunkt
/	/	Schrägstrich
:	:	Doppelpunkt
;	;	Strichpunkt
<	<	spitze Klammer auf (STZ für XML-Tags)
^<	<	spitze Klammer auf

---

>		spitze Klammer zu (STZ für XML-Tags)
^>	>	spitze Klammer zu
=	=	Gleichheitszeichen
^=	●	Rückweisungszeichen
?	?	Fragezeichen
^?	¿	spanisches Fragezeichen
@		Klammeraffe (STZ zur Druckaufbereitung)
^@	@	Klammeraffe
[	[	eckige Klammer auf
]	]	eckige Klammer zu
\		Backslash (STZ für Kann-Trennstelle)
^\	\	Backslash
_		Unterstreichungsstrich (STZ für festes Blank)
^_	_	Unterstreichungsstrich
^^	^	Zirkumflex (nicht als Akzent)
{	{	geschweifte Klammer auf
}	}	geschweifte Klammer zu
		senkrechter Strich
^		langer senkrechter Strich
1	1	Ziffern 0 bis 9
^1	<sup>1</sup>	kleine hochgestellte Ziffern 0 bis 9
a	a	Kleinbuchstaben a bis z
^a, ä	ä	kleiner Umlaut ä
^o, ö	ö	kleiner Umlaut ö
^u, ü	ü	kleiner Umlaut ü
^s, ß	ß	kleines scharfes s (#K+ß#K- → ss)
A	A	Großbuchstaben A bis Z
^A, Ä	Ä	großer Umlaut Ä
^O, Ö	Ö	großer Umlaut Ö
^U, Ü	Ü	großer Umlaut Ü
^S	SS	großes scharfes S (→ SS)

## Mit Steuerzeichen "# " codierte Sonderzeichen

#'x	×	hochgestellte Zeichen (ASCII-Zeichensatz)
#,x	×	tiefgestellte Zeichen (ASCII-Zeichensatz)
#;x	×	übersetzte Zeichen (ASCII-Zeichensatz)
#!x	×	untergesetzte Zeichen (ASCII-Zeichensatz)
#.!	§	Paragraphzeichen
#.,	„	Anführungszeichen unten
#.'	”	Anführungszeichen oben
#.(	◌	Ajin
#.)	◌	Alef
#.%	◌	Doppel-Alef
#.*	◌	Grad
#.-	′	Minute
#.=	”	Sekunde
#..	→	Verweispfeil
#./		doppelter senkrechter Strich
#.:	>	einfaches Anführungszeichen
#.;	<	einfaches Anführungszeichen
#.>	»	doppeltes Anführungszeichen
#.<	«	doppeltes Anführungszeichen
#.[	┌	Winkel links oben
#.]	┐	Winkel rechts oben

## Mit Steuerzeichen "#" codierte Sonderbuchstaben

#'x	×	hochgestellte Buchstaben (keine Sonderbuchstaben)
#,x	×	tiefgestellte Buchstaben (keine Sonderbuchstaben)
#;x	×	übergesetzte Buchstaben (keine Sonderbuchstaben)
#!x	×	untergesetzte Buchstaben (keine Sonderbuchstaben)
#.b	ḃ	b mit Querstrich
#.d	ḏ	d mit Querstrich (serbokroatisch)
#.D	Ḑ	D mit Querstrich (serbokroatisch)
#.h	ħ	h mit Querstrich (maltesisch)
#.H	Ĥ	H mit Querstrich (maltesisch)
#.i	ı	i ohne Punkt (türkisch)
#.j	ij	Ligatur ij
#.J	IJ	Ligatur IJ
#.l	ł	l mit Querstrich (polnisch)
#.L	Ł	L mit Querstrich (polnisch)
#.o	ø	o mit Schrägstrich (dänisch)
#.O	Ø	O mit Schrägstrich (dänisch)
#.p	þ	kleines Thorn (isländisch)
#.P	Þ	großes Thorn (isländisch)
#.s	ſ	langes s
#.z	ȝ	langes z / kleines Yogh (altenglisch)
#.Z	ȝ	großes Yogh (altenglisch)
#.^a, #.ä	æ	Ligatur ae
#.^A, #.Ä	Æ	Ligatur AE
#.^d	ð	kleines Eth (isländisch, altenglisch)
#.^D	Ð	großes Eth (isländisch, altenglisch)
#.^o, #.ö	œ	Ligatur oe
#.^O, #.Ö	Œ	Ligatur OE
#.^s, #.ß	ß	kleines scharfes s (#K+#.ß#K- → ß)
#.^S	ß	großes scharfes S (→ Œ)

## Mit Steuerzeichen "# (name)" codierte Sonderzeichen

Im Satzprogramm stehen folgende Zeichen nicht zur Verfügung:  
 "Box: ...", "Menge der ... Zahlen" und "... (groß)".

# (AEH)	~	ähnlich
# (AU)	°	Auslassung
# (AU/)	◻	Anfang einer Auslassung
# (/AUL)	∕	Ende einer Auslassung (links)
# (/AUR)	∖	Ende einer Auslassung (rechts)
# (BSL)	\	Backslash
# (BOL)	┌	Box: Ecke oben links
# (BOM)	┐	Box: Ecke oben Mitte
# (BOR)	└	Box: Ecke oben rechts
# (BML)	├	Box: Ecke Mitte links
# (BMM)	┤	Box: Ecke Mitte Mitte
# (BMR)	┘	Box: Ecke Mitte rechts
# (BUL)	└	Box: Ecke unten links
# (BUM)	┘	Box: Ecke unten Mitte
# (BUR)	┘	Box: Ecke unten rechts
# (C)	©	Copyright
# (CE)	¢	Cent-Zeichen
# (DIF)	∂	Differential
# (DIV)	÷	Division
# (DO)	\$	Dollarzeichen
# (DEA)	⌈	doppelte eckige Klammer auf
# (DEZ)	⌋	doppelte eckige Klammer zu
# (DEAG)	⌈	doppelte eckige Klammer auf (groß)
# (DEZG)	⌋	doppelte eckige Klammer zu (groß)
# (DP)	”	doppelt gestrichen

---

# (DSS)	$\parallel$	doppelter senkrechter Strich
# (DF)	$'''$	dreifach gestrichen
# (DM)	$\cap$	Durchschnittsmenge
# (DMG)	$\bigcap$	Durchschnittsmenge (groß)
# (E)	$\text{€}$	Euro-Zeichen
# (EAG)	$[$	eckige Klammer auf (groß)
# (EZG)	$]$	eckige Klammer zu (groß)
# (EF)	$'$	einfach gestrichen
# (EG)	$\vee$	es gibt
# (EGE)	$\exists$	es gibt (umgedrehtes E)
# (EI)	$\tau$	Einfügung
# (EI.)	$\tau$	Einfügung (mit Punkt)
# (EL)	$\in$	Element
# (ENT)	$\cong$	entspricht
# (EOM)	$\supset$	echte Obermenge
# (ETM)	$\subset$	echte Teilmenge
# (ER/)	$\ulcorner$	Anfang einer Ersetzung
# (ER./)	$\urcorner$	Anfang einer Ersetzung (mit Punkt)
# (/ER)	$\lrcorner$	Ende einer Ersetzung
# (/ER.)	$\llcorner$	Ende einer Ersetzung (mit Punkt)
# (ERL)	$\lceil$	Ersetzung (links)
# (ERL.)	$\lceil$	Ersetzung (links, mit Punkt)
# (ERR)	$\lrcorner$	Ersetzung (rechts)
# (ERR.)	$\llcorner$	Ersetzung (rechts, mit Punkt)
# (FA)	$\wedge$	für alle
# (FAA)	$\forall$	für alle (umgedrehtes A)
# (FUE)	$\infty$	fast unendlich
# (GES)	$\cap$	geschnitten mit
# (GAG)	$\{$	geschweifte Klammer auf (groß)
# (GZG)	$\}$	geschweifte Klammer zu (groß)



# (GR)	>	größer
# (GGL)	$\approx$	größer gleich (mit -)
# (GGLD)	$\cong$	größer gleich (mit =)
# (HSS)	'	Halber senkrechter Strich (oben)
# (ID)	$\equiv$	identisch
# (INT)	$\int$	Integral
# (INTG)	$\int$	Integral (groß)
# (IP)	:	Interpunktionszeichen
# (IW)	¤	int. Währungszeichen
# (KP)	$\times$	kartesisches Produkt
# (KPG)	$\times$	kartesisches Produkt (groß)
# (KL)	<	kleiner
# (KGL)	$\leq$	kleiner gleich (mit -)
# (KGLD)	$\leq$	kleiner gleich (mit =)
# (KGR)	$\approx$	kongruent (mit -)
# (KGRD)	$\equiv$	kongruent (mit =)
# (KRS)	○	Kreis
# (LM)	∅	leere Menge
# (LN)	¬	logisches Nicht
# (LO)	∨	logisches Oder
# (LU)	∧	logisches Und
# (MKR)	×	Malkreuz
# (MPU)	·	Malpunkt
# (MC)	ℂ	Menge der komplexen Zahlen
# (MN)	ℕ	Menge der natürlichen Zahlen
# (MQ)	ℚ	Menge der rationalen Zahlen
# (MR)	ℝ	Menge der reellen Zahlen
# (MZ)	ℤ	Menge der ganzen Zahlen
# (MKS)	˘	Metrik: Kurze Silbe
# (MKLS)	˘	Metrik: Kurze oder lange Silbe
# (MLS)	˘	Metrik: Lange Silbe

---

# (MPL)	$\mp$	Minus-Plus
# (NBL)	$\nabla$	Nabla-Operator
# (NEL)	$\notin$	nicht Element
# (NID)	$\neq$	nicht identisch (mit /)
# (NIDS)	$\neq$	nicht identisch (mit  )
# (NOM)	$\supsetneq$	nicht Obermenge
# (NTM)	$\not\subset$	nicht Teilmenge
# (P)	$\textcircled{P}$	Published
# (PA)	$\S$	Paragraphzeichen
# (PFB)	$\leftrightarrow$	Pfeil nach beiden Seiten
# (PFBD)	$\Leftrightarrow$	Pfeil nach beiden Seiten (doppelt)
# (PFL)	$\leftarrow$	Pfeil nach links
# (PFLD)	$\Leftarrow$	Pfeil nach links (doppelt)
# (PFLU)	$\nearrow$	Pfeil nach links oben
# (PFLU)	$\swarrow$	Pfeil nach links unten
# (PFLR)	$\rightleftarrows$	Pfeile nach links / rechts
# (PFO)	$\uparrow$	Pfeil nach oben
# (PFOO)	$\Uparrow$	Pfeile nach oben / oben
# (PFOU)	$\Downarrow$	Pfeile nach oben / unten
# (PFR)	$\rightarrow$	Pfeil nach rechts
# (PFRD)	$\Rightarrow$	Pfeil nach rechts (doppelt)
# (PFRU)	$\nearrow$	Pfeil nach rechts oben
# (PFRU)	$\searrow$	Pfeil nach rechts unten
# (PFU)	$\downarrow$	Pfeil nach unten
# (PF)	$\pounds$	Pfund-Zeichen
# (PLM)	$\pm$	Plus-Minus
# (PR)	$\Pi$	Produkt
# (PRG)	$\prod$	Produkt (groß)
# (PRM)	$\text{‰}$	Promillezeichen
# (QUA)	$\square$	Quadrat
# (R)	$\textcircled{R}$	Registered

# (RAG)	(	runde Klammer auf (groß)
# (RZG)	)	runde Klammer zu (groß)
# (SA)	<	spitze Klammer auf
# (SZ)	<	spitze Klammer zu
# (SR)	⊥	senkrecht
# (SUM)	Σ	Summe
# (SUMG)	∑	Summe (groß)
# (TM)	™	Trade Mark
# (UOM)	⊃	unechte Obermenge
# (UTM)	⊆	unechte Teilmenge
# (UE)	∞	unendlich
# (UGF)	≈	ungefähr
# (UGR)	≥	ungefähr größer
# (UKL)	≤	ungefähr kleiner
# (UGL)	≠	ungleich (mit /)
# (UGLS)	≠	ungleich (mit !)
# (UM/)	⌞	Anfang einer Umstellung
# (UM./)	⌞	Anfang einer Umstellung (mit Punkt)
# (/UM)	⌟	Ende einer Umstellung
# (/UM.)	⌟	Ende einer Umstellung (mit Punkt)
# (USS)	⋮	unterbrochener senkrechter Strich
# (VER)	∪	vereinigt mit
# (VM)	∪	Vereinigungsmenge
# (VMG)	⊃	Vereinigungsmenge (groß)
# (VH)	∞	verheiratet, Heirat
# (WI)	∠	Winkel
# (WU)	√	Wurzel
# (WUG)	√	Wurzel (groß)
# (ZWR)	–	Zeichen für Zwischenraum

**Beispiel für eine Box (mit FORMATIERE-Anweisungen):**

```
&? &a10 &+12 &m &+12 &m
&? #(BOL) @* ^- &t2 #(BOR)
&? ^| Beispiel für eine Box &t2 ^|
&? #(BML) @* ^- &t #(BOM) @* ^- &t #(BMR)
&? ^| linke &t ^| rechte &t ^|
&? ^| Spalte &t ^| Spalte &t ^|
&? #(BUL) @* ^- &t #(BUM) @* ^- &t #(BUR)
```

Beispiel für eine Box	
linke Spalte	rechte Spalte

## Akzente und diakritische Zeichen

### a) über den Buchstaben

%"	ö	Doppelakut
%'	ó	Betonungszeichen
% (	ô	Bogen (unten offen)
%)	õ	Halbkreis (oben offen)
%*	ȯ	Kringel, Ringel(chen)
%+	ő	Plus
%,	ọ	Komma, Apostroph
%-	o̅	Querstrich, Balken
%.	ȯ	Punkt
%/	ȯ	Akut
%\	ö	Gravis
%:	ö	Trema
%<	ô	Zirkumflex, Dach
%>	õ	Haček, Haken
%=	õ	Kreuz
%?	õ	Tilde
%[	ō	Brücke
%]	o̅	umgekehrte Brücke
%{	ȯ	Kreis rechts offen
%}	ȯ	Kreis links offen

## b) unter den Buchstaben

%"	◌̣	Doppelakut
%'	◌̇	Betonungszeichen
%(	◌̅	Bogen (unten offen)
%)	◌̆	Halbkreis (oben offen)
%"	◌̈	Kringel, Ringel(chen)
%"	♀	Plus
%"	◌̣	Komma
%"	◌̄	Querstrich, Balken
%"	◌̇	Punkt
%"	◌̆	Akut
%"	◌̣	Gravis
%"	◌̈	Trema
%"	◌̣	Cedille
%"	◌̣	Krummhaken, Ogonek
%"	◌̆	Zirkumflex, Dach
%"	◌̆	Haček, Haken
%"	♀	Kreuz
%"	◌̇	Tilde
%"	◌̣	Brücke
%"	◌̣	umgekehrte Brücke
%"	◌̆	Kreis rechts offen
%"	◌̆	Kreis links offen
%"	◌̆	front sign, subscript
%"	◌̆	back sign, subscript
%"	◌̆	laminal sign, subscript
%"	◌̆	raising sign, subscript
%"	◌̆	lowering sign, subscript

## Griechische Schrift

Anfang bzw. Ende Griechisch: #G+ bzw. #G-

a	α	A	Α	Alpha
b	β	B	Β	Beta
g	γ	G	Γ	Gamma
d	δ	D	Δ	Delta
e	ε	E	Ε	Epsilon
z	ζ	Z	Ζ	Zeta
h	η	H	Η	Eta
u, ^u	θ, ϑ	U	Θ	Theta
i	ι	I	Ι	Iota
k	κ	K	Κ	Kappa
l	λ	L	Λ	Lambda
m	μ	M	Μ	My
n	ν	N	Ν	Ny
j	ξ	J	Ξ	Xi
o	ο	O	Ο	Omikron
p	π	P	Π	Pi
r	ρ	R	Ρ	Rho
s, w, q	σ, ς, Ϸ	S, Q	Σ, ϸ	Sigma
t	τ	T	Τ	Tau
y	υ	Y	Υ	Ypsilon
f, ^f	φ, ϕ	F	Φ	Phi
x	χ	X	Χ	Chi
c	ψ	C	Ψ	Psi
v	ω	V	Ω	Omega
		W	Ϻ	Digamma
#.s	ς			Stigma = 6
#.k	ϙ			Koppa = 90
#.p	Ϟ			Sampi = 900
#.w	Ϸ			
!	·			griechisches Semikolon
;	;			griechisches Fragezeichen
?	;			griechisches Fragezeichen

## Akzente in der griechischen Schrift

% (	ó	Spiritus asper
%)	ò	Spiritus lenis
%/	ó	Akut
%\	ò	Gravis
%?	ō	Zirkumflex
%( /	ǒ	Spiritus asper + Akut
%( \	ḡ	Spiritus asper + Gravis
%( ?	ṑ	Spiritus asper + Zirkumflex
%) /	ǒ	Spiritus lenis + Akut
%) \	ḡ	Spiritus lenis + Gravis
%) ?	ṑ	Spiritus lenis + Zirkumflex
%;	Ϸ	Iota subscriptum
%/:	ö	Akut + Trema
%\:	ḡ	Gravis + Trema
^a	α	Alpha mit Iota subscriptum
^h	η	Eta mit Iota subscriptum
^v	ω	Omega mit Iota subscriptum
%-	ō	Querstrich, Balken (über dem Buchstaben)
%.	ó	Punkt (über dem Buchstaben)
%:	ö	Trema (über dem Buchstaben)
%--	ω	Querstrich, Balken (unter dem Buchstaben)
%..	Ϸ	Punkt (unter dem Buchstaben)
%::	ω	Trema (unter dem Buchstaben)

Soll ein Akzent vor einen griechischen Großbuchstaben gesetzt werden, so muss zwischen Akzent-Codierung und Buchstabe ein "\_" eingefügt werden (gilt nicht bei Daten für das Satzprogramm).



## Koptische Schrift

Anfang bzw. Ende Koptisch: #T+ bzw. #T-

a	Ⲁ	Alpha
b	Ⲃ	Beta
g	Ⲅ	Gamma
d	Ⲇ	Dalda
e	Ⲉ	Ei
z	Ⲋ	Zita
E	Ⲍ	(H)Ita
u	Ⲏ	Thita
i	Ⲑ	Jota
k	Ⲓ	Kappa
l	Ⲕ	Lawda
m	Ⲗ	Mi
n	Ⲙ	Ni
X	Ⲛ	Xi
o	Ⲝ	O(u)
p	Ⲟ	Pi
r	Ⲡ	Ro
s	Ⲣ	Simma
t	Ⲥ	Tau
y	ⲧ	He
F	ⲩ	Phi
K	ⲫ	Chi
P	ⲭ	Psi
O	ⲯ	O
S	ⲱ	Schai
f	ⲳ	Fai
B, x	ⲵ, Ⲷ	Chai
H	Ⲹ	
h	Ⲻ	Hori
D	Ⲽ	Dschandscha

G	ϸ	Gima
T	ⲧ	Ti
W	Ϩ	Stigma = 6
R	Ⲣ	Sampi = 900

### Sonderzeichen in der koptischen Schrift

!	Ⲁ	Ausrufezeichen
,	ⲁ	Komma
-	Ⲃ	Strich
.	ⲃ	Punkt
/	Ⲅ	Abkürzungsstrich
;	ⲅ	Fragezeichen
=	Ⲇ	Suffix-Trenner
?	ⲇ	Fragezeichen
Ⲉ	ⲉ	Vokalstrich
Ⲋ	ⲋ	Punkt
Ⲍ	ⲍ	Trema
Ⲏ	ⲏ	Punkt

## Hebräische Schrift

Anfang bzw. Ende Hebräisch: #H+ bzw. #H-

a	א	A	א	Alef
b	ב	B	ב	Bet
g	ג	G	ג	Gimel
d	ד	D	ד	Dalet
h	ה	H	ה	He
u	ו	U	ו	Waw
z	ז	Z	ז	Zajin
x	ח			Chet
j	ט	J	ט	Tet
i	י	I	י	Jud
k, ^k	כ, ך	K, ^K	כ, ך	Kaf
l	ל	L	ל	Lamed
m, ^m	מ, ם	M	מ	Mem
n, ^n	נ, ן	N	נ	Nun
s	ס	S	ס	Samech
y	ע			Ajin
p, ^p	פ, ף	P, ^P	פ, ף	Pe
c, ^c	צ, ץ	C	צ	Zade
q	ק	Q	ק	Kuf
r	ר	R	ר	Resch
w	ש	W	ש	Sin, Schin
t	ת	T	ת	Taw
#.u	וו			Ligatur Waw Waw
#.^u	וּ			Ligatur Waw Jud
#.i	יּ			Ligatur Jud Jud
'	'			einfaches Abkürzungszeichen
"	"			doppeltes Abkürzungszeichen
-	-			Maqgef
:	:			Soph pasuq
^.	.			

## Punktierung im Hebräischen

#"0	ְ	Schwa
#"1	ֿ	Chirek
#"2	ֿ	Zere
#"3	ֿ	Segol
#"4	ֿ	Kamaz
#"5	ֿ	Patach
#"6	ֿ	Kubuz
#"7	ֿ	Cholem
#"8	ׁ	Schin
#"9	ׂ	Sin
#"0#"3	ֿֿ	Chataf Segol
#"3#"0	ֿֿ	Chataf Segol
#"0#"4	ֿֿ	Chataf Kamaz
#"4#"0	ֿֿ	Chataf Kamaz
#"0#"5	ֿֿ	Chataf Patach
#"5#"0	ֿֿ	Chataf Patach
^k#"0	ֿ	Schluss-Kaf mit Schwa
^k#"4	ֿ	Schluss-Kaf mit Kamaz
u#"7	ֿ	Waw mit Cholem
#"-	ֿ	einfacher Rafe
#"=	ֿֿ	doppelter Rafe
#"*	ֿ	Kreis
#"+	ֿ	Kreuz
#">	ֿ	Haken

## Beispiel zur Codierung

#h+w#"8#"4l#"7u^m#h- → שְׁלוֹם

## Syrische Schrift

1. Spalte: nach rechts verbundene (End-) Buchstaben
2. Spalte: beidseitig verbundene Buchstaben
3. Spalte: nach links verbundene (Anfangs-) Buchstaben
4. Spalte: allein stehende Buchstaben

Anfang bzw. Ende Syrisch: #Y+ bzw. #Y-

^a		^A	Ⲁ	Ⲁ	Alef
^b	b	B	Ⲁ	Ⲁ	Beth
^g	g	G	Ⲁ	Ⲁ	Gomal
^d		^D	Ⲁ	Ⲁ	Dolath
^h		^H	Ⲁ	Ⲁ	He
^u		^U	Ⲁ	Ⲁ	Vaw
^z		^Z	Ⲁ	Ⲁ	Zain
^x	x	X	Ⲁ	Ⲁ	Khet
^j	j	J	Ⲁ	Ⲁ	Teth
^i	i	I	Ⲁ	Ⲁ	Jud
^k	k	K	Ⲁ	Ⲁ	Koph
^l	l	L	Ⲁ	Ⲁ	Lomad
^m	m	M	Ⲁ	Ⲁ	Mim
^n	n	N	Ⲁ	Ⲁ	Nun
^s	s	S	Ⲁ	Ⲁ	Semkath
^y	y	Y	Ⲁ	Ⲁ	Ee
^p	p	P	Ⲁ	Ⲁ	Pe
^c		^C	Ⲁ	Ⲁ	Sode
^q	q	Q	Ⲁ	Ⲁ	Qoph
^r		^R	Ⲁ	Ⲁ	Ris
^w	w	W	Ⲁ	Ⲁ	Sin
^t		^T	Ⲁ	Ⲁ	Tau

## Arabische Schrift

1. Spalte: nach rechts verbundene (End-) Buchstaben
2. Spalte: beidseitig verbundene Buchstaben
3. Spalte: nach links verbundene (Anfangs-) Buchstaben
4. Spalte: nicht verbundene (allein stehende) Buchstaben

Anfang bzw. Ende Arabisch: #A+ bzw. #A-

^a	^A	ا	ا	ʾ/a	Alif
^b	b B ^B	ب	ب ب ب	b	Bāʾ
^t	t T ^T	ت	ت ت ت	t	Tāʾ
^o	o O ^O	ث	ث ث ث	ṭ	Ṭāʾ
^j	j J ^J	ج	ج ج ج	ǧ	Ǧīm
^h	h H ^H	ح	ح ح ح	ḥ	Ḥāʾ
^x	x X ^X	خ	خ خ خ	ḫ	Ḫā
^d	d	د	د	d	Dāl
^D	D	ذ	ذ	ḍ	Ḍāl
^r	r	ر	ر	r	Rāʾ
^R	R	ز	ز	z	Zāy
^s	s S ^S	س	س س س	s	Sīn
^w	w W ^W	ش	ش ش ش	š	Šīn
^c	c C ^C	ص	ص ص ص	ṣ	Ṣād
^g	g G ^G	ض	ض ض ض	ḍ	Ḍād
^p	p P ^P	ط	ط ط ط	ṭ	Ṭāʾ
^z	z Z ^Z	ظ	ظ ظ ظ	ẓ	Ẓāʾ
^y	y Y ^Y	ع	ع ع ع	ʿ	ʿAin
^v	v V ^V	غ	غ غ غ	ǧ	Ǧain
^f	f F ^F	ف	ف ف ف	f	Fāʾ
^q	q Q ^Q	ق	ق ق ق	q	Qāf
^k	k K ^K	ك	ك ك ك	k	Kāf
^l	l L ^L	ل	ل ل ل	l	Lām
^m	m M ^M	م	م م م	m	Mīm
^n	n N ^N	ن	ن ن ن	n	Nūn
^e	e E ^E	ه	ه ه ه	h	Hāʾ
^u	^U	و	و	w/u	Wāw
^i	i I ^I	ي	ي ي ي	y/i	Yāʾ

## Ligaturen und Sonderzeichen im Arabischen

	A	ا	Alif vor Lām am Wortanfang
#.^f	#.^F	في	(fi) Ligatur Fā-Yāʾ
#.^l	#.^L	لي	(li) Ligatur Lām-Yāʾ
#.^x	#.^X	لا	(la) Ligatur Lām-Alif
	#.^A	ء	Hamza allein
#.i	#.I	إ	Hamza über Yāʾ
#.j	#.J	أ	Hamza unter Yāʾ
#.^u	#.^U	ؤ	Hamza über Wāw
#.^e	#.^E	ة	Tāʾ marbūṭa
!	!	!	Ausrufezeichen
"	"	–	Anführungszeichen
'	'	'	Apostroph
,	,	،	Komma
.	.	.	Punkt
:	:	:	Doppelpunkt
;	;	؛	Strichpunkt
?	?	؟	Fragezeichen

## Ziffern im Arabischen

0	٠	Null
1	١	Eins
2	٢	Zwei
3	٣	Drei
4	٤	Vier
5	٥	Fünf
6	٦	Sechs
7	٧	Sieben
8	٨	Acht
9	٩	Neun

## Zusätzliche Zeichen im Persischen

#. ^p	#. p	#. P	#. ^P	پ پ پ پ	p	Pe
#. ^h	#. h	#. H	#. ^H	چ چ چ چ	č	Čim
#. ^R			#. R	ژ ژ	ž	Že
#. ^k	#. k	#. K	#. ^K	گ گ گ گ	g	Gāf

## Vokalzeichen und Lesezeichen im Arabischen

## a) über den Buchstaben

%)	◌ <sup>ه</sup>	Hamza
%/	◌ <sup>ف</sup>	Fatḥa
%"	◌ <sup>ت</sup>	Fatḥa-Tanwīn
%,	◌ <sup>د</sup>	Ḍamma
%:	◌ <sup>د</sup>	Ḍamma-Tanwīn
%*	◌ <sup>و</sup>	Sukūn
%>	◌ <sup>ص</sup>	Tašdīd
%!	◌ <sup>ا</sup>	Alif
%?	◌ <sup>م</sup>	Madda
%.	◌ <sup>و</sup>	Waṣla
%\	◌ <sup>ف</sup>	Fatḥa über Ligatur Lām-Alif
%(	◌ <sup>و</sup>	Sukūn über Ligatur Lām-Alif
%<	◌ <sup>ص</sup>	Tašdīd über Ligatur Lām-Alif

## b) unter den Buchstaben

%) )	◌ <sup>ه</sup>	Hamza
%//	◌ <sup>ك</sup>	Kasra
%\ \	◌ <sup>ك</sup>	Kasra unter Ligatur (rechts)
%;	◌ <sup>ه</sup>	Hamza + Kasra
%" "	◌ <sup>ت</sup>	Kasra-Tanwīn
%; ;	◌ <sup>ي</sup>	i-Punkte für Schluss-Yā <sup>3</sup>
%, ,	◌ <sup>ك</sup>	i-Punkte + Kasra für Schluss-Yā <sup>3</sup>
%' '	◌ <sup>ت</sup>	i-Punkte + Kasra-Tanwīn für Schluss-Yā <sup>3</sup>



## Anmerkungen zur Datenerfassung

Es steht ein Makro \*CASH zur Verfügung, das die Unterscheidung von links verbundenen, beidseitig verbundenen, rechts verbundenen und unverbundenen Buchstaben vornimmt. Wenn dieses Makro verwendet wird, brauchen die Buchstaben (außer bei Dāl und Zāy) nur in Kleinschreibung erfasst zu werden. Dies gilt auch für die mit "#." codierten Zeichen. So genügt es also z. B. "a" statt "^A" und für persische Buchstaben z. B. "#.p" statt "#.^P" zu schreiben.

Das Makro verwandelt außerdem "a1" am Wortanfang in "A1" (um das Alif näher an das Lām heranzurücken).

Die Zeichenfolgen "fi" und "li" (falls nicht nach links verbunden) sowie "la" werden automatisch in die Codierungen für die entsprechenden Ligaturen umgewandelt.

Vokal- und Lesezeichen werden vor den Konsonanten geschrieben, auf bzw. unter dem sie stehen. Treffen mehr als ein Vokalzeichen / Lesezeichen auf einem Buchstaben zusammen, so sind sie in der Reihenfolge von oben nach unten zu schreiben.

Beispiel: %;a\*b%/ra%/ei%,m %.al%/>n%/b%,%>i

(bzw. %;A\*b%/r^A%/Ei%,^m %.AL%/>n%/b%,%>^i)

für اِبْرَاهِيْمُ النَّبِيُّ Ibrāhīmu 'n-nabīyu

(gesetzt: اِبْرَاهِيْمُ النَّبِيُّ Ibrāhīmu 'n-nabīyu)

## Russische (kyrillische) Schrift

Anfang bzw. Ende Russisch #R+ bzw. #R-

a	а	А	А	a	a
b	б	В	Б	b	be
v	в	В	В	v	we
g	г	Г	Г	g	ge
d	д	Д	Д	d	de
e	е	Е	Е	e	je
^e	ё	^Е	Ё	ë	jo
h	ж	Н	Ж	ž	sche
z	з	З	З	z	se
i	и	И	И	i	i
j	й	Й	Й	j	i kratkoe
k	к	К	К	k	ka
l	л	Л	Л	l	el
m	м	М	М	m	em
n	н	Н	Н	n	en
o	о	О	О	o	o
p	п	Р	П	p	pe
r	р	Р	Р	r	er
s	с	С	С	s	es
t	т	Т	Т	t	te
u	у	У	У	u	u
f	ф	Ф	Ф	f	ef
x	х	Х	Х	ch	cha
c	ц	С	Ц	c	ze
q	ч	Ч	Ч	č	tsche
w	ш	Ш	Ш	š	scha
^w	щ	^Ш	Щ	šč	schtscha
^p	ъ	^Р	Ъ	-	jor, twerdyi snak
y	ы	Ы	Ы	y	jery
^b	ь	^В	Ь	'	jer, mjagkij snak

^a	э	^A	Э	è	e oborotnoje
^u	ю	^U	Ю	ju	ju
^o	я	^O	Я	ja	ja
^i	и	^I	И	i	i s totschkoi
^y	ѣ	^Y	Ђ	ě,ja	jatj
^f	ѳ	^F	Ф	f	fita
^v	ѵ	^V	Ѵ	v	ischiza

### Abweichende Zeichen im Bulgarischen

^w	щ	^W	Щ	št
^p	ѡ	^P	Ђ	ă
^z	Ѣ	^Z	Ђ	o

### Abweichende Zeichen im Makedonischen

%/g	ѓ	%/G	Ѓ	ǵ
%/k	ќ	%/K	Ќ	ǰ
^s	s	^S	S	dz
^j	j	^J	J	j
x	x	X	X	h
^l	љ	^L	Љ	lj
^n	њ	^N	Њ	nj
^c	џ	^C	Џ	dž

### Abweichende Zeichen im Serbischen

^d	ђ	^D	Ђ	d
^j	j	^J	J	j
^l	љ	^L	Љ	lj
^n	њ	^N	Њ	nj
^h	ћ	^H	Ђ	ć
x	x	X	X	h
^c	џ	^C	Џ	dž

**Abweichende Zeichen im Ukrainischen**

g	г	G	Г	h
^g	є	^G	Є	je
i	и	I	И	y
%;^i	ï	%;^I	Ï	ï
^i	i	^I	I	i
,	'			-

**Abweichende Zeichen im Weißrussischen**

g	г	G	Г	h
%)u	ŷ	%)U	Ÿ	ů

**Akzente**

%)	ö	Bogen
%;	ö	Trema
%/	ó	Akut
%;\	ò	Gravis

## Cyrillische (alt-kirchenslavische) Schrift

Anfang bzw. Ende Cyrillisch: #C+ bzw. #C-

a	ⱁ	ⱂ	ⱃ	Az
b	ⱄ	ⱅ	ⱆ	Buki
v	ⱇ	ⱈ	ⱉ	Vědi
g	ⱊ	ⱋ	ⱌ	Glagol'
d	ⱍ	ⱎ	ⱏ	Dobro
e	ⱐ	ⱑ	ⱒ	Est'
^g	ⱓ	ⱔ	ⱕ	Est'
h	ⱖ	ⱗ	ⱘ	Živěte
^s	ⱙ	ⱚ	ⱛ	Zělo
z	ⱜ	ⱝ	ⱞ	Zemlja
i	ⱟ	Ⱡ	ⱡ	Iže
j	Ɫ	Ᵽ	Ɽ	I
k	ⱦ	Ⱨ	ⱨ	Kako
l	ⱪ	Ⱬ	ⱬ	Ljudi
m	Ɱ	Ɐ	Ɒ	Myslite
n	Ⱳ	ⱳ	ⱴ	Naš
o	ⱶ	ⱷ	ⱸ	On
p	ⱺ	ⱻ	ⱼ	Pokoj
r	ⱽ	Ȿ	Ɀ	Rci
s	Ɀ	Ɀ	Ɀ	Slovo
t	Ɀ	Ɀ	Ɀ	Tverdo
^u	Ɀ	Ɀ	Ɀ	Uk
f	Ɀ	Ɀ	Ɀ	Fert
x	Ɀ	Ɀ	Ɀ	ch
^o	Ɀ	Ɀ	Ɀ	Ó
c	Ɀ	Ɀ	Ɀ	Tsi
q	Ɀ	Ɀ	Ɀ	Tšerv'
w	Ɀ	Ɀ	Ɀ	Ša
^w	Ɀ	Ɀ	Ɀ	Šta

^p	ᵕ	^P	ᵕ		Jer
#.^p	ᵕ	#.^P	ᵕ	y	Jery
^b	ᵝ	^B	ᵝ		Jerek
#.^b	ᵝ	#.^B	ᵝ	y	Jery
^y	ᵞ	^Y	ᵞ	ě	Jet'
#.a	ᵠ	#.A	ᵠ	ja	Ja
#.o	ᵡ	#.O	ᵡ	ju	Ju
#.^g	ᵢ	#.^G	ᵢ	je	Je
^e	ᵣ	^E	ᵣ	ę	Ęs
^a	ᵤ	^A	ᵤ	ą	Ąs
#.^e	ᵤ	#.^E	ᵤ	ję	Jęs
#.^a	ᵤ	#.^A	ᵤ	ją	Jąs
^x	ᵥ	^X	ᵥ	ks	Ksi
^c	ᵗ	^C	ᵗ	ps	Psi
^f	ᵘ	^F	ᵘ	θ	Fita
u	ᵙ	U	ᵙ	ÿ	Ižica

## Phonetische Zeichen

Die Bezeichnung der phonetischen Zeichen lehnt sich an den "Phonetic Symbol Guide" von Geoffrey K. Pullum and William A. Ladusaw (Chicago 1986) an. Ist ein Buchstabenname Bestandteil dieser Bezeichnung, so ist der Name des Buchstabens jeweils nach vorne gestellt und diese Umstellung durch ein Komma hinter dem Namen kenntlich gemacht.

Diese Bezeichnung ist erweitert um den Versuch einer Benennung nach der Systematik der IPA (International Phonetic Association). Die dabei gebrauchten Abkürzungen bzw. Begriffe sind:

erste Stelle:

hv	Halb-Vokal
k	Konsonant
v	Vokal

Vokale:

hi	high
sem-hi	semi-high (lower high)
up-mid	upper-mid (higher mid)
mid	mid
lo-mid	lower-mid
sem-lo	semi-low (higher low)
low	low
back	back
ce	central
fr	front
r	rounded
unr	unrounded

Konsonanten:

affr	affricate	lb-pal	labial-palatal
alv	alveolar	lb-vel	labiovelar
alv-pal	alveolo-palatal	lat	lateral
appr	approximant	med-appr	median approximant
asp	aspirated	nas	nasal
bil	bilabial	pal	palatal
click	click	pal-alv	palato-alveolar
col	coloration	phar	pharyngal
db-fr	doubly articulated fricative	pl	plosive
dent	dental	post-alv	post-alveolar
flap	flap	rfl	retroflex
fric	fricative	tap	tap
gl	glottal	trill	trill
impl	implosive	uvu	uvular
intd	interdental	v+	voiced
lab	labial	v-	voiceless
lb-den	labiodental	vel	velar

Anfang bzw. Ende Phonetisch: #P+ bzw. #P-

i	i	i, lower case	v hi fr unr
I	ɪ	i, small captial	
^I	ɪ	iota	v sem-hi fr unr
^i	ï	i, barred	v hi ce unr
‰:i	ï	i umlaut	v hi ce unr
y	y	y, lower case	v hi fr r
Y	ɣ	y, small capital	v sem-hi fr r
u	u	u, lower case	v hi back r
U	ʊ	upsilon	v sem-hi back r
^u	ʉ	u barred	v hi ce r
‰:u	ü	u umlaut	v hi ce r
^m	ʉ	m, turned	v hi back unr
^W	ω	omega, closed	v sem-hi back r
e	e	e, lower case	v up-mid fr unr
E	ɛ	epsilon	v lo-mid fr unr
^0	◌ <sup>◌</sup>	schwa, superscript	
^e	ə	schwa	v mid ce unr
^E	ɜ	epsilon, reversed	v lo-mid ce unr
9	ɘ	e, reversed	v up-mid ce unr
3	æ	ash	v sem-lo fr unr
‰:3	æ̈	ash umlaut	v sem-lo back unr
o	o	o, lower case	v up-mid back r
O	ɔ	o, open	v lo-mid back r
^o	ɣ	gamma, baby	v up-mid back unr
^O	ʌ	v, inverted	v lo-mid back unr
‰:o	ö	o umlaut	v up-mid fr r
‰:O	ö	o umlaut, open	v lo-mid fr r
q	ø	o, slashed	v up-mid fr r
Q	œ	o-e ligature	v lo-mid fr r
^q	ø	o, barred	v up-mid ce r
^Q	ɐ	epsilon, closed reversed	v lo-mid ce r
a	a	a, lower case	v low fr unr



A	a	a, script	v low back unr
^a	æ	o-e ligature, small capital	v lo-mid fr r
^A	ʌ	a, turned script	v low back r
^3	ɐ	a, turned	v sem-lo ce unr
^G	ɜ	a, inverted	v sem-lo ce unr
^v	ɹ	r with right tail, turned	hv med-appr rfl
V	ʊ	v, script	hv med-appr lb-den
^y	ɥ	h, turned	hv med-appr lb-pal
^M	ɥ	m with long right leg, turned	hv med-appr vel
^R	ɹ	r, turned	hv med-appr post-alv
w	w	w, lower case	hv med-appr lb-vel
%;w	Ẃ	w umlaut	hv med-appr lb-vel ce
p	p	p, lower case	k pl bil v-
b	b	b, lower case	k pl bil v+
^p	φ	phi	k fric bil v-
^b	β	beta	k fric bil v+
B	ʙ	b, hooktop	k impl bil
t	t	t, lower case	k pl alv v-
d	d	d, lower case	k pl alv v+
T	θ	theta	k fric intd v-
^t	ð	eth	k fric intd v+
^T	ʈ	t-esh ligature	k affr pal-alv v-
^D	ɖ	d-yogh ligature	k affr pal-alv v+
^d	ɖ	d, right-tail	k pl rfl v+
D	ɖ	d, hooktop	k impl alv
^K	ʈ	t with right tail	k pl rfl v-
c	c	c, lower case	k pl pal v-
C	ɟ	j, barred dotless	k pl pal v+
^c	ç	c cedilla	k fric pal v-
k	k	k, lower case	k pl vel v-
^k	q	q, lower case	k pl uvu v-
g	g	g, lower case	k pl vel v+

G	ɡ	g, small capital	k pl uvu v+
^g	ɡ̊	g, hooktop	k impl vel
ʔ	ʔ	glottal stop	k pl gl v-
f	f	f, lower case	k fric lb-den v-
v	v	v, lower case	k fric lb-den v+
^P	ɾ	r with long leg	k alv fric trill
s	s	s, lower case	k fric alv v-
z	z	z, lower case	k fric alv v+
S	ʃ	esh	k fric pal-alv v-
Z	ʒ	yogh	k fric pal-alv v+
^s	ɕ	c, curly-tail	k fric alv-pal v-
^z	ʐ	z, curly-tail	k fric alv-pal v+
6	ʃ̥	esh, curly-tail	k db-fr pal-alv v-
^6	ʒ̥	yogh, curly-tail	k db-fr pal-alv v+
^S	ʃ̥	s with right tail	k fric rfl v-
^Z	ʐ̥	z with right tail	k fric rfl v+
j	j	j, lower case	k fric pal v+
^j	ᵹ	j, superscript	k col j
x	x	x, lower case	k fric vel v-
^x	ɣ	gamma	k fric vel v+
X	χ	chi	k fric uvu v-
^X	ɣ̥	r, inverted small capital	k fric uvu v+
h	h	h, lower case	k fric gl v-
H	ɦ	heng, hooktop	k db-fr vel
^h	ᵹ	h, superscript	k asp
^H	ɦ̊	h, hooktop	k fric gl v+
^F	ħ	h, crossed	k fric phar v-
^?	ʕ	glottal stop, reversed	k fric phar v+
m	m	m, lower case	k nas bil
M	ɱ	m with leftward tail at right	k nas lb-den
n	n	n, lower case	k nas alv
^n	ɳ	eng	k nas vel
N	N	n, small capital	k nas uvu

^N	ɲ	n with leftward hook at left	k nas pal
^9	ɳ	n with right tail	k nas rfl
1	l	l, lower case	k lat alv appr v+
^Y	ɻ	y, turned	k lat pal appr
^1	ɫ	l with tilde	k lat vel alv appr
1	ɭ	l, belted	k lat alv fric v-
L	ɭ	l with right tail	k lat rfl appr
^L	ɮ	l-yogh ligature	k lat alv fric v+
r	r	r, lower case	k alv trill
^r	ɾ	r, fish-hook	k alv tap
R	ʀ	r, small capital	k uvu trill
P	ɽ	r with right tail	k rfl flap
^V	ɽ	r, turned long-legged	k alv lat flap
4	ʀ	r, superscript	k col r
^w	w	w, subscript	k lab
W	ʍ	w, inverted	k db-fr lb-vel
0	◉	bull 's eye	k click bil
5	ɠ	glottal stop, inverted	k click lat alv
^f	ɸ	t, turned	k click dent
^C	ɕ	C, stretched	k click post-alv
"	"	quotation marks	
(	(	left parenthesis	
)	)	right parenthesis	
[	[	left sqare bracket	
]	]	right square bracket	
<	<	left angle bracket	
>	>	right angle bracket	
{	{	left curly bracket	
}	}	right curly bracket	
/	/	slash	
=	=	equals	
'	'	apostrophe	

---

^'	'	apostrophe, reversed
*	*	asterisk
,	,	comma
^@	Ꞁ	comma turned
^7	┌	corner, left
7	┐	corner, right
.	.	full stop
^.	˘	half-length mark
:	:	length mark
^8	ˆ	ligature, top
8	˜	ligature, bottom
^&	⌢	ligature, top and bottom
2	˘	lowering sign
-	-	minus sign
		pipe (lower case height)
^\		pipe (upper case height)
^		pipe (full height)
+	+	plus sign
^2	ˆ	raising sign
;	;	semicolon
^!	ı	stroke (inferior), vertical
!	ˆ	stroke (superior), vertical
^%	~	tilde
%'	◌́	nucleus stress
%)	◌̆	breve
%*	◌̊	over-ring
%+	◌ <sup>+</sup>	plus sign, superscript
%-	◌̄	macron
%.	◌̇	over-dot
%:	◌̈	umlaut
%/	◌́	acute accent
%\	◌̀	grave accent
%<	◌̂	circumflexe

%>	ö	wedge
%=	ö	mid-central sign, superscript
%?	õ	tilde, superscript
%[	õ	bridge, superscript
%{	ó	half-ring superscript, left
%}	ó	half-ring superscript, right
%!	ɔ	palatalization hook
%!!	ɔ	right hook
%''	ɔ	syllabicity mark
%**	ɔ	under-ring
%++	ɔ	plus sign, subscript
%--	ɔ	under-bar
%..	ɔ	under-dot
%;	ɔ	cedilla
%;;	ɔ	polish hook
%>>	ɔ	wedge, subscript
%==	ɔ	mid-central sign, subscript
%??	ɔ	tilde, subscript
%[[	ɔ	bridge, subscript
%{{	ɔ	half-ring, subscript left
%}}	ɔ	half-ring, subscript right
%[-	ɔ	front sign, subscript
%-]	ɔ	back sign, subscript
%[	ɔ	laminal sign, subscript
% -	ɔ	raising sign, subscript
%-	ɔ	lowering sign, subscript
%%(	oo	ligature, long, top
%%-	oo	macron, long
%%)	oo	ligature, long, bottom
%%++	oo	plus sign, subscript
%%--	oo	underbar, long
%% -	oo	raising sign, subscript
%%-	oo	lowering sign, subscript

## Auszeichnungen, Schriftumschaltungen, Druckeffekte

Anfang bzw. Ende <b>Fett</b> druck:	#F+ bzw. #F-
Anfang bzw. Ende <i>Kursiv</i> :	#/+ bzw. #/-
Anfang bzw. Ende Sperrung:	#S+ bzw. #S-
Anfang bzw. Ende KAPITÄLCHEN:	#K+ bzw. #K-
Anfang bzw. Ende Arabisch:	#A+ bzw. #A-
Anfang bzw. Ende Cyrillisch (Alt-Kirchenslav.):	#C+ bzw. #C-
Anfang bzw. Ende Griechisch:	#G+ bzw. #G-
Anfang bzw. Ende Hebräisch:	#H+ bzw. #H-
Anfang bzw. Ende Koptisch:	#T+ bzw. #T-
Anfang bzw. Ende Phonetisch:	#P+ bzw. #P-
Anfang bzw. Ende Russisch (Kyrillisch):	#R+ bzw. #R-
Anfang bzw. Ende Syrisch:	#Y+ bzw. #Y-
Anfang bzw. Ende <b>Mittellinie</b> :	#0+ bzw. #0-
Anfang bzw. Ende <u>einfache Unterstreich</u> ung:	#1+ bzw. #1-
Anfang bzw. Ende <u>doppelte Unterstreich</u> ung:	#2+ bzw. #2-
Anfang bzw. Ende <u>dicke Unterstreich</u> ung:	#3+ bzw. #3-
Anfang bzw. Ende <u>Unterpunkt</u> ierung:	#4+ bzw. #4-
Anfang bzw. Ende <u>tiefe Überstreich</u> ung:	#5+ bzw. #5-
Anfang bzw. Ende <u>hohe Überstreich</u> ung:	#6+ bzw. #6-
Anfang bzw. Ende <u>helle Unterleg</u> ung:	#7+ bzw. #7-
Anfang bzw. Ende <u>dunkle Unterleg</u> ung:	#8+ bzw. #8-
Ende aller Auszeichnungen:	#?-

## Hoch- und Tiefstellungen

Hochstellen um $\frac{1}{3}$ Zeile über die Grundlinie:	#H:
Tiefstellen um $\frac{1}{3}$ Zeile unter die Grundlinie:	#T:
Hochstellen um $\frac{1}{2}$ Zeile über die Grundlinie:	#O:
Tiefstellen um $\frac{1}{2}$ Zeile unter die Grundlinie:	#U:
Aufheben der Hoch- bzw. Tiefstellung (Grundlinie):	#G:

Einzelne Zeichen (aus dem ASCII-Zeichensatz) können auch durch "#'" bzw. "#," vor dem betreffenden Zeichen um  $\frac{1}{3}$  Zeile (ggf. zusätzlich) hoch- bzw. tiefgestellt werden.

## Alphabetische Liste der Sonderzeichen

%<	ô	Accent circonflexe (Akzent über d. Buchstaben)
%<<	ò	Accent circonflexe (Akzent unter d. Buchstaben)
%/	ó	Accent aigu (Akzent über d. Buchstaben)
%//	õ	Accent aigu (Akzent unter d. Buchstaben)
%\	ò	Accent grave (Akzent über d. Buchstaben)
%\\	õ	Accent grave (Akzent unter d. Buchstaben)
#.^a, #.ä	æ	ae-Ligatur
#.^A, #.Ä	Æ	AE-Ligatur
# (AEH)	~	ähnlich
#. (	ˆ	Ajin, Ain (Transkriptionszeichen)
%/	ó	Akut (Akzent über d. Buchstaben)
%//	õ	Akut (Akzent unter d. Buchstaben)
#.)	ˆ	Alef (Transkriptionszeichen)
#.%	ˆ	Alef, Doppel- (Transkriptionszeichen)
#.^d	ð	altenglisches Eth (klein)
#.^D	Ð	altenglisches Eth (groß)
#.z	z	altenglisches Yogh (klein)
#.Z	Ʒ	altenglisches Yogh (groß)
# (AU/)	□	Anfang einer Auslassung
# (ER/)	⌈	Anfang einer Ersetzung
# (ER./)	⌈	Anfang einer Ersetzung (mit Punkt)
# (UM/)	⌈	Anfang einer Umstellung
# (UM./)	⌈	Anfang einer Umstellung (mit Punkt)
"	"	Anführungszeichen
#. '	”	Anführungszeichen oben
#. ,	„	Anführungszeichen unten
#. >	»	Anführungszeichen, doppelt
#. <	«	Anführungszeichen, doppelt
#. :	>	Anführungszeichen, einfach
#. ;	<	Anführungszeichen, einfach
'	'	Apostroph (Hochkomma)

%,	◌́	Apostroph, Komma (Akzent über d. Buchstaben)
^'	◌̀	Apostroph, umgekehrter
# (AU)	◌°	Auslassung
# (/AUL)	◌⁄	Auslassung (links), Ende einer
# (/AUR)	◌\	Auslassung (rechts), Ende einer
# (AU/)	◌◻	Auslassung, Anfang einer
!	!	Ausrufezeichen
^!	¡	Ausrufezeichen, einleitend (spanisch)
—		Ausschluss, fester
%-]	◌₪	back sign, subscript
^\ # (BSL)	◌\ ◌\ ◌\ ◌\ ◌\ ◌\ ◌\ ◌◌	Backslash Backslash Backslash Balken, Querstrich (Akzent über d. Buchstaben) Balken, Querstrich (Akzent unter d. Buchstaben) Betonungszeichen (über dem Buchstaben) Betonungszeichen (unter dem Buchstaben) Blank, festes (fester Ausschluss) Bogen (unten offen) (Akzent über d. Buchstaben) Bogen (unten offen) (Akzent unter d. Buchstaben) Box: Ecke Mitte links Box: Ecke Mitte Mitte Box: Ecke Mitte rechts Box: Ecke oben links Box: Ecke oben Mitte Box: Ecke oben rechts Box: Ecke unten links Box: Ecke unten Mitte Box: Ecke unten rechts Brücke (über dem Buchstaben) Brücke (unter dem Buchstaben) Brücke, umgekehrte (über dem Buchstaben) Brücke, umgekehrte (unter dem Buchstaben) Cedille (Akzent unter d. Buchstaben)
%-	◌ō	Balken, Querstrich (Akzent über d. Buchstaben)
%--	◌◌	Balken, Querstrich (Akzent unter d. Buchstaben)
%'	◌◌́	Betonungszeichen (über dem Buchstaben)
%''	◌◌̀	Betonungszeichen (unter dem Buchstaben)
—		Blank, festes (fester Ausschluss)
% (	◌◌́	Bogen (unten offen) (Akzent über d. Buchstaben)
% ( (	◌◌̀	Bogen (unten offen) (Akzent unter d. Buchstaben)
# (BML)	◌┌	Box: Ecke Mitte links
# (BMM)	◌├	Box: Ecke Mitte Mitte
# (BMR)	◌┐	Box: Ecke Mitte rechts
# (BOL)	◌└	Box: Ecke oben links
# (BOM)	◌┘	Box: Ecke oben Mitte
# (BOR)	◌┘	Box: Ecke oben rechts
# (BUL)	◌└	Box: Ecke unten links
# (BUM)	◌┘	Box: Ecke unten Mitte
# (BUR)	◌┘	Box: Ecke unten rechts
% [	◌ō	Brücke (über dem Buchstaben)
% [ [	◌◌	Brücke (unter dem Buchstaben)
% ]	◌ö	Brücke, umgekehrte (über dem Buchstaben)
% ] ]	◌◌	Brücke, umgekehrte (unter dem Buchstaben)
%;	◌ç	Cedille (Akzent unter d. Buchstaben)



# (CE)	¢	Cent-Zeichen
# (C)	©	Copyright
# . d	đ	d mit Querstrich (serbokroatisches d)
# . D	Đ	D mit Querstrich (serbokroatisches D)
% <	ô	Dach, Zirkumflex (Akzent über d. Buchstaben)
% <<	ò	Dach, Zirkumflex (Akzent unter d. Buchstaben)
# . o	ø	dänisches ö (o mit Schrägstrich)
# . O	Ø	dänisches Ö (O mit Schrägstrich)
# (DIV)	÷	Division
^ \$	\$	Dollarzeichen
# (DO)	\$	Dollarzeichen
% "	ö	Doppelakut (Akzent über d. Buchstaben)
% "'	ò	Doppelakut (Akzent unter d. Buchstaben)
# . %	ʒ	Doppel-Alef (Transkriptionszeichen)
:	:	Doppelpunkt
# (DP)	”	doppelt gestrichen
# . >	»	doppelte Anführungszeichen
# . <	«	doppelte Anführungszeichen
# . '	”	doppelte Anführungszeichen oben
# . ,	„	doppelte Anführungszeichen unten
# (DEA)	⌈	doppelte eckige Klammer auf
# (DEAG)	⌈	doppelte eckige Klammer auf (groß)
# (DEZ)	⌋	doppelte eckige Klammer zu
# (DEZG)	⌋	doppelte eckige Klammer zu (groß)
# . /	∥	doppelter senkrechter Strich
# (DSS)	∥	doppelter senkrechter Strich
# (DF)	””	dreifach gestrichen
# (DM)	∩	Durchschnittsmenge
# (DMG)	∩	Durchschnittsmenge (groß)
# (EOM)	⊃	echte Obermenge
# (ETM)	⊂	echte Teilmenge
[	[	eckige Klammer auf
# (EAG)	[	eckige Klammer auf (groß)

# (DEA)	⌈	eckige Klammer auf, doppelt
# (DEAG)	⌈	eckige Klammer auf, doppelt (groß)
]	]	eckige Klammer zu
# (EZG)	⌋	eckige Klammer zu (groß)
# (DEZ)	⌋	eckige Klammer zu, doppelt
# (DEZG)	⌋	eckige Klammer zu, doppelt (groß)
# (EF)	'	einfach gestrichen
# . :	>	einfaches Anführungszeichen
# . ;	<	einfaches Anführungszeichen
# (EI)	⊢	Einfügung
# (EI .)	⊢	Einfügung (mit Punkt)
^!	¡	einleitendes Ausrufezeichen (spanisch)
^?	¿	einleitendes Fragezeichen (spanisch)
# (EL)	∈	Element
# (/AUL)	/	Ende einer Auslassung (links)
# (/AUR)	\	Ende einer Auslassung (rechts)
# (/ER)	⌋	Ende einer Ersetzung
# (/ER .)	⌋	Ende einer Ersetzung (mit Punkt)
# (/UM)	↷	Ende einer Umstellung
# (/UM .)	↷	Ende einer Umstellung (mit Punkt)
# (ENT)	≅	entspricht
# (ERL)	⌈	Ersetzung (links)
# (ERL .)	⌈	Ersetzung (links, mit Punkt)
# (ER . /)	⌈	Ersetzung (mit Punkt), Anfang einer
# (/ER .)	⌋	Ersetzung (mit Punkt), Ende einer
# (ERR)	⌋	Ersetzung (rechts)
# (ERR .)	⌋	Ersetzung (rechts, mit Punkt)
# (ER /)	⌈	Ersetzung, Anfang einer
# (/ER)	⌋	Ersetzung, Ende einer
# (EG)	∨	es gibt
# (EGE)	∃	es gibt (umgedrehtes E)
^&	&	Et-Zeichen
# . ^d	ð	Eth, isländisch / altenglisch (klein)

#. ^D	Ð	Eth, isländisch / altenglisch (groß)
# (E)	€	Euro-Zeichen
# (FUE)	∞	fast unendlich
—		festes Blank (fester Ausschluss)
?	?	Fragezeichen
^?	¿	Fragezeichen, einleitend (spanisch)
;	;	Fragezeichen, griechisches (innerhalb #G+ und #G-)
?	;	Fragezeichen, griechisches (innerhalb #G+ und #G-)
%[-	ϕ	front sign, subscript
# (FA)	∧	für alle
# (FAA)	∀	für alle (umgedrehtes A)
# (GES)	∩	geschnitten mit
{	{	geschweifte Klammer auf
# (GAG)	{	geschweifte Klammer auf (groß)
}	}	geschweifte Klammer zu
# (GZG)	}	geschweifte Klammer zu (groß)
=	=	Gleichheitszeichen
#. *	°	Grad
%\	ò	Gravis (Akzent über d. Buchstaben)
%\\	ó	Gravis (Akzent unter d. Buchstaben)
;	;	griechisches Fragezeichen (innerhalb #G+ und #G-)
?	;	griechisches Fragezeichen (innerhalb #G+ und #G-)
!	·	griechisches Semikolon (innerhalb #G+ und #G-)
# (GR)	>	größer
# (GGL)	≧	größer gleich (mit -)
# (GGLD)	≧	größer gleich (mit =)
#. ^D	Ð	großes Eth (isländisch, altenglisch)
#. P	Þ	großes Thorn (isländisch)
#. Z	Ʒ	großes Yogh (altenglisch)
%>	ö	Haček, Haken (Akzent über d. Buchstaben)
%>>	ǒ	Haček, Haken (Akzent unter d. Buchstaben)
# (HSS)	┌	Halber senkrechter Strich (oben)
%)	ö	Halbkreis (oben offen) (Akzent über d. Buchst.)

---

%) )	◊	Halbkreis (oben offen) (Akzent unter d. Buchst.)
# (VH)	∞	Heirat (verheiratet)
# ' x	×	hochgestellte Zeichen
^1	¹	hochgestellte Ziffern, kleine
'	'	Hochkomma (Apostroph)
# . i	ı	i ohne Punkt (türkisches i)
# (ID)	≡	identisch
# . j	ij	ij-Ligatur
# . J	IJ	IJ-Ligatur
# (INT)	∫	Integral
# (INTG)	∫	Integral (groß)
# (IW)	₺	internationales Währungszeichen
# (IP)	:	Interpunktionszeichen
# . ^d	ð	isländisches Eth (klein)
# . ^D	Ð	isländisches Eth (groß)
# . p	þ	isländisches Thorn (klein)
# . P	Þ	isländisches Thorn (groß)
# (KP)	×	kartesisches Produkt
# (KPG)	⊗	kartesisches Produkt (groß)
^@	@	Klammeraffe
^1	¹	kleine hochgestellte Ziffern
# (KL)	<	kleiner
# (KGL)	≅	kleiner gleich (mit -)
# (KGLD)	≅	kleiner gleich (mit =)
# . ^d	ð	kleines Eth (isländisch, altenglisch)
# . p	þ	kleines Thorn (isländisch)
# . z	ȝ	kleines Yogh (altenglisch)
,	,	Komma
%, ,	◌,	Komma (Akzent unter d. Buchstaben)
%,	◌,	Komma, Apostroph (Akzent über d. Buchstaben)
# (KGR)	≈	kongruent (mit -)
# (KGRD)	≅	kongruent (mit =)
# (KRS)	○	Kreis

%*	◌̇	Kringel, Ringel(chen) (Akzent über d. Buchst.)
%**	◌̈	Kringel, Ringel(chen) (Akzent unter d. Buchst.)
%; ;	◌̣	Krummhaken, Ogonek (Akzent unter d. Buchstaben)
# (MKLS)	≋	kurze oder lange Silbe (Metrik)
# (MKS)	˘	kurze Silbe (Metrik)
# . l	ł	l mit Querstrich (polnisches l)
# . L	Ł	L mit Querstrich (polnisches L)
% [ ]	◌̚	laminal sign, subscript
# (MLS)	–	lange Silbe (Metrik)
^		langer senkrechter Strich
^ _	–	langes Minuszeichen
# . s	ſ	langes s
# . z	z	langes z
# (LM)	∅	leere Menge
# . ^ a, # . ä	æ	Ligatur ae
# . ^ A, # . Ä	Æ	Ligatur AE
# . j	ij	Ligatur ij
# . J	IJ	Ligatur IJ
# . ^ o, # . ö	œ	Ligatur oe
# . ^ O, # . Ö	Œ	Ligatur OE
# (LN)	¬	logisches Nicht
# (LO)	∨	logisches Oder
# (LU)	∧	logisches Und
% -	◌̣	lowering sign, subscript
^ *	×	Malkreuz
# (MKR)	×	Malkreuz
^ .	·	Malpunkt
# (MPU)	·	Malpunkt
# (MC)	℄	Menge C
# (MN)	ℵ	Menge N
# (MQ)	ℚ	Menge Q
# (MR)	ℝ	Menge R
# (MZ)	ℤ	Menge Z

---

# (LM)	∅	Menge, leere
# (MKLS)	≈	Metrik: Kurze oder lange Silbe
# (MKS)	∨	Metrik: Kurze Silbe
# (MLS)	–	Metrik: Lange Silbe
# (MPL)	∓	Minus-Plus
–	–	Minuszeichen
^–	–	Minuszeichen, lang
# . –	′	Minute
# (NBL)	∇	Nabla-Operator
# (LN)	¬	Negationszeichen (logisches Nicht)
# (NEL)	∉	nicht Element
# (NID)	≠	nicht identisch (mit /)
# (NIDS)	≠	nicht identisch (mit )
# (NOM)	⊄	nicht Obermenge
# (NTM)	⊈	nicht Teilmenge
# (LN)	¬	Nicht, logisches
^#	#	Nummernzeichen
# . o	ø	o mit Schrägstrich (dänisches ö)
# . O	Ø	O mit Schrägstrich (dänisches Ö)
# (EOM)	⊃	Obermenge, echte
# (UOM)	⊇	Obermenge, unechte
# (LO)	∨	Oder, logisches
# . ^ o, # . ö	œ	oe-Ligatur
# . ^ O, # . Ö	Œ	OE-Ligatur
% ; ;	◌̣	Ogonek, Krummhaken (Akzent unter d. Buchstaben)
# . !	§	Paragraphzeichen
# (PA)	§	Paragraphzeichen
# (PFB)	↔	Pfeil nach beiden Seiten
# (PFBD)	↔	Pfeil nach beiden Seiten (doppelt)
# (PFL)	←	Pfeil nach links
# (PFLD)	⇐	Pfeil nach links (doppelt)
# (PFLO)	↖	Pfeil nach links oben
# (PFLU)	↙	Pfeil nach links unten

# (PFO)	↑	Pfeil nach oben
# (PFR)	→	Pfeil nach rechts
# (PFRD)	⇒	Pfeil nach rechts (doppelt)
# (PFRO)	↗	Pfeil nach rechts oben
# (PFRU)	↘	Pfeil nach rechts unten
# (PFU)	↓	Pfeil nach unten
# (PFLR)	⇔	Pfeile nach links / rechts
# (PFOO)	↑↑	Pfeile nach oben / oben
# (PFOU)	↑↓	Pfeile nach oben / unten
# (PF)	£	Pfund-Zeichen
# (PLM)	±	Plus-Minus
%+	⊕	Plus (über dem Buchstaben)
%++	⊕	Plus (unter dem Buchstaben)
+	+	Pluszeichen
# .l	ł	polnisches l (l mit Querstrich)
# .L	Ł	polnisches L (L mit Querstrich)
# (PR)	Π	Produkt
# (PRG)	∏	Produkt (groß)
# (PRM)	‰	Promillezeichen
^%	%	Prozentzeichen
# (P)	®	Published
.	.	Punkt
%.	◌̇	Punkt (Akzent über d. Buchstaben)
%. .	◌̈	Punkt (Akzent unter d. Buchstaben)
# (QUA)	□	Quadrat
%-	◌̄	Querstrich, Balken (Akzent über d. Buchstaben)
%--	◌̅	Querstrich, Balken (Akzent unter d. Buchstaben)
% -	◌̆	raising sign, subscript
# (R)	®	Registered
%*	◌̇	Ringel(chen), Kringel (Akzent über d. Buchst.)
%**	◌̈	Ringel(chen), Kringel (Akzent unter d. Buchst.)
^=	●	Rückweisungszeichen
(	(	runde Klammer auf

---

# (RAG)	(	runde Klammer auf (groß)
)	)	runde Klammer zu
# (RZG)	)	runde Klammer zu (groß)
^s, ß	ß	scharfes s (#K+ß #K- → ss)
^S	SS	scharfes S (→ SS)
#.^s, ß	ß	scharfes s (#K+#.ß #K- → ß)
#.^S	ß	scharfes S (→ ß)
^"	■	Schmierzeichen
/	/	Schrägstrich
#.=	"	Sekunde
!	·	Semikolon, griechisches (innerhalb #G+ und #G-)
# (SR)	⊥	senkrecht
		senkrechter Strich
#./		senkrechter Strich, doppelter
# (DSS)		senkrechter Strich, doppelter
# (HSS)	⌈	senkrechter Strich (oben), Halber
^		senkrechter Strich, lang
# (USS)	‡	senkrechter Strich, unterbrochener
#.d	đ	serbokroatisches d (d mit Querstrich)
#.D	Đ	serbokroatisches D (D mit Querstrich)
^!	¡	spanisches Ausrufezeichen
^?	¿	spanisches Fragezeichen
^<	<	spitze Klammer auf
^>	>	spitze Klammer zu
# (SA)	<	spitze Klammer auf
# (SZ)	>	spitze Klammer zu
^+	†	Sterbekreuz
*	*	Stern
#./		Strich, doppelter senkrechter
# (DSS)		Strich, doppelter senkrechter
		Strich, senkrechter
;	;	Strichpunkt
# (SUM)	Σ	Summe



# (SUMG)	$\sum$	Summe (groß)
# (ETM)	$\subset$	Teilmenge, echte
# (UTM)	$\subseteq$	Teilmenge, unechte
# .p	þ	Thorn, isländisch (klein)
# .P	Þ	Thorn, isländisch (groß)
# , x	×	tiefgestellte Zeichen
%?	ö	Tilde (Akzent über d. Buchstaben)
%??	o	Tilde (Akzent unter d. Buchstaben)
# (TM)	™	Trade Mark
%:	ö	Trema (Akzent über d. Buchstaben)
%::	o	Trema (Akzent unter d. Buchstaben)
# .i	ı	türkisches i (i ohne Punkt)
# ; x	ö	übersetzte Zeichen
%]	ö	umgekehrte Brücke (über dem Buchstaben)
%]]	o	umgekehrte Brücke (unter dem Buchstaben)
^'	ı	umgekehrter Apostroph
^a, ä	ä	Umlaut ä
^A, Ä	Ä	Umlaut Ä
^o, ö	ö	Umlaut ö
^O, Ö	Ö	Umlaut Ö
^u, ü	ü	Umlaut ü
^U, Ü	Ü	Umlaut Ü
# (UM./)	ˆ	Umstellung (mit Punkt), Anfang einer
# (/UM.)	ˆ	Umstellung (mit Punkt), Ende einer
# (UM/)	ˆ	Umstellung, Anfang einer
# (/UM)	ˆ	Umstellung, Ende einer
# (LU)	^	Und, logisches
# (UOM)	$\supseteq$	unechte Obermenge
# (UTM)	$\subseteq$	unechte Teilmenge
# (UE)	$\infty$	unendlich
# (UGF)	$\approx$	ungefähr
# (UGR)	$\gtrsim$	ungefähr größer
# (UKL)	$\lesssim$	ungefähr kleiner

---

# (UGL)	≠	ungleich (mit /)
# (UGLS)	≠	ungleich (mit !)
# (USS)	⋮	unterbrochener senkrechter Strich
# !x	⊘	untergesetzte Zeichen
^_	—	Unterstreichungsstrich
# (VER)	∪	vereinigt mit
# (VM)	∪	Vereinigungsmenge
# (VMG)	⊃	Vereinigungsmenge (groß)
# (VH)	∞	verheiratet, Heirat
# . .	→	Verweispfeil
# (IW)	¤	Währungszeichen, internationales
# (WI)	∠	Winkel
# . [	┌	Winkel links oben
# . ]	┐	Winkel rechts oben
# (WU)	√	Wurzel
# (WUG)	√	Wurzel (groß)
# . z	ƶ	Yogh, altenglisch (klein)
# . Z	Ʒ	Yogh, altenglisch (groß)
^1	¹	Ziffern, klein hochgestellt
%<	◌̂	Zirkumflex, Dach (Akzent über d. Buchstaben)
%<<	◌̃	Zirkumflex, Dach (Akzent unter d. Buchstaben)
# (ZWR)	—	Zwischenraum, Zeichen für

# Code-Tabellen

---

Allgemeines . . . . .	813
Tabellen für den ASCII-Code . . . . .	815
1. Interner TUSTEP-Code . . . . .	815
2. Internationaler ASCII-Code zum Vergleich . . . . .	815
3. Deutsche Variante des ASCII-Codes . . . . .	816
4. Code IBMPC: ASCII mit Umlauten aus CP437/CP850 . . . . .	817
5. Code CP437: PC United States . . . . .	818
6. Code CP850: PC Multilingual . . . . .	819
7. Code CP852: PC Slavic . . . . .	820
8. Code CP1250: Windows Central European . . . . .	821
9. Code CP1252: Windows Western . . . . .	822
10. Code CP10000: Macintosh Roman . . . . .	823
11. Code CP10029: Macintosh Central European . . . . .	824
12. Code DECMCS: VT100-Terminal . . . . .	825
13. Code ISO8859: Latin-1 . . . . .	826
14. Code LINUX: Linux-Konsole . . . . .	827
15. Code LATIN1: ISO 8859-1 . . . . .	828
16. Code LATIN2: ISO 8859-2 . . . . .	829
17. Code LATIN3: ISO 8859-3 . . . . .	830
18. Code LATIN4: ISO 8859-4 . . . . .	831
19. Code LATIN5: ISO 8859-9 . . . . .	832
20. Code LATIN6: ISO 8859-10 . . . . .	833
21. Code LATIN7: ISO 8859-13 . . . . .	834
22. Code LATIN8: ISO 8859-14 . . . . .	835
23. Code LATIN9: ISO 8859-15 . . . . .	836
24. Code LATIN10: ISO 8859-16 . . . . .	837
Tabellen für den EBCDIC-Code . . . . .	838
1. Interner TUSTEP-Code . . . . .	838
2. Internationaler EBCDIC-Code zum Vergleich . . . . .	838
3. Deutsche Variante des EBCDIC-Codes . . . . .	839
4. Code EBCDIC: US-EBCDIC . . . . .	839
TUSTEP-Umcodierung von ASCII nach EBCDIC . . . . .	840
TUSTEP-Umcodierung von EBCDIC nach ASCII . . . . .	840
Standard-Sortierfolge in TUSTEP . . . . .	841
Sonder-Sortierfolge in TUSTEP . . . . .	841
Umrechnungstabelle Hexadezimal – Dezimal . . . . .	842
Umrechnungstabelle Hexadezimal – Oktal . . . . .	843

## Allgemeines

a) zu dem in TUSTEP verwendeten Code

Für die interne Darstellung der Zeichen in TUSTEP wurde für die einzelnen Zeichen der internationale ASCII-Code (DIN 66003, Internationale Referenz-Version = 7-bit-Code nach der ISO-Norm 646) zugrunde gelegt.

Damit alle Akzentzeichen einheitlich codiert sind, wurde auf die Zeichen Gravis, Zirkumflex und Tilde (hexadezimal 60, 5E und 7E) als Akzente verzichtet. Jedoch wird das Zeichen Gravis als Markierungszeichen im Editorfenster verwendet und darf in Daten deshalb nicht vorkommen; das Zeichen Zirkumflex wird als Steuerzeichen verwendet.

Um den Speicher besser auszunutzen, wurde der 7-bit ASCII-Code auf 8 Bits erweitert. Zeichen, die bei der Dateneingabe mit vorangestelltem "^" markiert werden (z. B. ^a für ä), werden intern durch Setzen des achten Bits gekennzeichnet. Dazu gibt es einige Ausnahmen, die im Folgenden erläutert sind.

Nicht jeder interne Code, der durch Markieren eines Zeichens mit "^" erzeugt werden kann, entspricht immer einem druckbaren Zeichen. So ist z. B. "^m" im lateinischen Alphabet kein Zeichen zugeordnet, im hebräischen Alphabet steht "^m" für das Schluss-Mem.

Für die interne Darstellung der Zeichen in TUSTEP unter IBM-Betriebssystemen wurde der internationale EBCDIC-Code zugrunde gelegt. Dies ist der Code, der sich beim Umcodieren von ASCII nach EBCDIC ergibt, wenn die vom IBM-Betriebssystem MVS beim Lesen von ASCII-Bändern verwendete Umcodiertabelle verwendet wird (vgl. IBM: MVS/370 Magnetic Tape Labels and File Structure Administration).

b) zu den Tabellen

Die Tabellen sind für die Betriebssysteme, die den ASCII-Code bzw. den EBCDIC-Code verwenden, getrennt angegeben. Der ASCII-Code gilt für die TUSTEP-Versionen unter

Windows, Linux und macOS

und der EBCDIC-Code galt für die TUSTEP-Versionen unter

BS2000, MVS und VM/CMS

Die erste Tabelle gibt die Codes an, in der die Zeichen "TUSTEP-Zeichensatzes" (vgl. Seite 763) in den Dateien gespeichert werden. Die anderen Zeichen des TUSTEP-Zeichenvorrats werden durch Kombinationen der Zeichen des TUSTEP-Zeichensatzes gespeichert.

Die zweite Tabelle ist jeweils nur zur Information angegeben und hat für TUSTEP keine weitere Bedeutung.

In den weiteren Tabellen sind jeweils Codes angegeben, die für TUSTEP-Fenster mit dem Kommando #DEFINIERE (siehe Seite 132) eingestellt werden können. Sie werden u. U. auch beim Import von Daten aus Fremd-Dateien und beim Export von Daten in Fremd-Dateien mit dem Kommando #UMWANDLE (siehe Seite 237) zur Umcodierung der Zeichen verwendet. Auf den Code, in dem Daten in TUSTEP-Dateien abgespeichert werden, hat diese Code-Einstellung keinen Einfluss.

## Tabellen für den ASCII-Code

### 1. Interner TUSTEP-Code

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
00			SP	0	@	P		p				^0	^@	^P		^p	00
01			!	1	A	Q	a	q			^!	^1	^A	^Q	^a	^q	01
02			"	2	B	R	b	r			^"	^2	^B	^R	^b	^r	02
03			#	3	C	S	c	s			^#	^3	^C	^S	^c	^s	03
04			\$	4	D	T	d	t			^\$	^4	^D	^T	^d	^t	04
05			%	5	E	U	e	u			^%	^5	^E	^U	^e	^u	05
06			&	6	F	V	f	v			^&	^6	^F	^V	^f	^v	06
07			'	7	G	W	g	w			^'	^7	^G	^W	^g	^w	07
08			(	8	H	X	h	x			^(	^8	^H	^X	^h	^x	08
09			)	9	I	Y	i	y			)	^9	^I	^Y	^i	^y	09
0A			*	:	J	Z	j	z			^*	^:	^J	^Z	^j	^z	0A
0B			+	;	K	[	k	{			^+	^;	^K	^[	^k	^{	0B
0C			,	<	L	\	l				^,	^<	^L	^\	^l	^	0C
0D			-	=	M	]	m	}			^-	^=	^M	^]	^m	^}	0D
0E			.	>	N	^	n				^.	^>	^N		^n		0E
0F			/	?	O	_	o				^/	^?	^O	^_	^o		0F
	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	

### 2. Internationaler ASCII-Code zum Vergleich

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
00			SP	0	@	P		`	p								00
01			!	1	A	Q	a	q									01
02			"	2	B	R	b	r									02
03			#	3	C	S	c	s									03
04			\$	4	D	T	d	t									04
05			%	5	E	U	e	u									05
06			&	6	F	V	f	v									06
07			'	7	G	W	g	w									07
08			(	8	H	X	h	x									08
09			)	9	I	Y	i	y									09
0A			*	:	J	Z	j	z									0A
0B			+	;	K	[	k	{									0B
0C			,	<	L	\	l										0C
0D			-	=	M	]	m	}									0D
0E			.	>	N	^	n	~									0E
0F			/	?	O	_	o										0F
	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	

60=Gravis 7E=Tilde

## 3. Deutsche Variante des ASCII-Codes

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
00			SP	0	@	P	`	p									00
01			!	1	A	Q	a	q									01
02			"	2	B	R	b	r									02
03			#	3	C	S	c	s									03
04			\$	4	D	T	d	t									04
05			%	5	E	U	e	u									05
06			&	6	F	V	f	v									06
07			'	7	G	W	g	w									07
08			(	8	H	X	h	x									08
09			)	9	I	Y	i	y									09
0A			*	:	J	Z	j	z									0A
0B			+	;	K	Ä	k	ä									0B
0C			,	<	L	Ö	l	ö									0C
0D			-	=	M	Ü	m	ü									0D
0E			.	>	N	^	n	ß									0E
0F			/	?	O	_	o										0F
	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	

60=Gravis



## 4. Code IBMPC: ASCII mit Umlauten aus CP437/CP850

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
00			SP	0	@	P		p									00
01			!	1	A	Q	a	q	ü						ß		01
02			"	2	B	R	b	r									02
03			#	3	C	S	c	s									03
04			\$	4	D	T	d	t	ä	ö			-				04
05			%	5	E	U	e	u									05
06			&	6	F	V	f	v									06
07			'	7	G	W	g	w									07
08			(	8	H	X	h	x									08
09			)	9	I	Y	i	y		Ö							09
0A			*	:	J	Z	j	z		Ü							0A
0B			+	;	K	[	k	{									0B
0C			,	<	L	\	l										0C
0D			-	=	M	]	m	}									0D
0E			.	>	N	^	n		Ä								0E
0F			/	?	O	_	o										0F
	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	

Der Code "IBMPC" enthält

- alle Zeichen des ASCII-Zeichensatzes
- die deutschen Umlaute und das scharfe s
- das lange Minuszeichen und den langen senkrechten Strich.

## 5. Code CP437: PC United States

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
00			SP	0	@	P		p	Ç	É	á		⌞			≡	00
01		!	1	A	Q	a	q	ü	æ	í			⌞		ß	±	01
02		"	2	B	R	b	r	é	Æ	ó			⌞			≥	02
03		#	3	C	S	c	s	â	ô	ú			⌞			≤	03
04		\$	4	D	T	d	t	ä	ö	ñ		⌞	⌞				04
05		%	5	E	U	e	u	à	ò	Ñ		⌞	⌞				05
06		&	6	F	V	f	v	å	û							÷	06
07		'	7	G	W	g	w	ç	ù							≈	07
08		(	8	H	X	h	x	ê	ÿ	¿						°	08
09		)	9	I	Y	i	y	ë	ÿ	Ö				⌞			09
0A		*	:	J	Z	j	z	è	Ü	¬				⌞		·	0A
0B		+	;	K	[	k	{	ï	ç					⌞		√	0B
0C		,	<	L	\	l		î	£						∞		0C
0D		-	=	M	]	m	}	ì		ı							0D
0E		.	>	N	^	n		Ä		«							0E
0F		/	?	O	_	o		Å		»		⌞					0F
	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	

Der Code "CP437" enthält

- alle Zeichen des TUSTEP-Zeichensatzes, die auch in der Code Page 437 (siehe Windows Handbuch) definiert sind
- Akzentbuchstaben, die in der Code Page 437 definiert sind
- alle Zeichen, die mit den Steuerzeichen "# " und "# (name) " codiert werden und auch in der Code Page 437 definiert sind.

## 6. Code CP850: PC Multilingual

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
00		SP	0	@	P		p	Ç	É	á			Ł	Đ	Ó		00
01		!	1	A	Q	a	q	ü	æ	í			ł	Đ	ß	±	01
02		"	2	B	R	b	r	é	Æ	ó			Ł	Ê	Ô		02
03		#	3	C	S	c	s	â	ô	ú			Ł	Ë	Ò		03
04		\$	4	D	T	d	t	ä	ö	ñ			Ł	È	õ		04
05		%	5	E	U	e	u	à	ò	Ñ	À		Ł	É	Ö	§	05
06		&	6	F	V	f	v	å	û		Â		Ł	Í	÷		06
07		'	7	G	W	g	w	ç	ù		À		Ł	Î	þ		07
08		(	8	H	X	h	x	ê	ÿ	¿	©			Ł	Ï	º	08
09		)	9	I	Y	i	y	ë	ÿ	®				Ł	Ú		09
0A		*	:	J	Z	j	z	è	Û	¬				Ł	Û	·	0A
0B		+	;	K	[	k	{	ï	ø					Ł	Ü		0B
0C		,	<	L	\	l		î	£						Ł	Ý	0C
0D		-	=	M	]	m	}	ì	Ø	¡	¢				Ł	Ý	0D
0E		.	>	N	^	n		Ä	×	«					Ł		0E
0F		/	?	O	_	o		Å		»					Ł		0F
	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	

Der Code "CP850" enthält

- alle Zeichen des TUSTEP-Zeichensatzes, die auch in der Code Page 850 (siehe Windows Handbuch) definiert sind
- Akzentbuchstaben, die in der Code Page 850 definiert sind
- alle Zeichen, die mit den Steuerzeichen "# " und "# (name) " codiert werden und auch in der Code Page 850 definiert sind.

## 7. Code CP852: PC Slavic

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
00		SP	0	@	P		p	Ç	É	á			Ł	đ	Ó		00
01		!	1	A	Q	a	q	ü	Í	í			ł	Đ	ß		01
02		"	2	B	R	b	r	é	Í	ó			ł	Ď	Ô		02
03		#	3	C	S	c	s	â	ô	ú			ł	Ě	Ń		03
04		\$	4	D	T	d	t	ä	ö	Ą	ł	ł	ł	đ	ń		04
05		%	5	E	U	e	u	ű	Ł	ą	ł	ł	ł	Ń	ñ	Š	05
06		&	6	F	V	f	v	ć	ł	Ż	ł	ł	ł	Í	Š	÷	06
07		'	7	G	W	g	w	ç	Ś	ż	ł	ł	ł	Î	š		07
08		(	8	H	X	h	x	ł	ś	Ę	ł	ł	ł	Ě	Ř	°	08
09		)	9	I	Y	i	y	è	Ö	ę			ł	Ú		09	
0A		*	:	J	Z	j	z	õ	Ü	ł	ł		ł	ř		0A	
0B		+	;	K	[	k	{	õ	Ť	ž			ł	Ů	ů	0B	
0C		,	<	L	\	l		î	ť	Č				ý	ř	0C	
0D		-	=	M	]	m	}	ž	Ł	š	ł	ł	ł	Ý	ř	0D	
0E		.	>	N	^	n		Ä	×	«	ł		ł	Ů	ť	0E	
0F		/	?	O	_	o		Ć	č	»	ł	ł	ł			0F	
	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	

Der Code "CP852" enthält

- alle Zeichen des TUSTEP-Zeichensatzes, die auch in der Code Page 852 (siehe Windows Handbuch) definiert sind
- Akzentbuchstaben, die in der Code Page 852 definiert sind
- alle Zeichen, die mit den Steuerzeichen "# " und "# (name) " codiert werden und auch in der Code Page 852 definiert sind.

## 8. Code CP1250: Windows Central European

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
00		SP	0	@	P		p	€				°	Ř	Đ	ř	đ	00
01		!	1	A	Q	a	q					±	Ǻ	Ǻ	á	ń	01
02		"	2	B	R	b	r						Ǻ	Ǻ	â	ň	02
03		#	3	C	S	c	s		"	Ł	ł		Ǻ	Ó	ă	ó	03
04		\$	4	D	T	d	t	"	"	¤			Ǻ	Ô	ä	ô	04
05		%	5	E	U	e	u			Ą			Ł	Ó	í	õ	05
06		&	6	F	V	f	v	†		ı			Č	Ö	č	ö	06
07		'	7	G	W	g	w			Ş			Ç	x	ç	÷	07
08		(	8	H	X	h	x						Č	Ř	č	ř	08
09		)	9	I	Y	i	y	‰		©	ą		É	Ů	é	ů	09
0A		*	:	J	Z	j	z	Š	š	Ş	ş		Ě	Ú	ě	ú	0A
0B		+	;	K	[	k	{	<	>	«	»		Ě	Ů	è	ú	0B
0C		,	<	L	\	l		Š	ś	ı	Ł		Ě	Ů	ë	ü	0C
0D		-	=	M	]	m	}	Ť	ť				Í	Ý	í	ý	0D
0E		.	>	N	^	n		Ž	ž	®	ł	î	Ť	î	ț		0E
0F		/	?	O	_	o		Ž	ž	Ž	ž	Đ	ß	đ			0F
	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	

Der Code "CP1250" enthält

- alle Zeichen des TUSTEP-Zeichensatzes, die auch in der Code Page 1250 (siehe Windows Handbuch) definiert sind
- Akzentbuchstaben, die in der Code Page 1250 definiert sind
- alle Zeichen, die mit den Steuerzeichen "# " und "# (name) " codiert werden und auch in der Code Page 1250 definiert sind.

## 9. Code CP1252: Windows Western, ANSI

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
00		SP	0	@	P		p	€		°	À	Ð	à	ð			00
01		!	1	A	Q	a	q			±	Á	Ñ	á	ñ			01
02		"	2	B	R	b	r			¢	Â	Ò	â	ò			02
03		#	3	C	S	c	s		"	£	Ã	Ó	ã	ó			03
04		\$	4	D	T	d	t	"	"	¤	Ä	Ô	ä	ô			04
05		%	5	E	U	e	u				Å	Õ	å	õ			05
06		&	6	F	V	f	v	†		¡	Æ	Ö	æ	ö			06
07		'	7	G	W	g	w			§	Ç	×	ç	÷			07
08		(	8	H	X	h	x				È	Ø	è	ø			08
09		)	9	I	Y	i	y	‰		©	É	Ù	é	ù			09
0A		*	:	J	Z	j	z	Š	š		Ê	Ú	ê	ú			0A
0B		+	;	K	[	k	{	<	>	«	»	Ë	Û	ë	û		0B
0C		,	<	L	\	l		Œ	œ	¬		Ì	Ü	ì	ü		0C
0D		-	=	M	]	m	}					Í	Ý	í	ý		0D
0E		.	>	N	^	n		Ž	ž	®		Î	Þ	î	þ		0E
0F		/	?	O	_	o		ÿ	ÿ	¸		Ï	ß	ï	ÿ		0F
	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	

Der Code "CP1252" enthält

- alle Zeichen des TUSTEP-Zeichensatzes, die auch in der Code Page 1252 (siehe Windows Handbuch) definiert sind
- Akzentbuchstaben, die in der Code Page 1252 definiert sind
- alle Zeichen, die mit den Steuerzeichen "# " und "# (name) " codiert werden und auch in der Code Page 1252 definiert sind.

## 10. Code CP10000: Macintosh Roman

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
00		SP	0	@	P		p	Ä	ê	†	∞	¿					00
01		!	1	A	Q	a	q	Å	ë	°	±	i		·	Ò		01
02		"	2	B	R	b	r	Ç	í	ç	≤	¬	"		Ú		02
03		#	3	C	S	c	s	É	ì	£	≥	√	"	"	Û		03
04		\$	4	D	T	d	t	Ñ	î	§				‰	Ü		04
05		%	5	E	U	e	u	Ö	ï				≈	ˆ	ı		05
06		&	6	F	V	f	v	Ü	ñ		∂			÷	Ê		06
07		'	7	G	W	g	w	á	ó	ß			«		Ë		07
08		(	8	H	X	h	x	à	ò	®			»	ÿ	Ë		08
09		)	9	I	Y	i	y	â	ô	©				ÿ	È		09
0A		*	:	J	Z	j	z	ä	ö		∫				Í		0A
0B		+	;	K	[	k	{	ā	ō				À	¤	Î		0B
0C		,	<	L	\	l		å	ú				Ã	<	Ï		0C
0D		-	=	M	]	m	}	ç	ù	≠			Õ	>	Ì		0D
0E		.	>	N	^	n		é	û	Æ	æ	Œ			Ó		0E
0F		/	?	O	_	o		è	ü	Ø	ø	œ			Ô		0F
	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	

Der Code "CP10000" enthält

- alle Zeichen des TUSTEP-Zeichensatzes, die auch in der Code Page 10000 (siehe Macintosh-Handbuch) definiert sind
- Akzentbuchstaben, die in der Code Page 10000 definiert sind
- alle Zeichen, die mit den Steuerzeichen "# " und "# (name) " codiert werden und auch in der Code Page 10000 definiert sind.

## 11. Code CP10029: Macintosh Central European

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
00		SP	0	@	P		p	Ä	ž	†	ı	ŋ		ř	ū		00
01		!	1	A	Q	a	q	Ā	Ď	°	ī	Ń		Š	Ů		01
02		"	2	B	R	b	r	ā	í	Ę	≤	¬		"		Ú	02
03		#	3	C	S	c	s	É	ď	£	≥	√		"	"	ú	03
04		\$	4	D	T	d	t	Ā	Ě	§	ī	ń		š	ů		04
05		%	5	E	U	e	u	Ö	ē		ķ	Ń		š	ú		05
06		&	6	F	V	f	v	Ü	É		ð			÷	ś	Ů	06
07		'	7	G	W	g	w	á	ó	ß			«		Ǻ	ų	07
08		(	8	H	X	h	x	ą	é	®	ł		»	ō	Ť	Ÿ	08
09		)	9	I	Y	i	y	Č	ô	©	ł			Ř	ř	ý	09
0A		*	:	J	Z	j	z	ä	ö		ł			ř	í	ķ	0A
0B		+	;	K	[	k	{	č	ō	ę	ł	ł	ń	Ř	ž	ž	0B
0C		,	<	L	\	l		Ć	ú		ł	ł	Ń	<	ž	Ł	0C
0D		-	=	M	]	m	}	ć	ě	≠	ł	ł	Ń	>	Ů	ž	0D
0E		.	>	N	^	n		é	ě	g	ł	ł	ř	ř	ó	Ģ	0E
0F		/	?	O	_	o		Ž	ü	ı	Ń	Ń	Ń	Ŕ	ô		0F
	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	

Der Code "CP10029" enthält

- alle Zeichen des TUSTEP-Zeichensatzes, die auch in der Code Page 10029 (siehe Macintosh-Handbuch) definiert sind
- Akzentbuchstaben, die in der Code Page 10029 definiert sind
- alle Zeichen, die mit den Steuerzeichen "# " und "# (name) " codiert werden und auch in der Code Page 10029 definiert sind.



## 12. Code DECMCS: VT100-Terminals

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
00			SP	0	@	P		p					°	À		à	00
01		!	1	A	Q	a	q				i	±	Á	Ñ	á	ñ	01
02		"	2	B	R	b	r				ç		Â	Ò	â	ò	02
03		#	3	C	S	c	s				£		Ã	Ó	ã	ó	03
04		\$	4	D	T	d	t						Ä	Ô	ä	ô	04
05		%	5	E	U	e	u						Å	Õ	å	õ	05
06		&	6	F	V	f	v						Æ	Ö	æ	ö	06
07		'	7	G	W	g	w				§	·	Ç	Ç	ç	œ	07
08		(	8	H	X	h	x				¤		È	Ø	è	ø	08
09		)	9	I	Y	i	y				©		É	Ù	é	ù	09
0A		*	:	J	Z	j	z						Ê	Ú	ê	ú	0A
0B		+	;	K	[	k	{				«	»	Ë	Û	ë	û	0B
0C		,	<	L	\	l							Ì	Ü	ì	ü	0C
0D		-	=	M	]	m	}						Í	Ý	í	ý	0D
0E		.	>	N	^	n							Î		î		0E
0F		/	?	O	_	o						¿	Ï	ß	ï		0F
	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	

Der Code "DECMCS" enthält

- alle Zeichen des TUSTEP-Zeichensatzes, die auch im "DEC Multinational Character Set" (siehe VT100-Handbuch) definiert sind
- Akzentbuchstaben, die im "DEC Multinational Character Set" definiert sind
- alle Zeichen, die mit den Steuerzeichen "#" und "# (name)" codiert werden und auch im "DEC Multinational Character Set" definiert sind.

## 13. Code ISO8859: ISO-8859-1, Latin-1

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
00			SP	0	@	P		p				°	À	Ð	à	ð	00
01		!	1	A	Q	a	q				ı	±	Á	Ñ	á	ñ	01
02		"	2	B	R	b	r				ç		Â	Ò	â	ò	02
03		#	3	C	S	c	s				£		Ã	Ó	ã	ó	03
04		\$	4	D	T	d	t				¤		Ä	Ô	ä	ô	04
05		%	5	E	U	e	u						Å	Õ	å	õ	05
06		&	6	F	V	f	v				ı		Æ	Ö	æ	ö	06
07		'	7	G	W	g	w				§	·	Ç	×	ç	÷	07
08		(	8	H	X	h	x						È	Ø	è	ø	08
09		)	9	I	Y	i	y				©		É	Ù	é	ù	09
0A		*	:	J	Z	j	z						Ê	Ú	ê	ú	0A
0B		+	;	K	[	k	{				«	»	Ë	Û	ë	û	0B
0C		,	<	L	\	l					-		Ì	Ü	ì	ü	0C
0D		-	=	M	]	m	}						Í	Ý	í	ý	0D
0E		.	>	N	^	n					®		Î	Þ	î	þ	0E
0F		/	?	O	_	o						¿	Ï	ß	ï	ÿ	0F
	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	

Der Code "ISO8859" enthält

- alle Zeichen des TUSTEP-Zeichensatzes, die auch in ISO-8859-1 definiert sind
- Akzentbuchstaben, die im Code ISO-8859-1 definiert sind
- alle Zeichen, die mit den Steuerzeichen "# " und "# (name) " codiert werden und auch in ISO-8859-1 definiert sind.

## 14. Code LINUX: Linux-Konsole

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0		
00			SP	0	@	P		p					°		à		00	
01		!	1	A	Q	a	q					ı	±	Ñ	á	ñ	01	
02		"	2	B	R	b	r					ç			â	ò	02	
03		#	3	C	S	c	s					£				ó	03	
04		\$	4	D	T	d	t					¤		Ä	ä	ô	04	
05		%	5	E	U	e	u							Å	å		05	
06		&	6	F	V	f	v							Æ	Ö	æ	ö	06
07		'	7	G	W	g	w					§	·	Ç		ç	÷	07
08		(	8	H	X	h	x									è		08
09		)	9	I	Y	i	y							É	é	ù	09	
0A		*	:	J	Z	j	z								ê	ú	0A	
0B		+	;	K	[	k	{					«	»			ë	û	0B
0C		,	<	L	\	l						-			Ü	ì	ü	0C
0D		-	=	M	]	m	}									í	ý	0D
0E		.	>	N	^	n										î	þ	0E
0F		/	?	O	_	o							ı		ß	ï	ÿ	0F
	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0		

Der Code "LINUX" enthält alle Zeichen des Codes "ISO8859", die auch im Code "CP437" definiert sind.

## 15. Code Latin1: ISO-8859-1

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
00			SP	0	@	P		p				°	À	Ð	à	ð	00
01		!	1	A	Q	a	q				ı	±	Á	Ñ	á	ñ	01
02		"	2	B	R	b	r				ç		Â	Ò	â	ò	02
03		#	3	C	S	c	s				£		Ã	Ó	ã	ó	03
04		\$	4	D	T	d	t				¤		Ä	Ô	ä	ô	04
05		%	5	E	U	e	u						Å	Õ	å	õ	05
06		&	6	F	V	f	v				ı		Æ	Ö	æ	ö	06
07		'	7	G	W	g	w				§	·	Ç	×	ç	÷	07
08		(	8	H	X	h	x						È	Ø	è	ø	08
09		)	9	I	Y	i	y				©		É	Ù	é	ù	09
0A		*	:	J	Z	j	z						Ê	Ú	ê	ú	0A
0B		+	;	K	[	k	{				«	»	Ë	Û	ë	û	0B
0C		,	<	L	\	l					-		Ì	Ü	ì	ü	0C
0D		-	=	M	]	m	}						Í	Ý	í	ý	0D
0E		.	>	N	^	n					®		Î	Þ	î	þ	0E
0F		/	?	O	_	o						¿	Ï	ß	ï	ÿ	0F
	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	

Der Code "Latin1" enthält

- alle Zeichen des TUSTEP-Zeichensatzes, die auch in ISO-8859-1 definiert sind
- Akzentbuchstaben, die in ISO-8859-1 definiert sind
- alle Zeichen, die mit den Steuerzeichen "# " und "# (name) " codiert werden und auch in ISO-8859-1 definiert sind.

## 16. Code Latin2: ISO-8859-2

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0		
00			SP	0	@	P		p					°	Ř	Đ	ř	đ	00
01		!	1	A	Q	a	q				Ą	ą	Ǻ	Ǻ	á	ń		01
02		"	2	B	R	b	r						Ǻ	Ǻ	â	ň		02
03		#	3	C	S	c	s				Ł	ł	Ǻ	Ó	ă	ó		03
04		\$	4	D	T	d	t				ł		Ǻ	Ô	ă	ô		04
05		%	5	E	U	e	u				Ĺ	ĺ	Ĺ	Ó	í	ó		05
06		&	6	F	V	f	v				Ś	ś	Ć	Ö	ć	ö		06
07		'	7	G	W	g	w				Ş		Ç	x	ç	÷		07
08		(	8	H	X	h	x						Č	Ř	č	ř		08
09		)	9	I	Y	i	y				Š	š	É	Ů	é	ů		09
0A		*	:	J	Z	j	z				Ş	ş	Ě	Ú	ę	ú		0A
0B		+	;	K	[	k	{				Ť	ť	Ě	Ů	è	ú		0B
0C		,	<	L	\	l					Ž	ž	Ě	Ů	ë	ü		0C
0D		-	=	M	]	m	}						Í	Ý	í	ý		0D
0E		.	>	N	^	n					Ž	ž	Î	Ŧ	î	ț		0E
0F		/	?	O	_	o					Ž	ž	Ď	ß	ď			0F
	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0		

Der Code "Latin2" enthält

- alle Zeichen des TUSTEP-Zeichensatzes, die auch in ISO-8859-2 definiert sind
- Akzentbuchstaben, die in ISO-8859-2 definiert sind
- alle Zeichen, die mit den Steuerzeichen "# " und "# (name) " codiert werden und auch in ISO-8859-2 definiert sind.

## 17. Code Latin3: ISO-8859-3

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
00		SP	0	@	P			p					À		à		00
01		!	1	A	Q	a	q			Ɔ	ħ	Á	Ñ	á	ñ		01
02		"	2	B	R	b	r					Â	Ò	â	ò		02
03		#	3	C	S	c	s			£			Ó		ó		03
04		\$	4	D	T	d	t		¤			Ä	Ô	ä	ô		04
05		%	5	E	U	e	u						Ç	Ĝ	ç	ĝ	05
06		&	6	F	V	f	v			Ĥ	ĥ	Ĉ	Ö	ĉ	ö		06
07		'	7	G	W	g	w			Š	·	Ç	×	ç	÷		07
08		(	8	H	X	h	x						È	Ĝ	è	ĝ	08
09		)	9	I	Y	i	y			İ	ı	É	Ù	é	ù		09
0A		*	:	J	Z	j	z			Š	š	Ê	Ú	ê	ú		0A
0B		+	;	K	[	k	{			Ǧ	ǧ	Ë	Û	ë	û		0B
0C		,	<	L	\	l				Ĵ	ĵ	Ì	Ü	ì	ü		0C
0D		-	=	M	]	m	}						Í	Û	í	Û	0D
0E		.	>	N	^	n							Î	Ŝ	î	ŝ	0E
0F		/	?	O	_	o				Ž	ž	Ï	ß	ï			0F
	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	

Der Code "Latin3" enthält

- alle Zeichen des TUSTEP-Zeichensatzes, die auch in ISO-8859-3 definiert sind
- Akzentbuchstaben, die in ISO-8859-3 definiert sind
- alle Zeichen, die mit den Steuerzeichen "# " und "# (name) " codiert werden und auch in ISO-8859-3 definiert sind.

## 18. Code Latin4: ISO-8859-4

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0		
00			SP	0	@	P		p					°	Ā	Đ	ā	đ	00
01		!	1	A	Q	a	q				Ą	ą	Ā	Ń	á	ą	01	
02		"	2	B	R	b	r						Ā	Ō	â	ô	02	
03		#	3	C	S	c	s				Ŕ	ŕ	Ā	Ŗ	ā	ŗ	03	
04		\$	4	D	T	d	t				ŗ		Ā	Ô	ä	ô	04	
05		%	5	E	U	e	u				Ī	ī	Ā	Ō	ā	ō	05	
06		&	6	F	V	f	v				Ł	ł	Æ	Ö	æ	ö	06	
07		'	7	G	W	g	w				Ś		Į	×	į	÷	07	
08		(	8	H	X	h	x						Č	Ø	č	ø	08	
09		)	9	I	Y	i	y				Š	š	É	Ū	é	ų	09	
0A		*	:	J	Z	j	z				Ē	ē	Ê	Ú	ę	ú	0A	
0B		+	;	K	[	k	{				Ģ	ģ	Ë	Û	ë	û	0B	
0C		,	<	L	\	l					Ŧ	ŧ	Ė	Ü	ė	ü	0C	
0D		-	=	M	]	m	}						Í	Û	í	ű	0D	
0E		.	>	N	^	n					Ž	ž	Î	Ū	î	ű	0E	
0F		/	?	O	_	o							Ī	ß	ī		0F	
	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0		

Der Code "Latin4" enthält

- alle Zeichen des TUSTEP-Zeichensatzes, die auch in ISO-8859-4 definiert sind
- Akzentbuchstaben, die in ISO-8859-4 definiert sind
- alle Zeichen, die mit den Steuerzeichen "# " und "# (name) " codiert werden und auch in ISO-8859-4 definiert sind.

## 19. Code Latin5: ISO-8859-9

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0		
00			SP	0	@	P		p					°	À	Ğ	à	ğ	00
01		!	1	A	Q	a	q				ı	±	Á	Ñ	á	ñ	01	
02		"	2	B	R	b	r				ç		Â	Ò	â	ò	02	
03		#	3	C	S	c	s				£		Ã	Ó	ã	ó	03	
04		\$	4	D	T	d	t				¤		Ä	Ô	ä	ô	04	
05		%	5	E	U	e	u						Å	Õ	å	õ	05	
06		&	6	F	V	f	v				ı		Æ	Ö	æ	ö	06	
07		'	7	G	W	g	w				§	·	Ç	×	ç	÷	07	
08		(	8	H	X	h	x						È	Ø	è	ø	08	
09		)	9	I	Y	i	y				©		É	Ù	é	ù	09	
0A		*	:	J	Z	j	z						Ê	Ú	ê	ú	0A	
0B		+	;	K	[	k	{				«	»	Ë	Û	ë	û	0B	
0C		,	<	L	\	l					-		Ì	Ü	ì	ü	0C	
0D		-	=	M	]	m	}						Í	İ	í	ı	0D	
0E		.	>	N	^	n					®		Î	Ş	î	ş	0E	
0F		/	?	O	_	o						¿	Ï	ß	ï	ÿ	0F	
	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0		

Der Code "Latin5" enthält

- alle Zeichen des TUSTEP-Zeichensatzes, die auch in ISO-8859-9 definiert sind
- Akzentbuchstaben, die in ISO-8859-9 definiert sind
- alle Zeichen, die mit den Steuerzeichen "# " und "# (name) " codiert werden und auch in ISO-8859-9 definiert sind.



## 20. Code Latin6: ISO-8859-10

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0		
00			SP	0	@	P		p					°	À	Ð	ā	ð	00
01		!	1	A	Q	a	q				Ą	ą	Á	Ń	á	ñ	01	
02		"	2	B	R	b	r				Ě	ě	Â	Ō	â	ō	02	
03		#	3	C	S	c	s				Ġ	ġ	Ã	Ó	ã	ó	03	
04		\$	4	D	T	d	t				Ī	ī	Ä	Ô	ä	ô	04	
05		%	5	E	U	e	u				Ĭ	ĭ	Å	Õ	å	õ	05	
06		&	6	F	V	f	v				Ų	ų	Æ	Ö	æ	ö	06	
07		'	7	G	W	g	w				Š	š	Į	Ū	į	ū	07	
08		(	8	H	X	h	x				Ł	ł	Č	Ø	č	ø	08	
09		)	9	I	Y	i	y				Đ	đ	É	Ů	é	ů	09	
0A		*	:	J	Z	j	z				Š	š	Ě	Ú	ę	ú	0A	
0B		+	;	K	[	k	{				Ŧ	ŧ	Ë	Û	ë	û	0B	
0C		,	<	L	\	l					Ž	ž	Ě	Ü	ě	ü	0C	
0D		-	=	M	]	m	}						Í	Ý	í	ý	0D	
0E		.	>	N	^	n					Ū	ū	Î	Ɔ	î	Ɔ	0E	
0F		/	?	O	_	o							İ	ß	ï		0F	
	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0		

Der Code "Latin6" enthält

- alle Zeichen des TUSTEP-Zeichensatzes, die auch in ISO-8859-10 definiert sind
- Akzentbuchstaben, die in ISO-8859-10 definiert sind
- alle Zeichen, die mit den Steuerzeichen "# " und "# (name) " codiert werden und auch in ISO-8859-10 definiert sind.

## 21. Code Latin7: ISO-8859-13

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
00			SP	0	@	P		p				°	À	Š	ą	š	00
01		!	1	A	Q	a	q				“	±	Ī	Ń	ı	ń	01
02		”	2	B	R	b	r			ç			Ā	Ń	ā	ņ	02
03		#	3	C	S	c	s		£				Č	Ó	č	ó	03
04		\$	4	D	T	d	t		¤	”			Ä	Ö	ä	ö	04
05		%	5	E	U	e	u			”			Å	Ö	å	ö	05
06		&	6	F	V	f	v		!				ƒ	Ö	ę	ö	06
07		'	7	G	W	g	w		§	.			Ē	×	ē	÷	07
08		(	8	H	X	h	x		∅	∅			Č	Ů	č	ů	08
09		)	9	I	Y	i	y		©				É	Ł	é	ł	09
0A		*	:	J	Z	j	z		Ŕ	ŗ			Ž	Ś	ż	ś	0A
0B		+	;	K	[	k	{		«	»			Ě	Ū	ě	ū	0B
0C		,	<	L	\	l			-				Ĝ	Ü	g	ü	0C
0D		-	=	M	]	m	}						Ķ	Ž	ķ	ž	0D
0E		.	>	N	^	n			®				Ī	Ž	ī	ž	0E
0F		/	?	O	_	o			Æ	æ			Ļ	ß	ļ		0F
	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	

Der Code "Latin7" enthält

- alle Zeichen des TUSTEP-Zeichensatzes, die auch in ISO-8859-13 definiert sind
- Akzentbuchstaben, die in ISO-8859-13 definiert sind
- alle Zeichen, die mit den Steuerzeichen "# " und "# (name) " codiert werden und auch in ISO-8859-13 definiert sind.

## 22. Code Latin8: ISO-8859-14

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
00			SP	0	@	P		p				ƒ	À	Ŵ	à	ŵ	00
01		!	1	A	Q	a	q				Ĳ	í	Á	Ñ	á	ñ	01
02		"	2	B	R	b	r				ĳ	Ġ	Â	Ò	â	ò	02
03		#	3	C	S	c	s				£	ġ	Ã	Ó	ã	ó	03
04		\$	4	D	T	d	t				Č	Ĺ	Ä	Ô	ä	ô	04
05		%	5	E	U	e	u				č	ĺ	Å	Õ	å	õ	05
06		&	6	F	V	f	v				Đ		Æ	Ö	æ	ö	06
07		'	7	G	W	g	w				Š	Ɔ	Ç	Ț	ç	ț	07
08		(	8	H	X	h	x				Ŵ	ŵ	È	Ø	è	ø	08
09		)	9	I	Y	i	y				©	Ɔ	É	Ù	é	ù	09
0A		*	:	J	Z	j	z				Ŵ	ŵ	Ê	Ú	ê	ú	0A
0B		+	;	K	[	k	{				đ	š	Ë	Û	ë	û	0B
0C		,	<	L	\	l					ÿ	ÿ	Ì	Ü	ì	ü	0C
0D		-	=	M	]	m	}					Ŵ	Í	Ý	í	ý	0D
0E		.	>	N	^	n					®	ŵ	Î	Ŷ	î	ÿ	0E
0F		/	?	O	_	o					ÿ	š	Ï	ß	ï	ÿ	0F
	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	

Der Code "Latin8" enthält

- alle Zeichen des TUSTEP-Zeichensatzes, die auch in ISO-8859-14 definiert sind
- Akzentbuchstaben, die in ISO-8859-14 definiert sind
- alle Zeichen, die mit den Steuerzeichen "# " und "# (name) " codiert werden und auch in ISO-8859-14 definiert sind.

## 23. Code Latin9: ISO-8859-15

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
00			SP	0	@	P		p				°	À	Ð	à	ð	00
01		!	1	A	Q	a	q				ı	±	Á	Ñ	á	ñ	01
02		"	2	B	R	b	r				ç		Â	Ò	â	ò	02
03		#	3	C	S	c	s				£		Ã	Ó	ã	ó	03
04		\$	4	D	T	d	t				€	Ž	Ä	Ô	ä	ô	04
05		%	5	E	U	e	u						Å	Õ	å	õ	05
06		&	6	F	V	f	v				Š		Æ	Ö	æ	ö	06
07		'	7	G	W	g	w				Š	·	Ç	×	ç	÷	07
08		(	8	H	X	h	x				š	ž	È	Ø	è	ø	08
09		)	9	I	Y	i	y				©		É	Ù	é	ù	09
0A		*	:	J	Z	j	z						Ê	Ú	ê	ú	0A
0B		+	;	K	[	k	{				«	»	Ë	Û	ë	û	0B
0C		,	<	L	\	l					¬	ƒ	Ì	Ü	ì	ü	0C
0D		-	=	M	]	m	}					œ	Í	Ý	í	ý	0D
0E		.	>	N	^	n					®	ÿ	Î	Þ	î	þ	0E
0F		/	?	O	_	o						ı	Ï	ß	ï	ÿ	0F
	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	

Der Code "Latin9" enthält

- alle Zeichen des TUSTEP-Zeichensatzes, die auch in ISO-8859-15 definiert sind
- Akzentbuchstaben, die in ISO-8859-15 definiert sind
- alle Zeichen, die mit den Steuerzeichen "# " und "# (name) " codiert werden und auch in ISO-8859-15 definiert sind.

## 24. Code Latin10: ISO-8859-16

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
00			SP	0	@	P		p				°	À	Đ	à	đ	00
01		!	1	A	Q	a	q		Ą	±	Á	Ń	á	ń			01
02		"	2	B	R	b	r		ą	Č	Ā	Ò	â	ò			02
03		#	3	C	S	c	s		Ł	ł	Ǻ	Ó	ǻ	ó			03
04		\$	4	D	T	d	t		€	Ž	Ǻ	Ô	ǻ	ô			04
05		%	5	E	U	e	u		„	”	Č	Õ	ć	õ			05
06		&	6	F	V	f	v		Š		Æ	Ö	æ	ö			06
07		'	7	G	W	g	w		Š	·	Ç	Š	ç	ś			07
08		(	8	H	X	h	x		š	ž	È	Û	è	ú			08
09		)	9	I	Y	i	y		©	č	É	Ù	é	ù			09
0A		*	:	J	Z	j	z		Ş	ş	Ê	Ú	ê	ú			0A
0B		+	;	K	[	k	{		«	»	Ë	Û	ë	û			0B
0C		,	<	L	\	l			Ž	Ɖ	Ì	Ü	ì	ü			0C
0D		-	=	M	]	m	}			œ	Í	Ɖ	í	ę			0D
0E		.	>	N	^	n			ž	ÿ	Î	Ŧ	î	ț			0E
0F		/	?	O	_	o			Ž	ž	Ï	ß	ï	ÿ			0F
	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	

Der Code "Latin10" enthält

- alle Zeichen des TUSTEP-Zeichensatzes, die auch in ISO-8859-16 definiert sind
- Akzentbuchstaben, die in ISO-8859-16 definiert sind
- alle Zeichen, die mit den Steuerzeichen "# " und "# (name) " codiert werden und auch in ISO-8859-16 definiert sind.

## Tabellen für den EBCDIC-Code

## 1. Interner TUSTEP-Code

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
00					SP	&	-	^:	^C	^J	^Q	^X	{	}	\	0	00
01						^)	/	^;	a	j		^Y	A	J		1	01
02					^!	^*	^2	^<	b	k	s	^Z	B	K	S	2	02
03					^"	^+	^3	^=	c	l	t	^[	C	L	T	3	03
04					^#	^,	^4	^>	d	m	u	^\	D	M	U	4	04
05					^\$	^-	^5	^?	e	n	v	^]	E	N	V	5	05
06					^%	^.	^6	^@	f	o	w		F	O	W	6	06
07					^&	^/	^7	^A	g	p	x	^_	G	P	X	7	07
08					^'	^0	^8	^B	h	q	y		H	Q	Y	8	08
09					^(	^1	^9		i	r	z	^a	I	R	Z	9	09
0A					[	]		:	^D	^K	^R	^b	^h	^n	^t	^z	0A
0B					.	\$	,	#	^E	^L	^S	^c	^i	^o	^u	^{	0B
0C					<	*	%	@	^F	^M	^T	^d	^j	^p	^v	^	0C
0D					(	)	_	'	^G	^N	^U	^e	^k	^q	^w	^}	0D
0E					+	;	>	=	^H	^O	^V	^f	^l	^r	^x		0E
0F					!	^	?	"	^I	^P	^W	^g	^m	^s	^y		0F
	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	

## 2. Internationaler EBCDIC-Code zum Vergleich

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
00					SP	&	-						{	}	\	0	00
01								/	a	j	~		A	J		1	01
02									b	k	s		B	K	S	2	02
03									c	l	t		C	L	T	3	03
04									d	m	u		D	M	U	4	04
05									e	n	v		E	N	V	5	05
06									f	o	w		F	O	W	6	06
07									g	p	x		G	P	X	7	07
08									h	q	y		H	Q	Y	8	08
09								`	i	r	z		I	R	Z	9	09
0A					[	]		:									0A
0B					.	\$	,	#									0B
0C					<	*	%	@									0C
0D					(	)	_	'									0D
0E					+	;	>	=									0E
0F					!	^	?	"									0F
	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	

79=Gravis A1=Tilde

3. Deutsche Variante des EBCDIC-Codes

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
00					SP	&	-						ä	ü	Ö	0	00
01							/	a	j	ß			A	J		1	01
02								b	k	s			B	K	S	2	02
03								c	l	t			C	L	T	3	03
04								d	m	u			D	M	U	4	04
05								e	n	v			E	N	V	5	05
06								f	o	w			F	O	W	6	06
07								g	p	x			G	P	X	7	07
08								h	q	y			H	Q	Y	8	08
09							`	i	r	z			I	R	Z	9	09
0A					Ä	Ü	ö	:									0A
0B					.	\$	,	#									0B
0C					<	*	%	@									0C
0D					(	)	-	'									0D
0E					+	;	>	=									0E
0F					!	^	?	"									0F
	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	

79=Gravis

4. Code EBCDIC: US-EBCDIC

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
00					SP	&	-						{	}	\	0	00
01							/	a	j	~			A	J		1	01
02								b	k	s			B	K	S	2	02
03								c	l	t			C	L	T	3	03
04								d	m	u			D	M	U	4	04
05								e	n	v			E	N	V	5	05
06								f	o	w			F	O	W	6	06
07								g	p	x			G	P	X	7	07
08								h	q	y			H	Q	Y	8	08
09							`	i	r	z			I	R	Z	9	09
0A					ç	!		:									0A
0B					.	\$	,	#									0B
0C					<	*	%	@									0C
0D					(	)	-	'									0D
0E					+	;	>	=									0E
0F						~	?	"									0F
	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	

79=Gravis 4A=Cent 5F=log.Nicht 6A=unterbr.Strich A1=Tilde

**TUSTEP-Umcodierung von ASCII nach EBCDIC**

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
00	00	10	40	F0	7C	D7	79	97	20	30	41	58	76	9F	B8	DC	00
01	01	11	4F	F1	C1	D8	81	98	21	31	42	59	77	A0	B9	DD	01
02	02	12	7F	F2	C2	D9	82	99	22	1A	43	62	78	AA	BA	DE	02
03	03	13	7B	F3	C3	E2	83	A2	23	33	44	63	80	AB	BB	DF	03
04	37	3C	5B	F4	C4	E3	84	A3	24	34	45	64	8A	AC	BC	EA	04
05	2D	3D	6C	F5	C5	E4	85	A4	15	35	46	65	8B	AD	BD	EB	05
06	2E	32	50	F6	C6	E5	86	A5	06	36	47	66	8C	AE	BE	EC	06
07	2F	26	7D	F7	C7	E6	87	A6	17	08	48	67	8D	AF	BF	ED	07
08	16	18	4D	F8	C8	E7	88	A7	28	38	49	68	8E	B0	CA	EE	08
09	05	19	5D	F9	C9	E8	89	A8	29	39	51	69	8F	B1	CB	EF	09
0A	25	3F	5C	7A	D1	E9	91	A9	2A	3A	52	70	90	B2	CC	FA	0A
0B	0B	27	4E	5E	D2	4A	92	C0	2B	3B	53	71	9A	B3	CD	FB	0B
0C	0C	1C	6B	4C	D3	E0	93	6A	2C	04	54	72	9B	B4	CE	FC	0C
0D	0D	1D	60	7E	D4	5A	94	D0	09	14	55	73	9C	B5	CF	FD	0D
0E	0E	1E	4B	6E	D5	5F	95	A1	0A	3E	56	74	9D	B6	DA	FE	0E
0F	0F	1F	61	6F	D6	6D	96	07	1B	E1	57	75	9E	B7	DB	FF	0F
	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	

**TUSTEP-Umcodierung von EBCDIC nach ASCII**

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
00	00	10	80	90	20	26	2D	BA	C3	CA	D1	D8	7B	7D	5C	30	00
01	01	11	81	91	A0	A9	2F	BB	61	6A	7E	D9	41	4A	9F	31	01
02	02	12	82	16	A1	AA	B2	BC	62	6B	73	DA	42	4B	53	32	02
03	03	13	83	93	A2	AB	B3	BD	63	6C	74	DB	43	4C	54	33	03
04	9C	9D	84	94	A3	AC	B4	BE	64	6D	75	DC	44	4D	55	34	04
05	09	85	0A	95	A4	AD	B5	BF	65	6E	76	DD	45	4E	56	35	05
06	86	08	17	96	A5	AE	B6	C0	66	6F	77	DE	46	4F	57	36	06
07	7F	87	1B	04	A6	AF	B7	C1	67	70	78	DF	47	50	58	37	07
08	97	18	88	98	A7	B0	B8	C2	68	71	79	E0	48	51	59	38	08
09	8D	19	89	99	A8	B1	B9	60	69	72	7A	E1	49	52	5A	39	09
0A	8E	92	8A	9A	5B	5D	7C	3A	C4	CB	D2	E2	E8	EE	F4	FA	0A
0B	0B	8F	8B	9B	2E	24	2C	23	C5	CC	D3	E3	E9	EF	F5	FB	0B
0C	0C	1C	8C	14	3C	2A	25	40	C6	CD	D4	E4	EA	F0	F6	FC	0C
0D	0D	1D	05	15	28	29	5F	27	C7	CE	D5	E5	EB	F1	F7	FD	0D
0E	0E	1E	06	9E	2B	3B	3E	3D	C8	CF	D6	E6	EC	F2	F8	FE	0E
0F	0F	1F	07	1A	21	5E	3F	22	C9	D0	D7	E7	ED	F3	F9	FF	0F
	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	



## Standard-Sortierfolge in TUSTEP

In der folgenden Tabelle sind die einzelnen Zeichen in der Reihenfolge aufgeführt, in der sie standardmäßig (d. h. wenn nichts anderes über Parameter angegeben wird) aufsteigend sortiert werden. Zeichen, die in einer Spalte nebeneinander stehen, werden gleichwertig behandelt.

1: SP	11: * ^*	21: > ^>	31: } ^}
2: ! ^!	12: + ^+	22: ? ^?	32: 0 ^0
3: " ^^	13: , ^,	23: @ ^@	33: 1 ^1
4: # ^#	14: - ^-	24: [ ^[	...
5: \$ ^\$	15: . ^.	25: \ ^\	41: 9 ^9
6: % ^%	16: / ^/	26: ] ^]	42: a ^a A ^A
7: & ^&	17: : ^:	27: ^	43: b ^b B ^B
8: ' ^'	18: ; ^;	28: _ ^_	...
9: ( ^( 10: ) ^)	19: < ^< 20: = ^=	29: { ^{ 30:   ^	66: y ^y Y ^Y 67: z ^z Z ^Z

## Sonder-Sortierfolge in TUSTEP

In der folgenden Tabelle sind die einzelnen Zeichen in der Reihenfolge aufgeführt, in der sie in Sonderfällen (diese sind in der Beschreibung jeweils angegeben) aufsteigend sortiert werden. Zeichen, die in einer Spalte nebeneinander stehen, werden gleichwertig behandelt.

1: SP	11: * ^*	21: > ^>	31: } ^}
2: ! ^!	12: + ^+	22: ? ^?	32: 0 ^0
3: " ^^	13: , ^,	23: @ ^@	...
4: # ^#	14: - ^-	24: [ ^[	41: 9 ^9
5: \$ ^\$	15: . ^.	25: \ ^\	42: a A
6: % ^%	16: / ^/	26: ] ^]	...
7: & ^&	17: : ^:	27: ^	67: z Z
8: ' ^'	18: ; ^;	28: _ ^_	68: ^a ^A
9: ( ^( 10: ) ^)	19: < ^< 20: = ^=	29: { ^{ 30:   ^	...
			93: ^z ^Z

## Umrechnungstabelle Hexadezimal – Dezimal

	00	10	20	30	40	50	60	70	
00	0	16	32	48	64	80	96	112	00
01	1	17	33	49	65	81	97	113	01
02	2	18	34	50	66	82	98	114	02
03	3	19	35	51	67	83	99	115	03
04	4	20	36	52	68	84	100	116	04
05	5	21	37	53	69	85	101	117	05
06	6	22	38	54	70	86	102	118	06
07	7	23	39	55	71	87	103	119	07
08	8	24	40	56	72	88	104	120	08
09	9	25	41	57	73	89	105	121	09
0A	10	26	42	58	74	90	106	122	0A
0B	11	27	43	59	75	91	107	123	0B
0C	12	28	44	60	76	92	108	124	0C
0D	13	29	45	61	77	93	109	125	0D
0E	14	30	46	62	78	94	110	126	0E
0F	15	31	47	63	79	95	111	127	0F
	00	10	20	30	40	50	60	70	

	80	90	A0	B0	C0	D0	E0	F0	
00	128	144	160	176	192	208	224	240	00
01	129	145	161	177	193	209	225	241	01
02	130	146	162	178	194	210	226	242	02
03	131	147	163	179	195	211	227	243	03
04	132	148	164	180	196	212	228	244	04
05	133	149	165	181	197	213	229	245	05
06	134	150	166	182	198	214	230	246	06
07	135	151	167	183	199	215	231	247	07
08	136	152	168	184	200	216	232	248	08
09	137	153	169	185	201	217	233	249	09
0A	138	154	170	186	202	218	234	250	0A
0B	139	155	171	187	203	219	235	251	0B
0C	140	156	172	188	204	220	236	252	0C
0D	141	157	173	189	205	221	237	253	0D
0E	142	158	174	190	206	222	238	254	0E
0F	143	159	175	191	207	223	239	255	0F
	80	90	A0	B0	C0	D0	E0	F0	

## Umrechnungstabelle Hexadezimal – Oktal

	00	10	20	30	40	50	60	70	
00	0	20	40	60	100	120	140	160	00
01	1	21	41	61	101	121	141	161	01
02	2	22	42	62	102	122	142	162	02
03	3	23	43	63	103	123	143	163	03
04	4	24	44	64	104	124	144	164	04
05	5	25	45	65	105	125	145	165	05
06	6	26	46	66	106	126	146	166	06
07	7	27	47	67	107	127	147	167	07
08	10	30	50	70	110	130	150	170	08
09	11	31	51	71	111	131	151	171	09
0A	12	32	52	72	112	132	152	172	0A
0B	13	33	53	73	113	133	153	173	0B
0C	14	34	54	74	114	134	154	174	0C
0D	15	35	55	75	115	135	155	175	0D
0E	16	36	56	76	116	136	156	176	0E
0F	17	37	57	77	117	137	157	177	0F
	00	10	20	30	40	50	60	70	

	80	90	A0	B0	C0	D0	E0	F0	
00	200	220	240	260	300	320	340	360	00
01	201	221	241	261	301	321	341	361	01
02	202	222	242	262	302	322	342	362	02
03	203	223	243	263	303	323	343	363	03
04	204	224	244	264	304	324	344	364	04
05	205	225	245	265	305	325	345	365	05
06	206	226	246	266	306	326	346	366	06
07	207	227	247	267	307	327	347	367	07
08	210	230	250	270	310	330	350	370	08
09	211	231	251	271	311	331	351	371	09
0A	212	232	252	272	312	332	352	372	0A
0B	213	233	253	273	313	333	353	373	0B
0C	214	234	254	274	314	334	354	374	0C
0D	215	235	255	275	315	335	355	375	0D
0E	216	236	256	276	316	336	356	376	0E
0F	217	237	257	277	317	337	357	377	0F
	80	90	A0	B0	C0	D0	E0	F0	



**#DVORBEREITE**

---

Kommando . . . . .	847
Leistung . . . . .	848
Hinweise . . . . .	848
Beispiele . . . . .	848
Modi . . . . .	849
Wahl des Modus . . . . .	849
MODUS = T . . . . .	849
MODUS = -STD- . . . . .	849
MODUS = S . . . . .	850
MODUS = O . . . . .	850
MODUS = P . . . . .	850
MODUS = A . . . . .	850
MODUS = U . . . . .	851
Parameter . . . . .	852
Einstellen der Parameter-Konvention . . . . .	852
Auswahl der Daten . . . . .	852
Angaben zum Protokoll . . . . .	854
Alphabetisches Verzeichnis der Parameter . . . . .	857

## Kommando

#DVORBEREITE

QUELLE	=	datei	Name der Datei mit den Daten, von denen ein Protokoll erstellt werden soll, bzw. (bei MODUS=A und MODUS=U) Name der Datei mit dem Protokoll, aus dem Teile kopiert werden sollen.
	=	-STD-	Die Daten, von denen ein Protokoll erstellt werden soll, bzw. das Protokoll, aus dem Teile kopiert werden sollen, stehen/steht in der Standard-Text-Datei.
MODUS	=	T	Seiten-Zeilen-Nummer vor jedem Satz (für Daten, die im Textmodus nummeriert sind).
	=	-STD-	Seiteneinteilung entsprechend der Seitennummer der Sätze, Zeilennummer vor jedem Satz.
	=	S	Seiteneinteilung entsprechend der Seitennummer der Sätze, keine Seiten-Zeilen-Nummer.
	=	O	Ohne Seiten-Zeilen-Nummer.
	=	P	Zeilennummer vor jedem Satz (für Daten, die im Programmmodus nummeriert sind, auch für Segment-Dateien).
	=	A	QUELL-Datei ist eine Protokoll-Datei; Auswahl über die Seitennummer der Satznummer.
	=	U	QUELL-Datei ist eine Protokoll-Datei; Auswahl über die Seitennummer in der Seitenüberschrift.
	=	...	Druckertyp, für den die Daten aufbereitet werden sollen. Er kann auch über Parameter angegeben werden.  Mit dem Kommando #LISTE, DRUCKERTYPEN können die definierten Druckertypen aufgelistet werden.
LOESCHEN	=	-	* Daten in der PROTOKOLL-Datei nicht löschen.
	=	+	Daten in der PROTOKOLL-Datei zuerst löschen.
PARAMETER	=	-	* Keine Parameter.
	=	datei	Name der Datei mit den Parametern.
	=	*	Die Parameter folgen auf das Kommando und sind mit *EOF abgeschlossen.
PROTOKOLL	=	-STD-	* Protokoll in die Standard-Protokoll-Datei ausgeben.
	=	+	Protokoll ins Ablaufprotokoll ausgeben.
	=	datei	Name der Datei für das Protokoll.

## Leistung

Mit diesem Programm können Daten zum Drucken vorbereitet werden. In den Daten enthaltene Steuerzeichen werden dabei nicht interpretiert, sondern wie normale Zeichen behandelt. Zur Gestaltung des Ausdrucks sind u. a. folgende Angaben möglich: Anzahl der Spalten, Seiten- und Spaltenüberschrift, Art der Nummerierung, Zeilenabstand.

Außerdem können aus einer zur Spezifikation `QUELLE` angegebenen Protokoll-Datei bestimmte Seiten ausgewählt und in die zur Spezifikation `PROTOKOLL` angegebene Datei kopiert werden.

## Hinweise

Die Daten in der `QUELL`-Datei bleiben unverändert. Das Ergebnis des Programms (die zum Drucken vorbereiteten Daten) wird in die `PROTOKOLL`-Datei ausgegeben. Diese kann anschließend mit dem Kommando `#DRUCKE` ausgedruckt (auf einen Drucker ausgegeben) werden.

Sollen die in den Daten enthaltenen Steuerzeichen interpretiert werden, so müssen die Daten (statt mit dem Kommando `#DVORBEREITE`) mit dem Kommando `#FORMATIERE` bzw. `#SATZ` zum Drucken aufbereitet werden.

## Beispiele

Ausdrucken der ganzen Datei `xy` (Text-, Programm- oder Segment-Datei) auf einem HP-LaserJet, der den vom System vorgegebenen Druckernamen `pr1` hat:

```
#DV,xy,HPLJ,+  
#DR,,HPLJ,pr1
```

Ausdrucken des Segments `sort` (das z. B. das gleichnamige Kommandomakro enthält) aus der Segment-Datei `makro` auf einem PostScript-Drucker, der den vom System vorgegebenen Namen `ps2` hat:

```
#DV,makro,P,+,*  
BER      sort  
DRT      PS-12  
*EOF  
#DR,,PS-12,ps2
```



## Modi

### Wahl des Modus

Die Wahl des Modus richtet sich nach den Daten in der QUELL-Datei und nach der Nummerierungsart der Sätze:

Daten	Nummerierung	Modi
Texte	Textmodus	T, O, -STD-, S
Texte	Programmmodus	P, O
Programme	Programmmodus	P, O
Makros	Programmmodus	P, O
Protokolle	Textmodus	A, U

Falls zur Spezifikation MODUS nicht der gewünschte Modus, sondern ein Druckertyp angegeben wird, erfolgt die Wahl des Modus automatisch nach folgenden Kriterien: Wenn die Sätze der Datei im Programmmodus nummeriert sind (dies wird angenommen, wenn die Seitennummer des letzten Satzes Null ist) oder wenn die Datei eine Segment-Datei ist, wird Modus P eingestellt, andernfalls Modus T.

MODUS = T

Dieser Modus interpretiert die Satznummern im Textmodus. Vor jeden Satzanfang (also nicht bei Fortsetzungszeilen) wird die Seiten-Zeilen-Nummer und die Unterscheidungsnummer, falls diese nicht Null ist, ausgegeben. Dafür werden am linken Spaltenrand die ersten 15 Stellen benutzt. In den Kopftext (Überschrift) wird eine fortlaufende Nummer der druckaufbereiteten Seiten eingesetzt. Die Satznummern haben keinen Einfluss auf den Seitenwechsel und die in den Kopftext eingesetzte Seitennummer.

Dieser Modus wird für Dateien verwendet, die im Textmodus nummeriert und noch nicht druckaufbereitet sind, und wenn keiner der folgenden Modi geeigneter ist.

MODUS = -STD-

Dieser Modus interpretiert die Satznummern im Textmodus. Vor jeden Satzanfang (also nicht bei Fortsetzungszeilen) wird die Zeilennummer und die Unterscheidungsnummer, falls diese nicht Null ist, ausgegeben. Dafür sind am linken Spaltenrand 12 Stellen vorgesehen. Die Seitennummer der Sätze wird in den Kopftext (Überschrift) eingesetzt. Jedesmal wenn sich die Seitennummer (Teil der Satznummer) ändert, wird eine neue Seite begonnen.

Dieser Modus sollte nicht verwendet werden, wenn die Sätze so nummeriert sind, dass nur wenige Sätze mit der gleichen Seitennummer in der Datei aufeinander folgen.

MODUS = S

Dieser Modus unterscheidet sich vom Modus -STD- nur dadurch, dass keine Zeilennummer ausgegeben wird. Der Text beginnt unmittelbar am linken Spaltenrand.

Dieser Modus kann verwendet werden, wenn auf jeder Seite (d. h. in den Sätzen mit gleicher Seitennummer) ein Text steht, der jeweils auf eine eigene Seite ausgegeben werden soll, und wenn die Zeilennummern der Sätze nicht von Bedeutung sind.

MODUS = O

Bei diesem Modus bleibt die Satznummer der Sätze unberücksichtigt und wird nicht ausgegeben. Der Text beginnt unmittelbar am linken Spaltenrand. In den Kopftext (Überschrift) wird eine fortlaufende Nummer der druckaufbereiteten Seiten eingesetzt. Die Satznummern haben keinen Einfluss auf den Seitenwechsel und auf die in den Kopftext eingesetzte Seitennummer.

MODUS = P

Dieser Modus interpretiert die Satznummern im Programmmodus (bei allen anderen Modi wird die Satznummer im Textmodus interpretiert). Vor jeden Satzanfang wird die Zeilennummer und die Unterscheidungsnummer, falls diese nicht Null ist, ausgegeben. Dafür sind am linken Spaltenrand 15 Stellen vorgesehen. In den Kopftext (Überschrift) wird eine fortlaufende Nummer der druckaufbereiteten Seiten eingesetzt. Jedesmal wenn sich die Seitennummer (Teil der Satznummer) ändert (d. h. in einer Segment-Datei, wenn ein neues Segment beginnt), wird eine neue Seite begonnen.

Dieser Modus wird zum Ausdrucken von Programm- und Segment-Dateien verwendet, nicht für Text-Dateien.

MODUS = A

Bei diesem Modus wird als QUELL-Datei eine Datei erwartet, die bereits druckaufbereitete Daten enthält. Dies ist in der Regel eine Datei, die bei einem vorangehenden TUSTEP-Programm als PROTOKOLL-Datei beschrieben wurde. Wesentliches Merkmal einer solchen Datei ist, dass das erste Zeichen jedes Satzes ein Vorschubzeichen (z. B. »-« für neue Seite, »1«, »2« bis »7« für Vorschub um 1, 2 bis 7 Zeilen) oder ein anderes Steuerzeichen für das Kommando #DRUCKE ist. Aus einer solchen Datei können bestimmte Teile, die über Parameter (z. B. SKN) angegeben sind, ausgewählt und in eine andere PROTOKOLL-Datei kopiert werden. Erfolgt die Auswahl über die Seitennummer (mit Parameter SKN) so ist zu beachten, dass damit die Seitennummern aus den Satznummern der QUELL-Datei gemeint sind und nicht die Seitennummern, die evtl. in Überschriftzeilen enthalten sind. Soll über Seitennummern ausgewählt werden, die in Überschriftzeilen enthalten sind, so muss Modus U benutzt werden.

MODUS = U

In der QUELL-Datei werden, wie bei Modus A, Daten erwartet, die schon druckaufbereitet sind. Im Gegensatz zu Modus A wird bei der Auswahl über die Seitennummern mit dem Parameter SKN nicht die Seitennummer der Satznummer herangezogen, sondern die Seitennummer, die in der ersten Überschriftzeile (erste Zeile einer Seite, Vorschubzeichen »-«) steht. Welche Zahl in der ersten Überschriftzeile als Seitennummer gilt, ist beim Parameter SNR beschrieben. Der Parameter SKN ist bei Modus U obligat.

## Parameter

Jeder Parameter ist einer Parameterart zugeordnet. Sie ist jeweils als römische Zahl in [ ] angegeben. Damit ist festgelegt, in welcher Form die Angaben vom Programm erwartet werden. Die einzelnen Parameterarten sind in der Beschreibung »TUSTEP-Grundlagen« im Kapitel »{}-Parameter« bzw. »<>-Parameter« definiert.

Für manche Parameter gibt es voreingestellte Werte, die angenommen werden, falls die entsprechenden Werte nicht angegeben sind. Diese Voreinstellungen sind in < > angegeben.

Außer den im folgenden beschriebenen Parametern sind auch Parameter zur Definition von Zeichen- und Stringgruppen möglich. [ V ]

### Einstellen der Parameter-Konvention

Ob die Parameter nach der »{}-Parameter-Konvention« oder nach der »<>-Parameter-Konvention« interpretiert werden, kann mit dem Kommando #PARAMETER, {} bzw. #PARAMETER, <> eingestellt werden. Darüber hinaus kann die Konvention auch mit folgendem Parameter eingestellt werden.

**PAR** Parameter-Konvention einstellen.

{ } Nachfolgende Parameter nach der »{}-Parameter-Konvention« interpretieren.

<> Nachfolgende Parameter nach der »<>-Parameter-Konvention« interpretieren.

Die Einstellung mit dem Parameter PAR hat Vorrang vor der mit dem Kommando #PARAMETER gewählten Einstellung; sie gilt jeweils nur für die nachfolgenden Parameter bis zum nächsten Parameter PAR bzw. bis zum Ende der Parameter dieses Programms.

### Auswahl der Daten

Soll die gesamte Datei verarbeitet werden, braucht keiner der folgenden Parameter angegeben zu werden.

**BER** Angabe eines Bereichs (»Seite.Zeile–Seite.Zeile«, falls die Datei im Textmodus nummeriert ist; Zeile–Zeile, falls die Datei im Programmmodus nummeriert ist) oder einer Anfangsstelle (»Seite.Zeile« bzw. »Zeile«), falls nicht die ganze Datei verarbeitet werden soll. [ XI ]

Soll ein Segment einer Segment-Datei verarbeitet werden, kann anstelle des Bereichs der Name des Segments angegeben werden.

Dieser Parameter ist nur zugelassen, wenn die Satznummern in der Datei alle aufsteigend sind; außerdem schließen sich die Parameter BER, SKN und DAE gegenseitig aus; bei MODUS=P darf mit diesem Parameter nur ein Segmentname angegeben werden, bei MODUS=A und

MODUS=U ist dieser Parameter nicht zugelassen (Abhilfe: Auswahl mit SKN oder DAE).

**SKN** Angabe von Seitennummern (bei MODUS=P Zeilennummern, früher: Kartennummern) der Sätze, die ausgegeben werden sollen. [ XI ]

Die Angabe einer Seitennummer (Zeilennummer) bewirkt, dass alle Sätze mit dieser Seitennummer (Zeilennummer) ausgegeben werden. Sollen mehrere aufeinander folgende Seiten (Zeilen) ausgegeben werden, kann ein Bereich (»Seite-Seite« bzw. »Zeile-Zeile«) angegeben werden. Außerdem sind mehrere Angaben durch Apostroph getrennt möglich; die Seitennummern (Zeilennummern) müssen in aufsteigender Reihenfolge angegeben werden.

Dieser Parameter ist nur zugelassen, wenn die Satznummern in der Datei alle aufsteigend sind; außerdem schließen sich die Parameter BER, SKN und DAE gegenseitig aus; bei MODUS=U ist dieser Parameter obligat.

**DAE** Anzahl der auszugebenden Sätze, falls nur der Dateianfang und/oder das Dateiende ausgegeben werden soll. [ I ]

Es können zwei Zahlenwerte angegeben werden:

1. Zahl: Dateianfang <0>

Anzahl der Sätze am Dateianfang, die ausgegeben werden sollen

2. Zahl: Dateiende <0>

Anzahl der Sätze am Dateiende, die ausgegeben werden sollen

Die Parameter BER, SKN und DAE schließen sich gegenseitig aus; bei MODUS=U ist dieser Parameter nicht zugelassen (Abhilfe: MODUS=A).

**SNR** Zeichenfolgen, die die Seitennummer kennzeichnen. [ IX ]

Dieser Parameter ist nur bei MODUS=U zugelassen.

Bei MODUS=U erfolgt die Auswahl der zu kopierenden Seiten über die Seitennummer. Diese muss in der ersten Zeile jeder Seite (Zeile mit Vorschubzeichen »-«) stehen. Als Seitennummer wird die Zahl interpretiert, die als erste nach einer Zeichenfolge steht, die mit diesem Parameter angegeben ist. Ist dieser Parameter nicht angegeben, gilt als Seitennummer die Zahl, die nach der Zeichenfolge »Seite« steht bzw. die erste Zahl in der Zeile, falls die Zeichenfolge »Seite« nicht vorkommt.

## Angaben zum Protokoll

Für `MODUS=A` und `MODUS=U` müssen die Daten bereits druckaufbereitet sein. Deshalb sind für diese beiden Modi nur die Parameter zur »Auswahl der Daten« zugelassen.

**DR** Angaben zur Druckausgabesteuerung. [ 1 ]

Es können fünf Zahlenwerte angegeben werden:

1. Zahl: Spalten <1>

Anzahl der Spalten, die auf jeder Seite nebeneinander gedruckt werden sollen

2. Zahl: Rand <0>

Anzahl der Leerstellen links der ersten Spalte

3. Zahl: Breite <132>

Anzahl der Zeichen je Spalte

4. Zahl: Zwischenraum <0>

Anzahl der Leerstellen zwischen den Spalten

5. Zahl: Einrücken bei Fortsetzungszeilen <0>

Anzahl der Leerstellen am Anfang von Fortsetzungszeilen (bei Modus `T`, `P` und `-STD-` zusätzlich zu den 15 bzw. 12 Stellen, die am linken Spaltenrand für die Nummer vorgesehen sind)

**DRZ** Zusätzliche Angaben zur Druckausgabesteuerung. [ 1 ]

Es können sieben Zahlenwerte angegeben werden:

1. Zahl: Kopftext <3>

Anzahl der Zeilen für den Kopftext, einschließlich der Leerzeilen

2. Zahl: Höhe <60>

Anzahl der Zeilen je Reihe, ohne die Zeilen für den Kopf- und Fußtext

3. Zahl: Fußtext <0>

Anzahl der Zeilen für den Fußtext, einschließlich der Leerzeilen

4. Zahl: Reihen <1>

Anzahl der Reihen pro Seite. Jede Reihe besteht aus sovielen Zeilen, wie mit der 2. Zahl angegeben wird.

5. Zahl: Wiederholung des Fußtextes <0>

Anzahl der Zeilen, die vom Fußtext der Seite (von der ersten Zeile nach unten gezählt) auch nach jeder Reihe (außer nach der untersten Reihe, nach der alle Zeilen des Fußtextes gedruckt werden) wiederholt werden sollen

6. Zahl: Leerzeilen <0>

Anzahl der Leerzeilen zwischen den einzelnen Reihen

7. Zahl: Wiederholung des Kopftextes <0>

Anzahl der Zeilen, die vom Kopftext der Seite (von der letzten Zeile nach oben gezählt) auch vor jeder Reihe (außer vor der obersten Reihe, vor der alle Zeilen des Kopftextes gedruckt werden) wiederholt werden sollen

**KT**

Textteile, die als Kopftext oben auf jeder Seite gedruckt werden sollen.

[ II ] <: &Q3 @/ &D2 &U2 &#6::&Q0:>

In den Textteilen werden folgende Steueranweisungen durch die entsprechenden aktuellen Werte ersetzt:

- &Q0 Segmentname (von Parameter BER)
- &Q1 Projektname der QUELL-Datei (ohne Dateiname)
- &Q2 Dateiname der QUELL-Datei (ohne Projektname)
- &Q3 Projekt- und Dateiname der QUELL-Datei
- &D0 Wochentag (z. B. Sonntag)
- &D1 Datum xx.xx.xx (z. B. 02.04.96)
- &D2 Datum xx. xxx. xxxx (z. B. 2. Apr. 2008)
- &D3 Datum xx. xxxxxxxxxxx xxxx (z. B. 2. April 2008)
- &U1 Uhrzeit xx.xx (z. B. 12.00)
- &U2 Uhrzeit xx:xx (z. B. 12:00)
- &#n Seitennummer mit maximal n (n=1 bis 6) Stellen

Die Seitennummer kann nur einmal eingesetzt werden. Wird für die Seitennummer »- &#n -« (n=1 bis 6) angegeben, so wird die Seitennummer in die Mitte zwischen die Minuszeichen eingesetzt; die Minuszeichen werden bis auf ein Leerzeichen als Zwischenraum nach rechts bzw. links zur Seitennummer hin verschoben.

Werden pro Seite mehrere Reihen (4. Zahlwert des Parameters DRZ) ausgegeben und wird &#n in einer Zeile angegeben, die für jede Reihe wiederholt wird (4. bzw. 7. Zahlwert des Parameters DRZ), so wird statt der Seitennummer die laufende Nummer der jeweiligen Reihe eingesetzt.

Jeder der Textteile kann durch die Formatieranweisungen »@z« und »@/« in drei Teile gegliedert sein:

linksbündig @z auf Mitte zentriert @/ rechtsbündig

Diese einzelnen Teile werden linksbündig, auf Mitte zentriert und rechtsbündig eingesetzt. Jeder einzelne Teil kann (bei Teil zwei und drei einschließlich der davor stehenden Formatieranweisung) fehlen.

Jeder Textteil wird in eine neue Zeile des Kopftextes gedruckt. Bei mehrspaltigem Druck können auch Textteile für die einzelnen Spalten angegeben werden. Dazu gibt es folgende Regelung:

Beginnt ein Textteil mit »\* :«, so wird der Rest des Textteils über jede Spalte in den Kopftext eingetragen. Steht anstelle des Sterns eine Zahl, so wird der Rest des Textteils über die durch die Zahl bezeichnete Spalte eingetragen. Hat die Zahl den Wert 0, so gilt der Rest des Textteils für die ganze Zeile. Beginnt ein Textteil nicht in der beschriebenen Weise, so wird »0 :« angenommen (Normalfall).

Mit einem Textteil, der für eine ganze Zeile des Kopftextes gilt, wird immer eine neue Zeile begonnen. Ein Textteil, der über einer bestimmten Spalte stehen soll, wird in die gleiche Zeile wie der vorangehende Textteil eingetragen, falls diese Zeile nicht schon einen Text für die ganze Zeile oder für diese oder eine weiter rechts stehende Spalte enthält; andernfalls wird mit diesem Textteil eine neue Zeile begonnen.

**FT** Textteile, die als Fußtext unten auf jeder Seite gedruckt werden sollen. [ II ] <>

Es gelten die gleichen Regelungen wie für den Kopftext (Parameter  $\kappa T$ ). Die Seitennummer kann jedoch nicht in den Fußtext eingesetzt werden, wenn sie schon in den Kopftext eingesetzt worden ist.

**LZ** Anzahl der zusätzlichen Leerzeilen. [ I ]

Es können zwei Zahlenwerte angegeben werden:

1. Zahl: Anzahl der Leerzeilen vor jeder Zeile, in der ein neuer Satz beginnt <0>
2. Zahl: Anzahl der Leerzeilen vor jeder Fortsetzungszeile <0>

**DRT** Druckertyp, für den die Daten aufbereitet werden. [ XI ]

Dieser Parameter darf nicht angegeben werden, wenn zu Spezifikation  $MODUS$  schon ein Druckertyp angegeben wurde; außerdem ist er bei  $MODUS=A$  und bei  $MODUS=U$  nicht zulässig. In allen anderen Fällen ist dieser Parameter obligat, wenn das Protokoll in eine Datei (d. h. nicht ins Ablaufprotokoll) ausgegeben werden soll.

Hinweis: Mit dem Kommando  $\#LISTE, DRUCKERTYPEN$  werden die definierten Druckertypen aufgelistet.

**STZ** Angabe, ob die Steuerzeichen # (für Sonderzeichen, Auszeichnungen, Schriftumschaltungen, Druckeffekte) und % (für Akzente) interpretiert werden sollen, und ob das Steuerzeichen \_ (für festes Blank) als Leerstelle ausgedruckt und das Steuerzeichen \ (für Kann-Trennstelle) nicht ausgedruckt werden soll. [ I ] <0>

0 = Steuerzeichen ausdrucken, nicht interpretieren.

1 = Steuerzeichen #, %, \_ und \ interpretieren.



## Alphabetisches Verzeichnis der Parameter

<b>BER</b>	Auswählen eines Bereichs aus der QUELL-Datei . . . . .	852
<b>DAE</b>	Auswählen von Dateianfang / Dateiende . . . . .	853
<b>DR</b>	Druckausgabesteuerung . . . . .	854
<b>DRT</b>	Druckertyp . . . . .	856
<b>DRZ</b>	Druckausgabesteuerung – zusätzliche Angaben . . . . .	854
<b>FT</b>	Fußtext . . . . .	856
<b>KT</b>	Kopfertext . . . . .	855
<b>LZ</b>	Leerzeilen . . . . .	856
<b>PAR</b>	Parameter-Konvention . . . . .	852
<b>SKN</b>	Auswählen von Seiten . . . . .	853
<b>SNR</b>	Kennzeichnung der Seitennummer . . . . .	853
<b>STZ</b>	Steuerzeichen interpretieren . . . . .	856

\* \* \* \* \*



**#EINFUEGE**

---

Kommando . . . . .	861
Leistung . . . . .	861
Beispiel . . . . .	862
Anforderung an die Daten . . . . .	863
Wahl des Modus . . . . .	864
Arbeitsweise des Programms . . . . .	866
MODUS = PARALLEL . . . . .	866
MODUS = SORTIERT . . . . .	866
MODUS = MEHR . . . . .	866
MODUS = WENIGER . . . . .	867
MODUS = -STD- . . . . .	867
MODUS = KURZ . . . . .	867
MODUS = LANG . . . . .	867
MODUS = GRUPPEN . . . . .	868
Parameter . . . . .	869
Einstellen der Parameter-Konvention . . . . .	869
Auswahl der Daten in der QUELL-Datei . . . . .	869
Länge der Kürzel . . . . .	870
Kennzeichnung der Kürzel in der QUELL-Datei . . . . .	870
Kennzeichnung der Kürzel in der KUERZEL-Datei . . . . .	870
Kennzeichnung einer Gruppe in der KUERZEL-Datei . . . . .	871
Kennzeichnung des eingefügten Textteils . . . . .	871
Austauschen zum Vergleich der Kürzel . . . . .	872
Fehlerprotokoll . . . . .	872
Alphabetisches Verzeichnis der Parameter . . . . .	873

## Kommando

### #EINFUEGE

QUELLE	= datei	Name der Datei mit den Daten, in denen Textteile eingefügt werden sollen.
	= -STD-	Die Daten, in denen Textteile eingefügt werden sollen, stehen in der Standard-Text-Datei.
ZIEL	= datei	Name der Datei für die Daten mit den eingefügten Textteilen.
	= -STD-	Die Daten mit den eingefügten Textteilen in die Standard-Text-Datei ausgeben.
MODUS	= -STD-	* Kürzel eindeutig, Normalfall.
	= KURZ	Kürzel eindeutig, kurze einzufügende Textteile.
	= LANG	Kürzel eindeutig, lange einzufügende Textteile.
	= PARALLEL	Kürzel kommen in der KUERZEL-Datei und der QUELL-Datei parallel vor.
	= SORTIERT	Kürzel sind alphabetisch sortiert.
	= MEHR	Die KUERZEL-Datei enthält mehr Kürzel als die QUELL-Datei.
	= WENIGER	Die KUERZEL-Datei enthält weniger Kürzel als die QUELL-Datei.
	= GRUPPEN	Die KUERZEL-Datei enthält Gruppen mit jeweils den gleichen Kürzeln. Für jede Gruppe soll die QUELL-Datei neu kopiert werden.
LOESCHEN	= -	* Daten in der ZIEL-Datei nicht löschen.
	= +	Daten in der ZIEL-Datei zuerst löschen.
PARAMETER	= datei	Name der Datei mit den Parametern.
	= *	Die Parameter folgen auf das Kommando und sind mit *EOF abgeschlossen.
KUERZEL	= datei	Name der Datei mit den einzufügenden Textteilen.

## Leistung

Mit diesem Programm können Textteile, die in einer Datei stehen und über Kürzel (z. B. eine laufende Nummer) identifiziert werden, in die Daten einer anderen Datei eingefügt (eingemischt) werden. Sie werden jeweils an der Stelle in die Daten eingefügt, an der das gleiche Kürzel steht.

Anwendungsbeispiele (in Klammern ist angegeben, welche Daten dabei in der KUERZEL-Datei stehen):

- Fußnoten in den Text einmischen (Fußnoten)
- Kurzformen durch Volltext ersetzen (Volltext)
- Bausteinbriefe zusammenstellen (Bausteine)
- Serienbriefe erstellen (eine Adresse je Gruppe)

## Beispiel

Das Kommando

```
#EI, tdat, edat, P, +, *, fdat
AKD      / (/
EKD      /) /
AKK      /@ /
EKK      / /
NAE      / @f+ / @f- /
*EOF
```

fügt in den Text aus der Datei tdat

```
Text (1) zum
Einfügen (2) von
Fußnoten. (3)
```

die Fußnoten aus der Datei fdat ein

```
@1 Erste Fußnote
@2 Zweite @3 Dritte,
die sich über mehrere
Zeilen erstreckt.
```

und schreibt das Ergebnis in die Datei edat

```
Text @f+ Erste Fußnote @f- zum
Einfügen @f+ Zweite @f- von
Fußnoten. @f+ Dritte,
die sich über mehrere
Zeilen erstreckt. @f-
```

## Anforderungen an die Daten

Das Programm erwartet die Daten, in die Textteile eingefügt werden sollen, in der QUELL-Datei und die einzufügenden Textteile in der KUERZEL-Datei. Das Ergebnis wird in die ZIEL-Datei ausgegeben.

QUELL-Datei: An der Stelle, an der ein Textteil in die Daten eingefügt werden soll, muss das entsprechende Kürzel stehen. Dieses muss eindeutig mit einer Anfangskennung davor und einer Endekennung danach gekennzeichnet sein. Die Anfangskennung muss mit dem Parameter AKD, die Endekennung mit dem Parameter EKD angegeben werden. Falls hinter den Kürzeln in der gleichen Zeile keine weiteren Daten mehr stehen, kann die Endekennung fehlen, und der Parameter EKD braucht nicht angegeben zu werden. Ob die Kennzeichen für das Kürzel und das Kürzel selbst erhalten bleiben oder eliminiert werden sollen, kann mit dem Parameter KKZ gesteuert werden. Falls erforderlich, kann vor und/oder hinter dem eingefügten Textteil noch zusätzlich jeweils eine Zeichenfolge ergänzt werden; diese Zeichenfolgen können mit dem Parameter NAE angegeben werden.

KUERZEL-Datei: Jedem einzufügenden Textteil muss ein Kürzel vorangestellt sein, über das er identifiziert wird. Diese Kürzel müssen ebenfalls eindeutig mit einer Anfangskennung davor und einer Endekennung danach gekennzeichnet sein. Die Anfangskennung muss mit dem Parameter AKK, die Endekennung mit dem Parameter EKK angegeben werden. Falls hinter den Kürzeln in der gleichen Zeile keine weiteren Daten mehr stehen, kann die Endekennung fehlen, und der Parameter EKK braucht nicht angegeben zu werden. Eine Kennzeichnung der Textteile ist nicht notwendig. Jeder Textteil beginnt nach der Endekennung für das Kürzel und endet vor der Anfangskennung des Kürzels für den nächsten Textteil. Ein Textteil kann sich über mehrere Zeilen erstrecken. Die Zeilenwechsel innerhalb eines Textteils werden beibehalten, wenn der Textteil eingefügt wird. Sind die Kürzel in Gruppen eingeteilt, wobei jede Gruppe jeweils die gleichen Kürzel, aber verschiedene dazugehörige Textteile enthält, so muss am Beginn jeder Gruppe eine eindeutige Kennung stehen, die mit dem Parameter NKG angegeben werden muss.

Sowohl in der QUELL-Datei als auch in der KUERZEL-Datei dürfen mehrere Kürzel (jeweils mit Anfangs- und Endekennung) in einer Zeile stehen. Jedoch müssen zusammengehörende Anfangskennung, Kürzel und Endekennung jeweils vollständig in der gleichen Zeile stehen.

## Wahl des Modus

Wenn die Kürzel in der KUERZEL-Datei in Gruppen eingeteilt sind und für jede Gruppe die QUELL-Datei erneut in die ZIEL-Datei kopiert werden soll, muss der folgende Modus gewählt werden:

MODUS = GRUPPEN

Bei diesem Modus darf in der QUELL-Datei jedes Kürzel beliebig oft vorkommen, aber in der KUERZEL-Datei darf innerhalb einer Gruppe jedes Kürzel nur einmal vorkommen. Die Reihenfolge der Kürzel ist belanglos. Die zu den in der QUELL-Datei vorkommenden Kürzeln gehörenden Textteile einer Gruppe müssen jeweils klein genug sein, um vollständig in den Hauptspeicher kopiert werden zu können.

Wenn die Kürzel in der QUELL-Datei und in der KUERZEL-Datei in der gleichen Reihenfolge stehen, kann einer der folgenden Modi gewählt werden:

MODUS = PARALLEL

wenn jedem Kürzel in der QUELL-Datei ein Kürzel in der KUERZEL-Datei entspricht und umgekehrt (Ausnahme siehe »Arbeitsweise des Programms«).

MODUS = SORTIERT

wenn die Kürzel in QUELL- und KUERZEL-Datei alphabetisch sortiert sind.

Zum Sortieren muss im ersten Sortierschlüssel die Standard-Sortierfolge, im zweiten Sortierschlüssel die Sonder-Sortierfolge zugrunde gelegt werden.

MODUS = MEHR

wenn in der KUERZEL-Datei mehr Kürzel stehen als in der QUELL-Datei, aber alle in der QUELL-Datei vorkommenden Kürzel auch in der KUERZEL-Datei stehen.

MODUS = WENIGER

wenn in der KUERZEL-Datei weniger Kürzel stehen als in der QUELL-Datei, aber alle in der KUERZEL-Datei vorkommenden Kürzel auch in der QUELL-Datei stehen.

Wenn die Reihenfolge der Kürzel in der QUELL-Datei und in der KUERZEL-Datei nicht übereinstimmt, darf nur einer der folgenden Modi gewählt werden; dabei darf in der QUELL-Datei jedes Kürzel beliebig oft vorkommen, aber in der KUERZEL-Datei darf jedes Kürzel nur einmal vorkommen. Die Reihenfolge der Kürzel ist belanglos:

MODUS = -STD-

wenn die KUERZEL-Datei klein genug ist, um vollständig in den Hauptspeicher kopiert werden zu können.



MODUS = KURZ

wenn der Hauptspeicher für MODUS=STD nicht ausreicht und die zu den in der QUELL-Datei vorkommenden Kürzeln gehörenden Textteile in der KUERZEL-Datei kurz sind oder nicht viele verschiedene Kürzel in der QUELL-Datei vorkommen, so dass die benötigten Textteile von der KUERZEL-Datei in den Hauptspeicher kopiert werden können.

MODUS = LANG

wenn der Hauptspeicher für MODUS=KURZ nicht ausreicht.

Wenn der Hauptspeicher auch für MODUS=LANG nicht ausreicht, muss entweder die QUELL- und KUERZEL-Datei so sortiert werden, dass die Kürzel in alphabetischer Reihenfolge stehen, damit MODUS=Sortiert benutzt werden kann, oder das Programm muss mehrmals mit MODUS=LANG aufgerufen werden, wobei mit dem Parameter BER jeweils ein Bereich aus der QUELL-Datei ausgewählt wird.

## Arbeitsweise des Programms

Bei allen Modi werden Kürzel, die in der QUELL-Datei vorkommen, aber in der KUERZEL-Datei nicht gefunden werden, unverändert in die ZIEL-Datei übernommen. Ob in diesem Fall eine Fehlermeldung erfolgen soll, kann mit dem Parameter UND angegeben werden.

Beim Vergleich von Kürzeln aus der QUELL- und der KUERZEL-Datei gelten Groß- und Kleinbuchstaben als gleichwertig, falls mit dem Parameter GKU nichts anders bestimmt wird.

Für die Modi PARALLEL, SORTIERT, MEHR und WENIGER gilt:

Das Programm liest die Daten von der QUELL-Datei und schreibt sie in die ZIEL-Datei. Dabei werden die Daten nach Kürzeln durchsucht. Wird ein Kürzel gefunden, wird es mit dem nächsten Kürzel in der KUERZEL-Datei verglichen. Sind beide gleich, so wird der entsprechende Textteil von der KUERZEL-Datei geholt und in die Daten eingefügt. Andernfalls wird das in der QUELL-Datei gefundene mit dem Kürzel in der KUERZEL-Datei verglichen, dessen Textteil zuletzt eingefügt wurde. Stimmen diese beiden überein, wird der entsprechende Textteil von der KUERZEL-Datei geholt und in die Daten eingefügt. Stimmen auch diese beiden nicht überein, ist es vom angegebenen Modus abhängig, wie weiter verfahren wird:

MODUS = PARALLEL

Wenn wie oben beschrieben keine Übereinstimmung gefunden wurde, wird das Programm abgebrochen.

MODUS = SORTIERT

Wenn wie oben beschrieben keine Übereinstimmung gefunden wurde, wird solange in der KUERZEL-Datei weitergesucht, bis das Kürzel gefunden wird bzw. bis dort ein Kürzel kommt, das bei alphabetischer Ordnung (im ersten Sortierschlüssel wird die Standard-Sortierfolge, im zweiten Sortierschlüssel die Sonder-Sortierfolge zugrunde gelegt) danach steht. Im letzteren Fall gilt das Kürzel als nicht gefunden; es wird nach dem nächsten Kürzel in der QUELL-Datei gesucht.

MODUS = MEHR

Wenn wie oben beschrieben keine Übereinstimmung gefunden wurde, wird solange in der KUERZEL-Datei weitergesucht, bis das Kürzel gefunden wird. Die dabei in der KUERZEL-Datei übergangenen Kürzel werden nicht mehr berücksichtigt.

Achtung: Wenn also ein Kürzel bis zum Ende der KUERZEL-Datei nicht gefunden wird, können auch die in der QUELL-Datei folgenden nicht mehr gefunden werden, auch wenn sie in der KUERZEL-Datei stehen.

MODUS = WENIGER

Wenn wie oben beschrieben keine Übereinstimmung gefunden wurde, wird solange in der QUELL-Datei weitergesucht, bis das Kürzel gefunden wird. Die dabei in der QUELL-Datei übergangenen Kürzel gelten als nicht gefunden und bleiben unverändert.

MODUS = -STD-

Das Programm liest von der KUERZEL-Datei die Kürzel mit den dazugehörigen Textteilen ein und erstellt eine interne Tabelle.

Nachdem die Tabelle erstellt ist, werden die Daten von der QUELL-Datei gelesen und in die ZIEL-Datei geschrieben. Dabei werden die Daten nach Kürzeln durchsucht. Wird ein Kürzel gefunden, wird der entsprechende Textteil aus der Tabelle geholt und in die Daten eingefügt.

MODUS = KURZ

Das Programm liest zunächst die QUELL-Datei und erstellt mit den darin vorkommenden Kürzeln intern eine Tabelle. Dann werden in der KUERZEL-Datei diese Kürzel gesucht und die Tabelle um die dazugehörigen Textteile ergänzt.

Nachdem die Tabelle erstellt ist, werden die Daten von der QUELL-Datei gelesen und in die ZIEL-Datei geschrieben. Dabei werden die Daten nach Kürzeln durchsucht. Wird ein Kürzel gefunden, wird der entsprechende Textteil aus der Tabelle geholt und in die Daten eingefügt.

Dieser Modus hat gegenüber dem Modus -STD- den Nachteil, dass die QUELL-Datei zweimal statt nur einmal gelesen werden muss und der Zeitaufwand entsprechend größer ist.

MODUS = LANG

Das Programm liest zunächst die QUELL-Datei und erstellt mit den darin vorkommenden Kürzeln intern eine Tabelle. Dann werden in der KUERZEL-Datei diese Kürzel gesucht und die Tabelle durch einen Zeiger ergänzt. Dieser Zeiger gibt die Stelle in der KUERZEL-Datei an, an der das Kürzel steht.

Nachdem die Tabelle erstellt ist, werden die Daten von der QUELL-Datei gelesen und in die ZIEL-Datei geschrieben. Dabei werden die Daten nach Kürzeln durchsucht. Wird ein Kürzel gefunden, wird mit Hilfe des Zeigers der entsprechende Textteil aus der KUERZEL-Datei geholt und in die Daten eingefügt.

Dieser Modus hat gegenüber dem Modus KURZ den Nachteil, dass die KUERZEL-Datei im Random-Zugriff statt sequentiell gelesen werden muss und der Zeitaufwand deshalb um ein vielfaches größer ist.

MODUS = GRUPPEN

Das Programm liest zunächst die QUELL-Datei und erstellt mit den darin vorkommenden Kürzeln intern eine Tabelle. Dann werden in der KUERZEL-Datei diese Kürzel in der ersten Gruppe gesucht und die Tabelle um die dazugehörigen Textteile ergänzt.

Nachdem die Tabelle erstellt ist, werden die Daten von der QUELL-Datei gelesen und in die ZIEL-Datei geschrieben. Dabei werden die Daten nach Kürzeln durchsucht. Wird ein Kürzel gefunden, wird der entsprechende Textteil aus der Tabelle geholt und in die Daten eingefügt.

Nachdem alle Daten der QUELL-Datei abgearbeitet sind, werden die Textteile in der Tabelle gelöscht und die in der Tabelle stehenden Kürzel in der KUERZEL-Datei in der nächsten Gruppe gesucht und die dazugehörigen Textteile in die Tabelle eingetragen. Dann wird die QUELL-Datei wieder von vorne abgearbeitet. Dieses Verfahren wird für jede Gruppe in der KUERZEL-Datei wiederholt.

## Parameter

Jeder Parameter ist einer Parameterart zugeordnet. Sie ist jeweils als römische Zahl in [ ] angegeben. Damit ist festgelegt, in welcher Form die Angaben vom Programm erwartet werden. Die einzelnen Parameterarten sind in der Beschreibung »TUSTEP-Grundlagen« im Kapitel »{}-Parameter« bzw. »<>-Parameter« definiert.

Für manche Parameter gibt es voreingestellte Werte, die angenommen werden, falls die entsprechenden Werte nicht angegeben sind. Diese Voreinstellungen sind in < > angegeben.

Außer den im folgenden beschriebenen Parametern sind auch Parameter zur Definition von Zeichen- und Stringgruppen möglich. [ V ]

### Einstellen der Parameter-Konvention

Ob die Parameter nach der »{}-Parameter-Konvention« oder nach der »<>-Parameter-Konvention« interpretiert werden, kann mit dem Kommando #PARAMETER, {} bzw. #PARAMETER, <> eingestellt werden. Darüber hinaus kann die Konvention auch mit folgendem Parameter eingestellt werden.

- PAR** Parameter-Konvention einstellen.
- { } Nachfolgende Parameter nach der »{}-Parameter-Konvention« interpretieren.
  - <> Nachfolgende Parameter nach der »<>-Parameter-Konvention« interpretieren.

Die Einstellung mit dem Parameter PAR hat Vorrang vor der mit dem Kommando #PARAMETER gewählten Einstellung; sie gilt jeweils nur für die nachfolgenden Parameter bis zum nächsten Parameter PAR bzw. bis zum Ende der Parameter dieses Programms.

### Auswahl der Daten in der QUELL-Datei

Soll die gesamte Datei bearbeitet werden, braucht keiner der folgenden Parameter angegeben zu werden.

- BER** Angabe eines Bereichs (»Seite.Zeile–Seite.Zeile«, falls die Datei im Textmodus nummeriert ist; Zeile–Zeile, falls die Datei im Programmmodus nummeriert ist) oder einer Anfangsstelle (»Seite.Zeile« bzw. »Zeile«), falls nicht die ganze Datei bearbeitet werden soll. [ XI ]

Soll ein Segment einer Segment-Datei verarbeitet werden, kann anstelle des Bereichs der Name des Segments angegeben werden.

Dieser Parameter ist nur zugelassen, wenn die Satznummern in der Datei alle aufsteigend sind.

**MAX** Angabe für Testzwecke, wieviele Eingabesätze maximal bearbeitet werden sollen. [ 1 ] <999999999>

### Länge der Kürzel

**MKL** Maximale Länge eines Kürzels (ohne Anfangs- und Endekennung). Ist ein Kürzel länger, so kann es nicht berücksichtigt werden. [ 1 ] <8>

### Groß- und Kleinbuchstaben in Kürzeln

**GKU** Angabe, ob Groß- und Kleinbuchstaben beim Vergleich auf Übereinstimmung von Kürzeln unterschieden werden sollen oder nicht. [ 1 ] <0>

0 = Groß- und Kleinbuchstaben nicht unterscheiden

1 = Groß- und Kleinbuchstaben unterscheiden

### Kennzeichnung der Kürzel in der QUELL-Datei

Von den beiden folgenden Parametern muss der Parameter **AKD** immer angegeben werden. Der Parameter **EKD** ist nur notwendig, wenn nach dem Kürzel in der gleichen Zeile noch weitere Daten folgen.

**AKD** Zeichenfolgen, die den Anfang eines Kürzels kennzeichnen. [ IX ]

**EKD** Zeichenfolgen, die das Ende eines Kürzels kennzeichnen. [ IX ]

Das Kürzel beginnt jeweils nach der Anfangskennung und endet vor der Endekennung bzw. am Ende der Zeile. Enthält die so als Kürzel abgegrenzte Zeichenfolge noch führende und/oder abschließende Leerzeichen, so werden diese Leerzeichen ignoriert.

Mit der Suche nach der Anfangskennung für das nächste Kürzel wird jeweils nach der Endekennung eines Kürzels begonnen. Wurde für das Kürzel keine Endekennung mehr in der gleichen Zeile gefunden, wird mit der Suche nach der nächsten Anfangskennung am Anfang der folgenden Zeile begonnen.

### Kennzeichnung der Kürzel in der KUERZEL-Datei

Von den beiden folgenden Parametern muss der Parameter **AKK** immer angegeben werden. Der Parameter **EKK** ist nur notwendig, wenn nach dem Kürzel in der gleichen Zeile noch weitere Daten folgen.

**AKK** Zeichenfolgen, die den Anfang eines Kürzels kennzeichnen. [ IX ]

**EKK** Zeichenfolgen, die das Ende eines Kürzels kennzeichnen. [ IX ]

Das Kürzel beginnt jeweils nach der Anfangskennung und endet vor der Endekennung bzw. am Ende der Zeile. Enthält die so als Kürzel

abgegrenzte Zeichenfolge noch führende und/oder abschließende Leerzeichen, so werden diese Leerzeichen ignoriert.

Mit der Suche nach der Anfangskennung für das nächste Kürzel wird jeweils nach der Endekennung eines Kürzels begonnen. Wurde für das Kürzel keine Endekennung mehr in der gleichen Zeile gefunden, wird mit der Suche nach der nächsten Anfangskennung am Anfang der folgenden Zeile begonnen.

### Kennzeichnung einer Gruppe in der **KUERZEL**-Datei

Dieser Parameter ist bei **MODUS=GRUPPEN** obligat und nur bei diesem Modus zugelassen. Mit ihm wird angegeben, wie der Beginn einer Gruppe von Kürzeln gekennzeichnet ist. Das Kennzeichen muss in der **KUERZEL**-Datei jeweils am Anfang der Sätze stehen, mit denen eine neue Gruppe beginnt. Für diese Kennzeichnung kann auch eine Zeichenfolge dienen, die gleichzeitig den Anfang eines Kürzels kennzeichnet (also auch mit dem Parameter **AKK** angegeben ist).

**NKG** Zeichenfolgen, die am Satzanfang den Beginn einer neuen Kürzelgruppe kennzeichnen. [ IX ]

### Kennzeichnung des eingefügten Textteils in der **ZIEL**-Datei

**KKZ** Zusatzangaben, ob die Kennzeichen der Kürzel und das Kürzel aus der **QUELL**-Datei in die **ZIEL**-Datei übernommen werden sollen. [ I ]

Es können drei Zahlenwerte angegeben werden:

1. Zahl: Die Anfangskennung des Kürzels <0>

0 = nicht übernehmen

1 = übernehmen

2. Zahl: Das Kürzel <0>

0 = nicht übernehmen

1 = übernehmen

3. Zahl: Die Endekennung des Kürzels <0>

0 = nicht übernehmen

1 = übernehmen und den Textteil danach einfügen

2 = übernehmen und den Textteil davor einfügen

**NAE** Textteile (neue Anfangs- und Endekennungen), die am Anfang bzw. am Ende des eingefügten Textteils ergänzt werden sollen. [ II ]

Es können zwei Textteile angegeben werden, von denen der erste unmittelbar vor, der zweite unmittelbar hinter dem eingefügten Textteil ergänzt wird.

## Austauschen zum Vergleich der Kürzel

Beim Vergleich der Kürzel aus der QUELL-Datei mit denen in der KUERZEL-Datei werden die Kürzel in unveränderter Form verglichen. Groß- und Kleinbuchstaben gelten dabei als gleichwertig. Mit dem folgenden Parameter können vor dem Vergleich Zeichenfolgen in den Kürzeln ausgetauscht werden. Dieses Austauschen dient nur zum Vergleich; das Kürzel wird ggf. in unveränderter Form in die ZIEL-Datei ausgegeben.

**XKD** Zeichenfolgenpaare (und Ausnahmezeichenfolgen), deren jeweils erste Zeichenfolge durch die jeweils zweite Zeichenfolge in den Kürzeln, die aus der QUELL-Datei stammen, vor dem Vergleich ersetzt werden soll. [ X ]

## Fehlerprotokoll

**UND** Angabe, ob die Kürzel der QUELL-Datei, die in der KUERZEL-Datei nicht gefunden wurden, im Ablaufprotokoll protokolliert werden sollen oder nicht. [ I ] <0>

- 0 = nicht protokollieren, nicht als Fehler anrechnen
- 1 = protokollieren, nicht als Fehler anrechnen
- 2 = protokollieren, abrechnen bei zu vielen Fehlern
- 3 = wie 0, auch keine Sammelmeldung zum Schluss

**UNG** Angabe, ob die Kürzel der KUERZEL-Datei, die in der QUELL-Datei nicht gefunden wurden, im Ablaufprotokoll protokolliert werden sollen oder nicht. [ I ] <0>

- 0 = nicht protokollieren, nicht als Fehler anrechnen
- 1 = protokollieren, nicht als Fehler anrechnen
- 2 = protokollieren, abrechnen bei zu vielen Fehlern
- 3 = wie 0, auch keine Sammelmeldung zum Schluss



## Alphabetisches Verzeichnis der Parameter

<b>AKD</b>	Anfangskennung eines Kürzels in der QUELL-Datei . . . . .	870
<b>AKK</b>	Anfangskennung eines Kürzels in der KUERZEL-Datei . . . . .	870
<b>BER</b>	Auswählen eines Bereichs aus der QUELL-Datei . . . . .	869
<b>EKD</b>	Endekennung eines Kürzels in der QUELL-Datei . . . . .	870
<b>EKK</b>	Endekennung eines Kürzels in der KUERZEL-Datei . . . . .	870
<b>GKU</b>	Groß- und Kleinbuchstaben unterscheiden . . . . .	870
<b>KKZ</b>	Zusatzangaben zu den Kennzeichen der Kürzel . . . . .	871
<b>MAX</b>	Maximum für Testzwecke . . . . .	870
<b>MKL</b>	Maximale Länge eines Kürzels (Kürzellänge) . . . . .	870
<b>NAE</b>	Neue Anfangs- und Endekennungen . . . . .	871
<b>NKG</b>	Kennzeichen für neue Kürzelgruppe . . . . .	871
<b>PAR</b>	Parameter-Konvention . . . . .	869
<b>UND</b>	Fehlerprotokoll für undefinierte Kürzel . . . . .	872
<b>UNG</b>	Fehlerprotokoll für ungenutzte Kürzel . . . . .	872
<b>XKD</b>	Austauschen zum Vergleich der Kürzel . . . . .	872

\* \* \* \* \*



**#FAUFBEREITE**

---

Kommando . . . . .	877
Leistung . . . . .	877
Parameter . . . . .	878
Einstellen der Parameter-Konvention . . . . .	878
Auswahl der Daten . . . . .	878
Angaben zum Protokoll . . . . .	879
Angabe zur Nummerierung . . . . .	882
Zusammenfassen mehrerer Sätze zu einer Texteinheit . . . . .	882
Ersetzen von Zeichenfolgen . . . . .	885
Ergänzen eines Textteils . . . . .	885
Aufteilen der Texteinheit in Formularfelder . . . . .	885
Alphabetisches Verzeichnis der Parameter . . . . .	890

## Kommando

#FAUFBEREITE

QUELLE	= <code>datei</code>	Name der Datei mit den Daten, aus denen Formulare erstellt werden sollen.
	= <code>-STD-</code>	Die Daten, aus denen Formulare erstellt werden sollen, stehen in der Standard-Text-Datei.
MODUS	= <code>-</code>	* Reihenfolge im Protokoll wie in den Eingabedaten.
	= <code>-STD-</code>	Die Eingabedaten so umsortieren, dass die Formulare nach dem Schneiden stapelweise sortiert sind.
LOESCHEN	= <code>-</code>	* Daten in der <code>PROTOKOLL</code> -Datei nicht löschen.
	= <code>+</code>	Daten in der <code>PROTOKOLL</code> -Datei zuerst löschen.
PARAMETER	= <code>datei</code>	Name der Datei mit den Parametern.
	= <code>*</code>	Die Parameter folgen auf das Kommando und sind mit <code>*EOF</code> abgeschlossen.
FORMULAR	= <code>-</code>	* Kein Formular als Grundlage.
	= <code>datei</code>	Name der Datei mit dem Formular, das als Grundlage dient.
PROTOKOLL	= <code>-STD-</code>	* Das Protokoll mit den Formularen in die Standard-Protokoll-Datei ausgeben.
	= <code>datei</code>	Name der Datei für das Protokoll mit den Formularen.

## Leistung

Mit diesem Programm können Daten zum Drucken in einem vorgegebenen Format (z. B. Adressaufkleber, Bibliothekskärtchen, vorgedruckte Formulare) aufbereitet werden. Dabei kann u. a. angegeben werden:

- die Formulargröße,
- konstante Texte für jedes Formular (auch abhängig vom Vorkommen bestimmter Textteile im jeweiligen Formular),
- Zeilen- und Zeichenpositionen für die einzelnen Textteile,
- welche Textteile in Fortsetzungsformularen wiederholt werden sollen, falls ein Formular nicht ausreicht.

## Parameter

Jeder Parameter ist einer Parameterart zugeordnet. Sie ist jeweils als römische Zahl in [ ] angegeben. Damit ist festgelegt, in welcher Form die Angaben vom Programm erwartet werden. Die einzelnen Parameterarten sind in der Beschreibung »TUSTEP-Grundlagen« im Kapitel »{}-Parameter« bzw. »<>-Parameter« definiert.

Für manche Parameter gibt es voreingestellte Werte, die angenommen werden, falls die entsprechenden Werte nicht angegeben sind. Diese Voreinstellungen sind in < > angegeben.

Mit bestimmten Parametern können über Anfangs- und/oder Endekennungen sowie über Kennungen, die als öffnende bzw. schließende Klammern dienen, Textteile für die weitere Bearbeitung ausgewählt werden. Die Funktionsweise dieser Parameter ist auch in den »TUSTEP-Grundlagen« am Ende des Kapitels »Parameter« beschrieben.

Außer den im folgenden beschriebenen Parametern sind auch Parameter zur Definition von Zeichen- und Stringgruppen möglich. [ V ]

Parameter, die mit + gekennzeichnet sind, müssen in Spalte 7 eine 2 enthalten, wenn sie für die Fortsetzungsformulare gelten sollen.

### Einstellen der Parameter-Konvention

Ob die Parameter nach der »{}-Parameter-Konvention« oder nach der »<>-Parameter-Konvention« interpretiert werden, kann mit dem Kommando #PARAMETER, {} bzw. #PARAMETER, <> eingestellt werden. Darüber hinaus kann die Konvention auch mit folgendem Parameter eingestellt werden.

- PAR** Parameter-Konvention einstellen.
- { } Nachfolgende Parameter nach der »{}-Parameter-Konvention« interpretieren.
  - <> Nachfolgende Parameter nach der »<>-Parameter-Konvention« interpretieren.

Die Einstellung mit dem Parameter PAR hat Vorrang vor der mit dem Kommando #PARAMETER gewählten Einstellung; sie gilt jeweils nur für die nachfolgenden Parameter bis zum nächsten Parameter PAR bzw. bis zum Ende der Parameter dieses Programms.

### Auswahl der Daten

Soll die gesamte Datei verarbeitet werden, braucht keiner der folgenden Parameter angegeben zu werden.

- BER** Angabe eines Bereichs (»Seite.Zeile–Seite.Zeile«, falls die Datei im Textmodus nummeriert ist; Zeile–Zeile, falls die Datei im Programmmodus nummeriert ist) oder einer Anfangsstelle (»Seite.Zeile« bzw. »Zeile«), falls nicht die ganze Datei verarbeitet werden soll. [ XI ]

Soll ein Segment einer Segment-Datei verarbeitet werden, kann anstelle des Bereichs der Name des Segments angegeben werden.

Dieser Parameter ist nur zugelassen, wenn die Satznummern in der Datei alle aufsteigend sind.

**MAX**

Angabe zu Testzwecken, von wievielen Texteinheiten maximal Formulare aufbereitet werden sollen bzw. wieviele Formulare (einschließlich der Fortsetzungsformulare) maximal aufbereitet bzw. wieviele Seiten maximal aufbereitet werden sollen. [ 1 ]

Es können drei Zahlenwerte angegeben werden:

1. Zahl: Aufzubereitende Texteinheiten <999999999>
2. Zahl: Aufzubereitende Formulare <999999999>
3. Zahl: Aufzubereitende Seiten <999999999>

**Angaben zum Protokoll****DR**

Angaben zur Druckausgabesteuerung. [ 1 ]

Es können sieben Zahlenwerte angegeben werden:

1. Zahl: Spalten <1>  
Anzahl der Spalten (Formulare), die auf jeder Seite nebeneinander gedruckt werden sollen
2. Zahl: Rand <10>  
Anzahl der Leerstellen links der ersten Spalte
3. Zahl: Breite <64>  
Anzahl der Zeichen je Spalte (Formular)
4. Zahl: Zwischenraum <0>  
Anzahl der Leerstellen zwischen den Spalten (Formularen)
5. Zahl: Einrücken bei Fortsetzungszeilen <0>  
Anzahl der Leerstellen am Anfang von Fortsetzungszeilen, bevor ein Zeilenwechsel durch eine mit dem Parameter *zw* angegebene Zeichenfolge erzwungen wurde
6. Zahl: Einrücken nach Zeilenwechsel <0>  
Anzahl der Leerstellen am Anfang einer Zeile nach einem Zeilenwechsel, der durch eine mit dem Parameter *zw* angegebene Zeichenfolge erzwungen wurde
7. Zahl: Leerstellen bei Fortsetzungszeilen <0>  
Leerstellen am Anfang von Fortsetzungszeilen, nachdem ein Zeilenwechsel durch eine mit dem Parameter *zw* angegebene Zeichenfolge erzwungen wurde

**DRZ**

Zusätzliche Angaben zur Druckausgabesteuerung. [ I ]

Es können sieben Zahlenwerte angegeben werden:

1. Zahl: Kopftext <3>

Anzahl der Zeilen für den Kopftext, einschließlich der Leerzeilen

2. Zahl: Höhe <60>

Anzahl der Zeilen je Reihe (Formular), ohne die Zeilen für den Kopf- und Fußtext

3. Zahl: Fußtext <0>

Anzahl der Zeilen für den Fußtext, einschließlich der Leerzeilen

4. Zahl: Reihen <1>

Anzahl der Reihen pro Seite (Formulare pro Spalte). Jede Reihe besteht aus sovielen Zeilen, wie mit der 2. Zahl angegeben wird.

5. Zahl: Wiederholung des Fußtextes

Anzahl der Zeilen, die vom Fußtext der Seite (von der ersten nach unten gezählt) auch nach jeder Formularreihe (außer nach der untersten Reihe, nach der alle Zeilen des Fußtextes gedruckt werden) wiederholt werden sollen

6. Zahl: Leerzeilen <0>

Anzahl der Leerzeilen zwischen den einzelnen Formularreihen

7. Zahl: Wiederholung des Kopftextes <0>

Anzahl der Zeilen, die vom Kopftext der Seite (von der letzten nach oben gezählt) auch vor jeder Formularreihe (außer vor der obersten Reihe, vor der alle Zeilen des Kopftextes gedruckt werden) wiederholt werden sollen

**KT**

Textteile, die als Kopftext oben auf jeder Seite gedruckt werden sollen.

[ II ] <: &Q3 @/ &D2 &U2 &#6:: &Q0:>

In den Textteilen werden folgende Steueranweisungen durch die entsprechenden aktuellen Werte ersetzt:

&Q0 Segmentname (von Parameter BER)

&Q1 Projektname der QUELL-Datei (ohne Dateiname)

&Q2 Dateiname der QUELL-Datei (ohne Projektname)

&Q3 Projekt- und Dateiname der QUELL-Datei

&D0 Wochentag (z. B. Sonntag)

&D1 Datum xx.xx.xx (z. B. 02.04.96)

&D2 Datum xx. xxx. xxxx (z. B. 2. Apr. 2008)

&D3 Datum xx. xxxxxxxxxxxx xxxx (z. B. 2. April 2008)

&U1 Uhrzeit xx.xx (z. B. 12.00)

&U2 Uhrzeit xx:xx (z. B. 12:00)



&#n Seitennummer mit maximal n (n=1 bis 6) Stellen

Die Seitennummer kann nur einmal eingesetzt werden. Wird für die Seitennummer »- &#n -« (n=1 bis 6) angegeben, so wird die Seitennummer in die Mitte zwischen die Minuszeichen eingesetzt; die Minuszeichen werden bis auf ein Leerzeichen als Zwischenraum nach rechts bzw. links zur Seitennummer hin verschoben.

Werden pro Seite mehrere Reihen (4. Zahlwert des Parameters DRZ) ausgegeben und wird &#n in einer Zeile angegeben, die für jede Reihe wiederholt wird (4. bzw. 7. Zahlwert des Parameters DRZ), so wird statt der Seitennummer die laufende Nummer der jeweiligen Reihe eingesetzt.

Jeder der Textteile kann durch die Formatieranweisungen »@z« und »@/« in drei Teile gegliedert sein:

linksbündig @z auf Mitte zentriert @/ rechtsbündig

Diese einzelnen Teile werden linksbündig, auf Mitte zentriert und rechtsbündig eingesetzt. Jeder einzelne Teil kann (bei Teil zwei und drei einschließlich der davor stehenden Formatieranweisung) fehlen.

Jeder Textteil wird in eine neue Zeile des Kopftextes gedruckt. Bei mehrspaltigem Druck können auch Textteile für die einzelnen Spalten angegeben werden. Dazu gibt es folgende Regelung:

Beginnt ein Textteil mit »\* :«, so wird der Rest des Textteils über jede Spalte in den Kopftext eingetragen. Steht anstelle des Sterns eine Zahl, so wird der Rest des Textteils über die durch die Zahl bezeichnete Spalte eingetragen. Hat die Zahl den Wert 0, so gilt der Rest des Textteils für die ganze Zeile. Beginnt ein Textteil nicht in der beschriebenen Weise, so wird »0 :« angenommen (Normalfall).

Mit einem Textteil, der für eine ganze Zeile des Kopftextes gilt, wird immer eine neue Zeile begonnen. Ein Textteil, der über einer bestimmten Spalte stehen soll, wird in die gleiche Zeile wie der vorangehende Textteil eingetragen, falls diese Zeile nicht schon einen Text für die ganze Zeile oder für diese oder eine weiter rechts stehende Spalte enthält; andernfalls wird mit diesem Textteil eine neue Zeile begonnen.

**FT** Textteile, die als Fußtext unten auf jeder Seite gedruckt werden sollen. [ II ] <>

Es gelten die gleichen Regelungen wie für den Kopftext (Parameter KT). Die Seitennummer kann jedoch nicht in den Fußtext eingesetzt werden, wenn sie schon in den Kopftext eingesetzt worden ist.

**SM** Angabe, nach wievielen Seiten jeweils wieder eine Seite mit Schneidemarken gedruckt werden soll. [ I ] <100>

**DRT** Druckertyp, für den die Daten aufbereitet werden. [ XI ]

Dieser Parameter ist obligat.

Hinweis: Mit dem Kommando #LISTE, DRUCKERTYPEN werden die definierten Druckertypen aufgelistet.

## Angabe zur Nummerierung

**NR** Erste Seitennummer für die Ausgabe. [ I ] <1>

## Zusammenfassen mehrerer Sätze zu einer Texteinheit

Falls jeder Eingabesatz schon eine vollständige Texteinheit (das sind alle Daten für ein Formular einschließlich der eventuell notwendigen Fortsetzungsformulare) enthält, sollte keiner der folgenden Parameter angegeben werden. Andernfalls können mit den Parametern ANR, ALZ, AA und/oder AE jeweils mehrere Sätze zu einer Texteinheit zusammengefasst werden.

Ist einer der Parameter ANR, ALZ, AA und AE angegeben ist, werden vor der Auswertung dieser Parameter evtl. am Anfang und am Ende des Eingabesatzes stehende Leerzeichen entfernt.

Beim Zusammenfassen wird zwischen den einzelnen Eingabesätzen je ein Leerzeichen ergänzt, jedoch nicht an den Stellen, an denen eine Silbentrennung aufgehoben wird (siehe Parameter STR).

**ANR** Angaben, ob aufeinander folgende Sätze, deren Satznummern teilweise oder vollständig übereinstimmen, zu einer Texteinheit zusammengefasst werden sollen. (Abschnitt auf Grund der Nummerierung). [ I ] <0>

Es kann ein Zahlenwert angegeben werden:

- 0 = kein Zusammenfassen von Sätzen auf Grund der Satznummer
- 1 = alle aufeinander folgenden Sätze mit der gleichen Seitennummer zu einer Texteinheit zusammenfassen.
- 2 = alle aufeinander folgenden Sätze mit der gleichen Seiten- und der gleichen Zeilennummer (ohne Berücksichtigung der Unterscheidungsnummer) zu einer Texteinheit zusammenfassen.
- 3 = alle aufeinander folgenden Sätze mit der gleichen Satznummer zu einer Texteinheit zusammenfassen.

Ist 0 angegeben (Voreinstellung), so wird die Zusammenfassung nur auf Grund der Parameter ALZ, AA und AE vorgenommen; ist einer der Werte 1 bis 3 angegeben, so ist eine weitere Unterteilung der so gebildeten Texteinheiten auf Grund der genannten Parameter möglich.

**ALZ** Angaben, ob Leerzeilen als Kennzeichen für den Anfang bzw. für das Ende einer Texteinheit dienen sollen. [ I ]

Es können zwei Zahlenwerte angegeben werden:

- 1. Zahl: Anfang einer Texteinheit <0>

- 0 = Leerzeilen sind kein Kennzeichen für den Anfang einer Texteinheit.
- 1 = Leerzeilen kennzeichnen den Anfang einer Texteinheit. Folgen mehrere Leerzeilen unmittelbar aufeinander, so beginnt mit jeder einzelnen Leerzeile eine Texteinheit.

## 2. Zahl: Ende einer Texteinheit <0>

- 0 = Leerzeilen sind kein Kennzeichen für das Ende einer Texteinheit.
- 1 = Leerzeilen kennzeichnen das Ende einer Texteinheit. Folgen mehrere Leerzeilen unmittelbar aufeinander, so endet mit jeder einzelnen Leerzeile eine Texteinheit.

Ist jeweils 0 angegeben (Voreinstellung), so wird die Zusammenfassung nur auf Grund der Parameter ANR, AA und AE vorgenommen; andernfalls ist eine weitere Unterteilung der so gebildeten Texteinheiten auf Grund der genannten Parameter möglich.

**AA** Zeichenfolgen, die am Anfang des Eingabesatzes den Anfang einer Texteinheit (Abschnittsanfang) kennzeichnen. [ VIII-a ]

Falls mit dem Parameter AAZ nichts anderes angegeben ist, werden vor der Überprüfung, ob ein Satz mit einer der angegebenen Zeichenfolgen beginnt, führende Leerzeichen entfernt.

Beginnen mit dem Parameter AA angegebene Zeichenfolgen mit Leerzeichen, so können diese nur dann einen Abschnittsanfang kennzeichnen, wenn mit dem Parameter AAZ angegeben wird, dass die führenden Leerzeichen erst nach der Überprüfung entfernt werden sollen; andernfalls ist die Angabe solcher Zeichenfolgen wirkungslos.

**AAZ** Zusatzangaben zum Parameter AA. [ 1 ] <0>

Es kann ein Zahlenwert angegeben werden:

- 0 = Leerzeichen am Anfang von Eingabesätzen vor der Auswertung des Parameters AA entfernen.
- 1 = Leerzeichen am Anfang von Eingabesätzen nach der Auswertung des Parameters AA entfernen.
- 2 = wie 0, jedoch zusätzlich auch die Zeichenfolgen »#[09]« und »#[0009]« entfernen.

**AE** Zeichenfolgen, die am Ende des Eingabesatzes das Ende einer Texteinheit (Abschnittsende) kennzeichnen. [ VIII-b ]

Falls mit dem Parameter AEZ nichts anderes angegeben ist, werden vor der Überprüfung, ob ein Satz mit einer der angegebenen Zeichenfolgen endet, abschließende Leerzeichen entfernt.

Enden mit dem Parameter AE angegebene Zeichenfolgen mit Leerzeichen, so können diese nur dann ein Abschnittsende kennzeichnen, wenn mit dem Parameter AEZ angegeben wird, dass die abschließenden Leerzeichen erst nach der Überprüfung entfernt werden sollen; andernfalls ist die Angabe solcher Zeichenfolgen wirkungslos.

**AEZ**

Zusatzangaben zum Parameter AE. [ I ] <0>

Es kann ein Zahlenwert angegeben werden:

- 0 = Leerzeichen am Ende von Eingabesätzen vor der Auswertung des Parameters AE entfernen.
- 1 = Leerzeichen am Ende von Eingabesätzen nach der Auswertung des Parameters AE entfernen.

**STR**

Angaben zur Silbentrennung. [ I ]

Es können zwei Zahlenwerte angegeben werden:

1. Zahl: Silbentrennung in den Eingabedaten <0>

- 0 = Silbentrennungen nicht aufheben.
- 1 = Silbentrennung bei der Eingabe aufheben; dabei wird angenommen, dass im nachfolgenden Eingabesatz alle Zeichen bis zum ersten Leerzeichen zum getrennten Wort gehören.
- 2 = wie 1, jedoch Silbentrennung bei der Eingabe nur aufheben, falls entweder der letzte Buchstabe vor und der erste Buchstabe nach dem Silbentrennzeichen (im nachfolgenden Eingabesatz) Kleinbuchstaben sind oder der letzte Buchstabe vor und die ersten beiden Buchstaben nach dem Silbentrennzeichen (im nachfolgenden Eingabesatz) Großbuchstaben sind.
- 3 = wie 2, jedoch wird angenommen, dass im nachfolgenden Eingabesatz alle Zeichen bis zum ersten Leerzeichen bzw. bis zur ersten öffnenden spitzen Klammer zum getrennten Wort gehören.
- 4 = wie 2, jedoch wird angenommen, dass im nachfolgenden Eingabesatz alle Zeichen bis zum ersten Leerzeichen, das außerhalb von spitzen Klammern steht, zum getrennten Wort gehören.

2. Zahl: Silbentrennung beim Formatieren <1>

- 0 = Nach den alten Silbentrennregeln trennen
- 1 = Nach den neuen Silbentrennregeln von 1996 trennen

Falls angegeben ist, dass die Silbentrennung bei der Eingabe aufgehoben werden soll, werden in allen Eingabesätzen führende und abschließende Leerzeichen entfernt.

Sollen bei der Eingabe Zeichenfolgen ersetzt werden (Parameter x), so wird nach dem Ersetzen festgestellt, ob der Satz mit einem Silbentrennzeichen endet.

In den Eingabedaten gilt als Silbentrennzeichen ein »-«, das (nachdem abschließende Leerzeichen entfernt wurden) als letztes Zeichen in einem Eingabesatz steht, wenn das zweitletzte Zeichen ebenfalls ein »-« ist, oder wenn das zweitletzte Zeichen ein Buchstabe und das drittletzte Zeichen kein Steuerzeichen (\$, &, @, \, \_, #, %) ist.

Beim Zusammenfassen mehrerer Eingabesätze zu einer Texteinheit werden Silbentrennungen nur innerhalb einer Texteinheit aufgehen.

ben. Steht ein Silbentrennzeichen am Ende des letzten Eingabesatzes einer Texteinheit, wird an dieser Stelle die Silbentrennung nicht aufgehoben.

Beim Aufheben der Silbentrennung wird ein getrenntes »ck«, das als »k-« und »k« geschrieben ist, nicht wieder zu »ck«.

## Ersetzen von Zeichenfolgen

**XX** Zeichenfolgenpaare (und Ausnahmezeichenfolgen), deren jeweils erste Zeichenfolge durch die jeweils zweite Zeichenfolge in der Texteinheit ersetzt werden soll. [ X ]

## Ergänzen eines Textteils

**ERG** Textteil, der vor jeder Texteinheit ergänzt werden soll. [ II ]

Hiermit können konstante Texte für die Formulare vorgegeben werden. Durch die darin enthaltenen Kennzeichen kann gesteuert werden, an welche Stelle sie im Formular gedruckt werden sollen.

## Aufteilen der Texteinheit in Formularfelder

**TTK** Zeichenfolgen, die die einzelnen Textteile kennzeichnen. Die Textteile werden in der Reihenfolge in das Formular eingespeichert, in der die Kennzeichen mit diesem Parameter angegeben sind. Falls ein Kennzeichen in den Daten mehr als einmal vorkommt, werden die damit gekennzeichneten Textteile zu einem Textteil zusammengefasst. Die Kennzeichen werden nicht ins Formularfeld übernommen. [ IX ]

**TTF** Angabe (parallel zu TTK), ob ein Textteil in ein eigenes Formularfeld eingespeichert oder an einen anderen Textteil angehängt werden soll. [ I ] <0>

0 = Textteil soll an keinen anderen angehängt werden

n = Textteil soll an den n-ten Textteil (das ist der Textteil, der mit der n-ten Zeichenfolge des Parameters TTK gekennzeichnet ist) angehängt werden. Dieser kann seinerseits an einen anderen Textteil angehängt sein.

Das Abarbeiten erfolgt von links nach rechts (in der Reihenfolge, in der auch die Kennzeichen mit dem Parameter TTK angegeben sind).

**EGA / EGE** Textteile (parallel zu TTK), die am Anfang bzw. am Ende des jeweiligen Textteils ergänzt werden sollen. Hiermit können in Abhängigkeit vom Vorkommen bestimmter Textteile (Kennzeichen) konstante Texte ergänzt werden. [ II ]

Die so eingefügten Textteile werden nicht daraufhin überprüft, ob sie Kennzeichen enthalten, die mit dem Parameter TTK definiert sind.

Sind solche Kennzeichen darin enthalten, werden sie wie Text behandelt.

**AZN**    **n** Anfangszeilennummern (parallel zu TTK) der einzelnen Formularfelder. [ 1 ] <0>

0 = Textteil ignorieren

n = Textteil ab der n-ten Zeile einspeichern. Ist die n-te Zeile schon belegt, darf in die nächste freie Zeile eingespeichert werden, falls der dazugehörige Wert des Parameters EZN dies zulässt.

Für Textteile, die auf Grund einer entsprechenden Angabe mit dem Parameter TTF an einen anderen Textteil angehängt werden, kann bei diesem Parameter der Wert 0 angegeben werden; diese 0 dient in diesem Fall nur als Platzhalter, damit die weiteren Zahlenwerte den zum Parameter TTK angegebenen Zeichenfolgen richtig zugeordnet werden.

**EZN**    **n** Endzeilennummern (parallel zu AZN) der einzelnen Formularfelder. [ 1 ] <0>

0 = Textteil darf nur in einer (nämlich der mit dem Parameter AZN angegebenen) Zeile eingespeichert werden.

n = Textteil darf ab der mit dem Parameter AZN angegebenen Zeile bis einschließlich der n-ten Zeile eingespeichert werden.

**DPA**    **n** Druckpositionanfang (parallel zu AZN) der einzelnen Formularfelder. [ 1 ] <1>

Damit wird die erste Position bestimmt, ab der der Textteil in den einzelnen (mit AZN und EZN angegebenen) Zeilen eingespeichert werden soll.

**DPE**    **n** Druckpositionende (parallel zu AZN) der einzelnen Formularfelder. <3. Zahlenwert des Parameters DR>

Damit wird die letzte Position bestimmt, bis zu der der Textteil in den einzelnen (mit AZN und EZN angegebenen) Zeilen eingespeichert werden soll.

**ZEN**    **n** Angaben (parallel zu AZN) zum Zentrieren des Textes in den einzelnen Formularfeldern. [ 1 ] <0>

0 = linksbündig

1 = auf Mitte zentrieren

2 = rechtsbündig

3 = mit Randausgleich

Soll auch in vertikaler Richtung zentriert werden, so ist auf diese Werte zu addieren:

0 = nach oben

10 = auf Mitte

20 = nach unten

- ZW** Zeichenfolgen, bei denen eine neue Zeile begonnen werden soll. Die jeweilige Zeichenfolge wird nicht ausgedruckt. [ IX ]
- ZA / ZE** Die Textteile sollen so in Zeilen aufgeteilt werden, dass Zeichenfolgen, die unter ZA bzw. ZE angegeben sind, am Anfang bzw. am Ende einer Zeile stehen. [ IX ]
- ZAB / ZEB** Falls ein Textteil in mehrere Zeilen aufgeteilt werden muss (ggf. zusätzlich zur Aufteilung auf Grund der Parameter ZW, ZA, ZE), so soll vorzugsweise so unterteilt werden, dass Zeichenfolgen, die unter ZAB bzw. ZEB angegeben sind, am Anfang bzw. am Ende einer Zeile stehen. [ IX ]
- TTS** Angabe (parallel zu TTK), unter welchen Umständen Silbentrennung versucht werden soll, falls der entsprechende Textteil (ggf. zusätzlich zur Aufteilung auf Grund der Parameter ZW, ZA, ZE, ZAB, ZEB) in mehrere Zeilen aufgeteilt werden muss. [ I ] <2>
- 0 = nur an Stellen trennen, die durch \ gekennzeichnet sind  
 1 = Silbentrennung außer an Stellen, die mit \ gekennzeichnet sind nur, wenn zum Randausgleich irgendwo mehr als 1 Leerzeichen eingeschoben werden müsste  
 n = Silbentrennung außer an Stellen, die mit \ gekennzeichnet sind, nur, falls noch mehr als n Zeichen in der Zeile frei sind
- Weitere Angaben zur Silbentrennung mit Parameter STR.
- TAB** Zeichenfolgen, bei denen zum (logisch) nächsten Tabulator gesprungen werden soll. Die jeweilige Zeichenfolge wird nicht ausgedruckt. Falls die Tabulatorposition bereits überschritten wurde, wird kein Sprung ausgeführt. [ IX ]
- TNR** Angabe (parallel zu TAB), falls beim Vorkommen einer Zeichenfolge, die mit dem Parameter TAB angegeben ist, nicht nur bis zur nächsten Tabulatorposition gesprungen werden soll, sondern mindestens bis zu einer bestimmten (nämlich der Position des hier angegebenen Tabulators). Falls die Nummer des (logisch) nächsten Tabulators größer ist als die hier angegebene Nummer, wird zu diesem Tabulator gesprungen. [ I ]
- TPO** Tabulatorpositionen, d. h. Anfangspositionen der einzelnen Tabulatorfelder. [ I ]
- TFZ** Angaben (parallel zu TPO) zum Zentrieren des Textes in den einzelnen Tabulatorfeldern. [ I ] <0>
- 0 = linksbündig  
 1 = auf Mitte zentrieren  
 2 = rechtsbündig
- TTV** Angabe (parallel zu TTK), auf wieviele (gleich große) Formularfelder der Textteil verteilt werden soll. Dabei werden die Daten für das zweite und die folgenden Formularfelder wie bei einem Überlauf behandelt, d. h. sie werden in das unter UEF angegebene Formularfeld eingespei-

chert. Diese Folgefelder müssen in diesem Fall jedoch alle gleich groß sein. [ I ]

**UEL**     **n** Angabe (parallel zu TTK), was geschehen soll, falls ein Textteil überläuft, d. h. falls er nicht in das angegebene Formularfeld passt. [ I ] <0>

- 0 = in die letzten 3 Positionen » . . . « einspeichern mit Meldung
- 1 = in die letzten 3 Positionen » . . . « einspeichern ohne Meldung
- 2 = Der Rest soll in ein mit dem Parameter UEF angegebene Formularfeld eingespeichert werden
- 3 = Fortsetzungsformular erzeugen
- 4 = Rest ignorieren mit Meldung
- 5 = Rest ignorieren ohne Meldung

**UEF**     **n** Angabe (parallel zu TTK), in welches Formularfeld (Überlauffeld) der Rest eines Textteils eingespeichert werden soll (siehe Parameter UEL).

n = Nummer des Formularfeldes

Dabei kann n größer sein als die Anzahl der Zeichenfolgen (Kennzeichen), die zum Parameter TTK angegeben sind. In diesem Fall müssen auch zu den Parametern AZN, EZN, DPA, DPE, ZEN und UEL entsprechend viele Angaben gemacht werden.

**UEW**     Angaben (parallel zu TTK), welche Textteile in den Fortsetzungsformularen wiederholt werden sollen. [ I ] <0>

0 = Textteil nicht wiederholen

n = Textteil soll wiederholt werden. Falls mit dem Parameter UTR Trennzeichenfolgen angegeben sind, wird der Textteil nur bis zu der Stelle wiederholt, an der die n-te Trennzeichenfolge auftritt. Die Auswahl durch die Parameter AUL, EUL, (UL und ) UL erfolgt ggf. zuvor.

**AUL / EUL**     Zeichenfolgen, die den Anfang bzw. das Ende des Teils eines Textteils kennzeichnen, der im Fortsetzungsformular wiederholt werden soll. [ IX ]

**(UL / )UL**     Klammern zur Auswahl des Teils im jeweiligen Textteil (falls AUL und/oder EUL nicht angegeben) bzw. zur Eliminierung von Teilen aus dem bereits mit AUL/EUL ausgewählten Teil, der im Fortsetzungsformular wiederholt werden soll. [ IX ]

Die Wirkung der Klammern wird mit dem zweiten Zahlenwert des Parameters ULI bestimmt.

**ULI**     Zusatzangaben zu AUL, EUL und (UL, ) UL. [ I ] <1, 0>

Es können zwei Zahlenwerte angegeben werden:

1. Zahl: Angabe analog zum Parameter AEI für #KOPIERE
2. Zahl: Angabe analog zum Parameter KLI für #KOPIERE



- UTR** Trennzeichenfolgen, bis zu denen ein Textteil auf Fortsetzungsformularen wiederholt werden soll (siehe auch Parameter UEW). [ IX ]
- UTE** Textteil, der den auf Grund des Parameters UTR abgetrennten Text ersetzen soll. [ II ] </ . . . />
- FNR** Angabe, an welche Stelle im Formular eine laufende Nummer der Formulare eingesetzt werden soll. Fortsetzungsformulare werden dabei nicht mitgezählt; diese erhalten die gleiche Nummer wie das dazugehörige Erstformular. [ I ]
- Es können vier Zahlenwerte angegeben werden:
1. Zahl: Nummer der Zeile
  2. Zahl: Erste Zeichenposition
  3. Zahl: Letzte Zeichenposition
  4. Zahl: Nummer des ersten Formulars <1>
- NFF** Angaben, an welche Stelle im Formular eine laufende Nummer eingesetzt werden soll, falls Fortsetzungsformulare generiert werden müssen. Dabei ist zu beachten, dass das angegebene Feld groß genug sein muss, auch die mit dem Parameter NFE angegebenen Zeichenfolgen aufzunehmen. [ I ]
- Es können drei Zahlenwerte angegeben werden:
1. Zahl: Nummer der Zeile
  2. Zahl: Erste Zeichenposition
  3. Zahl: Letzte Zeichenposition
- NFE** Textteile, die am Anfang bzw. am Ende des mit dem Parameter NFF angegebenen Feldes eingesetzt werden sollen, falls Fortsetzungsformulare generiert werden müssen. [ II ] </-/-/>
- MF** Anzahl der Fortsetzungsformulare, die maximal (für eine Texteinheit) generiert werden sollen. [ I ] <2>
- Wenn die angegebene Zahl überschritten wird, wird der Rest der Texteinheit nicht mehr abgearbeitet und eine entsprechende Fehlermeldung ausgegeben.
- LCH** Angabe, an welche Stelle im Formular in keinem Fall etwas gedruckt werden darf (z. B. für ein Loch auf Bibliothekskärtchen). Diese Stelle wird übersprungen (ggf. wird dabei ein Wort wie bei einem Zeilenwechsel getrennt), wenn sie innerhalb eines Formularfeldes liegt, in das ein Textteil gedruckt werden soll. [ I ]
- Es können vier Zahlenwerte angegeben werden:
1. Zahl: Erste Zeile
  2. Zahl: Letzte Zeile
  3. Zahl: Erste Zeichenposition
  4. Zahl: Letzte Zeichenposition

## Alphabetisches Verzeichnis der Parameter

<b>(UL</b>	Ausklammern für Wiederholung bei Überlauf . . . . .	888
<b>)UL</b>	Ausklammern für Wiederholung bei Überlauf . . . . .	888
<b>AA</b>	Anfang einer Texteinheit (Abschnittsanfang) . . . . .	883
<b>AAZ</b>	Zusatzangaben zum Parameter AA . . . . .	883
<b>AE</b>	Ende einer Texteinheit (Abschnittsende) . . . . .	883
<b>AEZ</b>	Zusatzangaben zum Parameter AE . . . . .	884
<b>ALZ</b>	Bilden einer Texteinheit (Abschnitt) bei Leerzeilen . . . . .	882
<b>ANR</b>	Bilden einer Texteinheit (Abschnitt) nach Nummer . . . . .	882
<b>AUL</b>	Anfangskennung für Wiederholung bei Überlauf . . . . .	888
<b>AZN</b>	Anfangszeilennummern der Textteile im Formular . . . . .	886
<b>BER</b>	Auswahl eines Bereichs aus der QUELL-Datei . . . . .	878
<b>DPA</b>	Druckpositionanfang der einzelnen Textteile . . . . .	886
<b>DPE</b>	Druckpositionende der einzelnen Textteile . . . . .	886
<b>DR</b>	Angaben zur Druckausgabesteuerung . . . . .	879
<b>DRT</b>	Druckertyp für die Druckausgabe . . . . .	881
<b>DRZ</b>	Zusätzliche Angaben zur Druckausgabesteuerung . . . . .	880
<b>EGA</b>	Ergänzung am Anfang der einzelnen Textteile . . . . .	885
<b>EGE</b>	Ergänzung am Ende der einzelnen Textteile . . . . .	885
<b>ERG</b>	Ergänzen vor jeder Texteinheit . . . . .	885
<b>EUL</b>	Endekennung für Wiederholung bei Überlauf . . . . .	888
<b>EZN</b>	Endzeilennummern der Textteile im Formular . . . . .	886
<b>FNR</b>	Position für Formularnummer . . . . .	889
<b>FT</b>	Angabe des Fußtextes . . . . .	881
<b>KT</b>	Angabe des Kopftextes . . . . .	880
<b>LCH</b>	Freiraum für Lochung . . . . .	889
<b>MAX</b>	Maximum für Testzwecke . . . . .	879
<b>MFF</b>	Maximale Anzahl von Fortsetzungsformularen . . . . .	889
<b>NFE</b>	Ergänzung für Nummerierung der Fortsetzungsformulare . . . . .	889
<b>NFF</b>	Position für Nummerierung der Fortsetzungsformulare . . . . .	889
<b>NR</b>	Nummer der ersten Seite . . . . .	882
<b>PAR</b>	Parameter-Konvention . . . . .	878
<b>SM</b>	Häufigkeit von Schneidemarken . . . . .	881
<b>STR</b>	Angaben zur Silbentrennung . . . . .	884
<b>TAB</b>	Zeichenfolgen, die als Tabulator dienen . . . . .	887
<b>TFZ</b>	Zentrieren von Tabulatorfeldern . . . . .	887
<b>TNR</b>	Tabulatornummer . . . . .	887
<b>TPO</b>	Tabulatorpositionen . . . . .	887
<b>TTF</b>	Fortsetzung von Textteilen . . . . .	885
<b>TTK</b>	Kennzeichen für Textteile . . . . .	885
<b>TTS</b>	Silbentrennung innerhalb von Textteilen . . . . .	887
<b>TTV</b>	Verteilen eines Textteils auf mehrere Formularfelder . . . . .	887
<b>UEF</b>	Fortsetzung eines Formularfeldes bei Überlauf . . . . .	888
<b>UEL</b>	Regelung für überlaufende Formularfelder . . . . .	888
<b>UEW</b>	Wiederholen von Textteilen bei Überlauf . . . . .	888
<b>ULI</b>	Index für AUL, EUL und (UL, )UL . . . . .	888
<b>UTE</b>	Ersatz für den mit UTR abgetrennten Text . . . . .	889

---

<b>UTR</b>	Trennzeichen für Text-Wiederholung bei Überlauf . . . . .	889
<b>XX</b>	Ersetzen von Zeichenfolgen in der Texteinheit . . . . .	885
<b>ZA</b>	Zeilenanfang innerhalb von Textteilen . . . . .	887
<b>ZAB</b>	Zeilenanfang bei Bedarf innerhalb von Textteilen . . . . .	887
<b>ZE</b>	Zeilenende innerhalb von Textteilen . . . . .	887
<b>ZEB</b>	Zeilenende bei Bedarf innerhalb von Texteinheiten . . . . .	887
<b>ZEN</b>	Zentrieren der einzelnen Textteile . . . . .	886
<b>ZW</b>	Zeilenwechsel innerhalb von Textteilen . . . . .	887

\*\*\*\*\*



**#FORMATIERE**

FORMATIERE

---

Kommando . . . . .	895
Leistung . . . . .	896
Hinweise . . . . .	896
Beispiele . . . . .	896
Parameter . . . . .	898
Einstellen der Parameter-Konvention . . . . .	898
Angaben zur Silbentrennung . . . . .	898
Auswahl der Daten . . . . .	899
Angaben zum Protokoll . . . . .	899
Angabe zur Nummerierung . . . . .	902
Ersetzen von Zeichenfolgen . . . . .	902
Makro-Definitionen . . . . .	903
Angaben zum Randausgleich . . . . .	903
Umdrehen von Textteilen . . . . .	903
Alphabetisches Verzeichnis der Parameter . . . . .	905
Makros . . . . .	906
Steueranweisungen . . . . .	907
Wortzwischenraum . . . . .	907
Blocksatz . . . . .	907
Abschnitte, Zeilen- und Seitenumbruch, Leerzeilen . . . . .	908
Zeilenabstand . . . . .	909
Einrücken . . . . .	910
Zentrieren . . . . .	910
Positionieren . . . . .	911
Tabulatoren . . . . .	911
Ränder . . . . .	912
Fußnoten . . . . .	912
Datum und Uhrzeit einfügen . . . . .	913
Auffüllen, Wiederholen . . . . .	913
Druck unterdrücken . . . . .	913
Text und Steueranweisungen ignorieren . . . . .	913
Silbentrennung . . . . .	914
Markierungen . . . . .	915
Formate . . . . .	915
Zeichenvorrat . . . . .	916

## Kommando

### #FORMATIERE

QUELLE	= <code>datei</code>	Name der Datei mit den Daten (mit Steueranweisungen), die formatiert werden sollen.
	= <code>-STD-</code>	Die Daten (mit Steueranweisungen), die formatiert werden sollen, stehen in der Standard-Text-Datei.
ZIEL	= <code>-</code>	* Nur Protokoll-Ausgabe.
	= <code>datei</code>	Name der Datei für die formatierten Daten (mit Steueranweisungen) in der neuen Seiten-Zeilen-Einteilung.
	= <code>-STD-</code>	Die formatierten Daten (mit Steueranweisungen) in der neuen Seiten-Zeilen-Einteilung in die Standard-Text-Datei ausgegeben.
MODUS	= <code>...</code>	Druckertyp, für den die Daten aufbereitet werden sollen. Er kann auch über Parameter angegeben werden.  Mit dem Kommando #LISTE, DRUCKERTYPEN können die definierten Druckertypen aufgelistet werden.
LOESCHEN	= <code>-</code>	* Daten in der ZIEL- und in der PROTOKOLL-Datei nicht löschen.
	= <code>+</code>	Daten in der ZIEL- und in der PROTOKOLL-Datei zuerst löschen.
PARAMETER	= <code>-</code>	* Keine Parameter.
	= <code>datei</code>	Name der Datei mit den Parametern.
	= <code>*</code>	Die Parameter folgen auf das Kommando und sind mit *EOF abgeschlossen.
VORSPANN	= <code>-</code>	* Kein Vorspann.
	= <code>datei</code>	Name der Datei mit dem Vorspann (Steueranweisungen für eigene Voreinstellungen).
	= <code>*</code>	Der Vorspann (Steueranweisungen für eigene Voreinstellungen) folgt auf das Kommando (ggf. nach den Parametern) und ist mit *EOF abgeschlossen.
PROTOKOLL	= <code>-STD-</code>	* Das Protokoll mit den formatierten Daten in die Standard-Protokoll-Datei ausgeben.
	= <code>datei</code>	Name der Datei für das Protokoll mit den formatierten Daten.
	= <code>-</code>	Keine Protokoll-Ausgabe.

## Leistung

Mit diesem Programm können Texte zum Drucken aufbereitet werden. Die Aufteilung des Textes auf Zeilen und Seiten (einschließlich Silbentrennung, Randausgleich und Fußnoten) wird automatisch vorgenommen und kann über Anweisungen, die im Text enthalten sind, gesteuert werden.

## Hinweise

Die Daten in der QUELL-Datei bleiben unverändert. Das Ergebnis des Programms (die zum Drucken aufbereiteten Daten) wird in die PROTOKOLL-Datei ausgegeben. Diese kann anschließend mit dem Kommando #DRUCKE ausgedruckt (auf einen Drucker ausgegeben) werden.

In die ZIEL-Datei werden die Daten so ausgegeben, wie sie in der QUELL-Datei stehen (mit Ausnahme der Zeichenfolgen, die durch entsprechende Parameterangaben ausgetauscht werden). Dabei wird jedoch die Zeileneinteilung und die Seiten-Zeilenummer dem formatierten Ergebnis angeglichen.

Die ZIEL-Datei kann z. B. als neue Korrekturgrundlage dienen und hat dabei gegenüber der QUELL-Datei den Vorteil, dass im Editor die im ausgedruckten Ergebnis stehenden Seitenzahlen zum Auffinden der zu korrigierenden Textstellen verwendet werden können.

Soll vom formatierten Text ein Register erstellt werden, so muss als Grundlage des Registers die ZIEL-Datei verwendet werden, damit sichergestellt ist, dass die als Referenz verwendeten Seitenzahlen dem aktuellen Stand entsprechen.

Sollen die in den Daten enthaltenen Steuerzeichen nicht interpretiert, sondern mit ausgedruckt werden, so müssen die Daten (statt mit dem Kommando #FORMATIERE) mit dem Kommando #DVORBEREITE zum Drucken aufbereitet werden.

## Beispiele

Formatieren und Ausdrucken der Datei `xy` auf einem HP-LaserJet, der den vom System vorgegebenen Druckernamen `pr1` hat:

```
#FO,xy,,HPLJ,+  
#DR,,HPLJ,pr1
```

Ausdrucken der Datei `alt` auf einem PostScript-Drucker, der das Papier beidseitig bedrucken kann und den vom System vorgegebenen Druckernamen `ps2` hat. Die Daten enthalten Makros, die mit spitzen Klammern gekennzeichnet sind. Als Seitenüberschrift soll nur die Seitenzahl ausgegeben werden. Der Ausdruck der einzelnen Seiten soll in der Reihenfolge erfolgen, dass diese nach Falten des Papierstapels zu einem Heftchen in der richtigen Reihenfolge stehen:



```

#FO,alt,neu,,+,*
DRT      PS-Q2
MKZ      2
          Haupt-Überschrift
MKR      "1"&?0 #2+ &,"
MKR      "/1"&, #2- $ &!3"
          Zwischen-Überschrift
MKR      "2"&?8 #1+ &,"
MKR      "/2"&, #1- $ &!1"
          Absatz
MKR      "p"&!0"
...
KT       " @z - &#2 - "
*EOF

#DR,,PS-Q2,ps2,SEITEN=HEFT,OPTIONEN=D_KURZ

```

## Parameter

Jeder Parameter ist einer Parameterart zugeordnet. Sie ist jeweils als römische Zahl in [ ] angegeben. Damit ist festgelegt, in welcher Form die Angaben vom Programm erwartet werden. Die einzelnen Parameterarten sind in der Beschreibung »TUSTEP-Grundlagen« im Kapitel »{}-Parameter« bzw. »<>-Parameter« definiert.

Für manche Parameter gibt es voreingestellte Werte, die angenommen werden, falls die entsprechenden Werte nicht angegeben sind. Diese Voreinstellungen sind in < > angegeben.

Außer den im folgenden beschriebenen Parametern sind auch Parameter zur Definition von Zeichen- und Stringgruppen möglich. [ V ]

### Einstellen der Parameter-Konvention

Ob die Parameter nach der »{}-Parameter-Konvention« oder nach der »<>-Parameter-Konvention« interpretiert werden, kann mit dem Kommando #PARAMETER, {} bzw. #PARAMETER, <> eingestellt werden. Darüber hinaus kann die Konvention auch mit folgendem Parameter eingestellt werden.

- PAR** Parameter-Konvention einstellen.
- { } Nachfolgende Parameter nach der »{}-Parameter-Konvention« interpretieren.
  - <> Nachfolgende Parameter nach der »<>-Parameter-Konvention« interpretieren.

Die Einstellung mit dem Parameter PAR hat Vorrang vor der mit dem Kommando #PARAMETER gewählten Einstellung; sie gilt jeweils nur für die nachfolgenden Parameter bis zum nächsten Parameter PAR bzw. bis zum Ende der Parameter dieses Programms.

### Angaben zur Silbentrennung

Beim Formatieren werden bei Bedarf Silbentrennungen automatisch durchgeführt; mit Steueranweisungen, die im Vorspann oder in den Daten stehen, kann sie eingeschränkt bzw. aus- und eingeschaltet werden.

- STR** Angaben zur Silbentrennung. [ I ]
- Es können zwei Zahlenwerte angegeben werden:
1. Zahl: Silbentrennung in den Eingabedaten <1>
    - 0 = Eingabedaten enthalten keine Silbentrennung
    - 1 = Silbentrennung bei der Eingabe aufheben
  2. Zahl: Silbentrennung beim Formatieren <1>

- 0 = Nach den alten Silbentrennregeln trennen
- 1 = Nach den neuen Silbentrennregeln von 1996 trennen

Sollen bei der Eingabe Zeichenfolgen ersetzt werden (Parameter X), so wird nach dem Ersetzen festgestellt, ob der Satz mit einem Silbentrennzeichen beginnt.

In den Eingabedaten gilt als Silbentrennzeichen ein »-«, das (nachdem abschließende Leerzeichen entfernt wurden) als letztes Zeichen in einem Eingabesatz steht, wenn das zweitletzte Zeichen ebenfalls ein »-« ist, oder wenn das zweitletzte Zeichen ein Buchstabe und das drittletzte Zeichen kein Steuerzeichen (\$, &, @, \, \_, #, %) ist.

Beim Aufheben der Silbentrennung wird ein getrenntes »ck«, das als »k-« und »k« geschrieben ist, nicht wieder zu »ck«.

## Auswahl der Daten

Soll die gesamte Datei verarbeitet werden, braucht keiner der folgenden Parameter angegeben zu werden.

**BER** Angabe eines Bereichs (»Seite.Zeile–Seite.Zeile«, falls die Datei im Textmodus nummeriert ist; Zeile–Zeile, falls die Datei im Programmmodus nummeriert ist) oder einer Anfangsstelle (»Seite.Zeile« bzw. »Zeile«), falls nicht die ganze Datei verarbeitet werden soll. [ XI ]

Soll ein Segment einer Segment-Datei verarbeitet werden, kann anstelle des Bereichs der Name des Segments angegeben werden.

Dieser Parameter ist nur zugelassen, wenn die Satznummern in der Datei alle aufsteigend sind.

**MAX** Angaben für Testzwecke, wieviele Sätze maximal eingelesen bzw. ausgegeben werden sollen bzw. wieviele Seiten maximal aufbereitet werden sollen. [ I ]

Es können drei Zahlenwerte angegeben werden:

1. Zahl: Einzulesende Sätze <999999999>
2. Zahl: Auszugebende Sätze <999999999>
3. Zahl: Auszugebende Seiten <999999999>

## Angaben zum Protokoll

**DR** Angaben zur Druckausgabesteuerung. [ I ]

Es können vier Zahlenwerte angegeben werden:

1. Zahl: Spalten <1>

Anzahl der Spalten, die auf jeder Seite nebeneinander gedruckt werden sollen

2. Zahl: Rand <10>

Anzahl der Leerstellen links der ersten Spalte

3. Zahl: Breite <64>

Anzahl der Zeichen je Spalte

4. Zahl: Zwischenraum <0>

Anzahl der Leerstellen zwischen den Spalten

**DRZ**

Zusätzliche Angaben zur Druckausgabesteuerung. [ I ]

Es können sieben Zahlenwerte angegeben werden:

1. Zahl: Kopftext <3>

Anzahl der Zeilen für den Kopftext, einschließlich der Leerzeilen

2. Zahl: Höhe <60>

Anzahl der Zeilen je Reihe, ohne die Zeilen für den Kopf- und Fußtext

3. Zahl: Fußtext <0>

Anzahl der Zeilen für den Fußtext, einschließlich der Leerzeilen

4. Zahl: Reihen <1>

Anzahl der Reihen pro Seite. Jede Reihe besteht aus sovielen Zeilen, wie mit der 2. Zahl angegeben wird.

5. Zahl: Wiederholung des Fußtextes <0>

Anzahl der Zeilen, die vom Fußtext der Seite (von der ersten Zeile nach unten gezählt) auch nach jeder Reihe (außer nach der untersten Reihe, nach der alle Zeilen des Fußtextes gedruckt werden) wiederholt werden sollen

6. Zahl: Leerzeilen <0>

Anzahl der Leerzeilen zwischen den einzelnen Reihen

7. Zahl: Wiederholung des Kopftextes <0>

Anzahl der Zeilen, die vom Kopftext der Seite (von der letzten Zeile nach oben gezählt) auch vor jeder Reihe (außer vor der obersten Reihe, vor der alle Zeilen des Kopftextes gedruckt werden) wiederholt werden sollen

**KT**

Textteile, die als Kopftext oben auf jeder Seite gedruckt werden sollen.

[ II ] <:&Q3 @/ &D2 &U2 &#6::&Q0:>

In den Textteilen werden folgende Steueranweisungen durch die entsprechenden aktuellen Werte ersetzt:

&Q0 Segmentname (von Parameter BER)

&Q1 Projektname der QUELL-Datei (ohne Dateiname)

&Q2 Dateiname der QUELL-Datei (ohne Projektname)

&Q3 Projekt- und Dateiname der QUELL-Datei

- &D0 Wochentag (z. B. Sonntag)
- &D1 Datum xx.xx.xx (z. B. 02.04.96)
- &D2 Datum xx. xxx. xxxx (z. B. 2. Apr. 2008)
- &D3 Datum xx. xxxxxxxxxxxx xxxx (z. B. 2. April 2008)
- &U1 Uhrzeit xx.xx (z. B. 12.00)
- &U2 Uhrzeit xx:xx (z. B. 12:00)
- &#n Seitennummer mit maximal n (n=1 bis 6) Stellen

Die Seitennummer kann nur einmal eingesetzt werden. Wird für die Seitennummer »- &#n -« (n=1 bis 6) angegeben, so wird die Seitennummer in die Mitte zwischen die Minuszeichen eingesetzt; die Minuszeichen werden bis auf ein Leerzeichen als Zwischenraum nach rechts bzw. links zur Seitennummer hin verschoben.

Werden pro Seite mehrere Reihen (4. Zahlwert des Parameters DRZ) ausgegeben und wird &#n in einer Zeile angegeben, die für jede Reihe wiederholt wird (4. bzw. 7. Zahlwert des Parameters DRZ), so wird statt der Seitennummer die laufende Nummer der jeweiligen Reihe eingesetzt.

Jeder der Textteile kann durch die Formatieranweisungen »@z« und »@/« in drei Teile gegliedert sein:

linksbündig @z auf Mitte zentriert @/ rechtsbündig

Diese einzelnen Teile werden linksbündig, auf Mitte zentriert und rechtsbündig eingesetzt. Jeder einzelne Teil kann (bei Teil zwei und drei einschließlich der davor stehenden Formatieranweisung) fehlen.

Jeder Textteil wird in eine neue Zeile des Kopftextes gedruckt. Bei mehrspaltigem Druck können auch Textteile für die einzelnen Spalten angegeben werden. Dazu gibt es folgende Regelung:

Beginnt ein Textteil mit »\* :«, so wird der Rest des Textteils über jede Spalte in den Kopftext eingetragen. Steht anstelle des Sterns eine Zahl, so wird der Rest des Textteils über die durch die Zahl bezeichnete Spalte eingetragen. Hat die Zahl den Wert 0, so gilt der Rest des Textteils für die ganze Zeile. Beginnt ein Textteil nicht in der beschriebenen Weise, so wird »0 :« angenommen (Normalfall).

Mit einem Textteil, der für eine ganze Zeile des Kopftextes gilt, wird immer eine neue Zeile begonnen. Ein Textteil, der über einer bestimmten Spalte stehen soll, wird in die gleiche Zeile wie der vorangehende Textteil eingetragen, falls diese Zeile nicht schon einen Text für die ganze Zeile oder für diese oder eine weiter rechts stehende Spalte enthält; andernfalls wird mit diesem Textteil eine neue Zeile begonnen.

#### KTZ

Angabe, ob auf der ersten Seite der Kopftext gedruckt werden soll.

[1] <0>

0 = Kopftext unterdrücken

1 = Kopftext drucken

- FT** Textteile, die als Fußtext unten auf jeder Seite gedruckt werden sollen. [ II ] <>
- Es gelten die gleichen Regelungen wie für den Kopftext (Parameter `KT`). Die Seitennummer kann jedoch nicht in den Fußtext eingesetzt werden, wenn sie schon in den Kopftext eingesetzt worden ist.
- DRT** Druckertyp, für den die Daten aufbereitet werden. [ XI ]
- Wenn zur Spezifikation `MODUS` kein Druckertyp angegeben wurde, ist dieser Parameter obligat. Andernfalls darf dieser Parameter nicht angegeben werden.
- Hinweis: Mit dem Kommando `#LISTE, DRUCKERTYPEN` werden die definierten Druckertypen aufgelistet.

### Angabe zur Nummerierung

- NR** Erste Seitennummer für die Ausgabe. [ I ] <1>
- ROM** Angabe, ob die Seitenzahl in arabischen Ziffern oder als römische Zahlen in den Kopf- bzw. den Fußtext eingesetzt werden soll. [ I ] <0>
- 0 = arabische Ziffern  
1 = römische Zahlen mit Kleinbuchstaben  
2 = römische Zahlen mit Großbuchstaben
- FNN** Nummer, mit der die Fußnotennummerierung begonnen werden soll. [ I ] <1>

### Ersetzen von Zeichenfolgen

- X** Zeichenfolgenpaare (und Ausnahmezeichenfolgen), deren jeweils erste Zeichenfolge durch die jeweils zweite Zeichenfolge bei der Eingabe von der `QUELL`-Datei ersetzt werden soll. [ X ]
- Hiermit können z. B. Kürzel, die nicht mit einem Dollarzeichen beginnen bzw. nicht in spitze Klammern eingeschlossen sind, mit einem Dollarzeichen versehen werden, damit sie als Makro erkannt und entsprechend dem Parameter `MKR` aufgelöst werden.
- Das Ersetzen erfolgt im Eingabesatz. Dabei wird zuvor am Anfang und am Ende des Satzes je ein Leerzeichen ergänzt, das nach dem Ersetzen wieder entfernt wird.
- Soll die Silbentrennung aufgehoben werden (Parameter `STR`), so wird nach dem Ersetzen festgestellt, ob der Satz mit einem Silbentrennzeichen endet.
- XPR** Zeichenfolgenpaare (und Ausnahmezeichenfolgen), deren jeweils erste Zeichenfolge durch die jeweils zweite Zeichenfolge bei der Ausgabe in die `PROTOKOLL`-Datei ersetzt werden soll. [ X ]

Das Ersetzen erfolgt für jedes Wort einzeln (es können also niemals Leerzeichen ausgetauscht werden).

**XXX** Zeichenfolgenpaare (und Ausnahmezeichenfolgen), deren jeweils erste Zeichenfolge durch die jeweils zweite Zeichenfolge bei der Ausgabe in die ZIEL-Datei ersetzt werden soll. [ X ]

Hiermit können z. B. Dollarzeichen, die zur Kennzeichnung von Kürzeln als Makros mit dem Parameter X eingefügt wurden, wieder entfernt werden.

## Makro-Definitionen

**MKR** Paare von Textteilen, wobei der jeweils erste Textteil das Makrokürzel (ohne Dollarzeichen bzw. spitze Klammern) angibt, das durch den jeweils zweiten Textteil aufgelöst werden soll. [ IV ]

Die Auflösung des Makros erfolgt nur logisch. In die ZIEL-Datei wird das unaufgelöste Makro ausgegeben.

**MKZ** Angabe, auf welche Weise Makros gekennzeichnet sind. [ I ] <1>

0 = Eingabedaten enthalten keine Makros.

1 = Makros sind mit dem Dollarzeichen gekennzeichnet.

2 = Makros sind mit den spitzen Klammern oder mit dem Dollarzeichen gekennzeichnet.

## Angaben zum Randausgleich

**REZ** Angabe, ob der Rand einer einzelnen Zeile (also der ersten Zeile eines Abschnitts mit nur zwei Zeilen) ausgeglichen werden soll. [ I ] <1>

0 = Rand soll nicht ausgeglichen werden

1 = Rand soll ausgeglichen werden

## Umdrehen von Textteilen

In der Regel braucht keiner der folgenden Parameter angegeben zu werden, da die Voreinstellungen die Standardfälle abdecken.

Arabische, hebräische und syrische Textteile müssen von rechts nach links gedruckt werden. Das Kommando #DRUCKE kann aber nur von links nach rechts drucken. Deshalb ist es notwendig, diese Textteile umzudrehen. Dabei ist u. a. zu berücksichtigen, dass Zahlen, Steuerzeichen und Codierungen, die aus mehr als einem Zeichen bestehen, nicht umgedreht werden.

**AUM** Zeichenfolgen, die den Anfang der Textteile kennzeichnen, die umgedreht werden sollen. [ IX ]

Voreinstellung:

AUM ' #[AHY]+'

**EUM**

Zeichenfolgen, die das Ende der Textteile kennzeichnen, die umgedreht werden sollen. [ IX ]

Voreinstellung:

EUM ' #[CGKPR]+'#[AHY?CGKPR]-'

**NUM**

Zeichenfolgen, die nicht umgedreht werden sollen, falls sie in den umzudrehenden Textteilen vorkommen. [ IX ]

Voreinstellung:

>KL () <> \ [ \ ] \ { \ }  
 <AZ '%{%' '%{%' '{%' '  
 <BU '{\a}' '#. {\a}' '\_'  
 NUM '{#}' '# [=+-] {3} {0}' '# [', ; ! . " ] ?'  
 NUM '# ({00} {!})' '# {!} [+ -] '  
 NUM '{C:KL}' '{00}' {S:AZ} {S:BU}'

**XUM**

Zeichenfolgenpaare (und Ausnahmezeichenfolgen), deren jeweils erste Zeichenfolge durch die jeweils zweite Zeichenfolge innerhalb der nicht umzudrehenden (mit dem Parameter NUM angegebenen) Zeichenfolgen ersetzt werden soll. [ X ]

Voreinstellung:

<AZ '%{%' '%{%' '{%' '  
 XUM '# [0123456789/SF]+' '# {+2=} -'  
 XUM '# [0123456789/SF]-' '# {+2=} +' '  
 XUM '# . : ' # . ; ' # . ; ' # . : '  
 XUM ' ( ' ) ' ' ( ' \ [ ' \ ] ' \ ] ' \ [ ' \ { ' \ } ' \ } ' \ { ' < ' > ' > ' < '  
 XUM ' ' {S:AZ} ' # " < ' # " > ' # . ( ' # . ) ' # ( {00} {!} ) ' ' '



## Alphabetisches Verzeichnis der Parameter

<b>AUM</b>	Anfangskennung zum Umdrehen . . . . .	903
<b>BER</b>	Auswählen eines Bereichs aus der QUELL-Datei . . . . .	899
<b>DR</b>	Druckausgabesteuerung . . . . .	899
<b>DRT</b>	Druckertyp . . . . .	902
<b>DRZ</b>	Druckausgabesteuerung – zusätzliche Angaben . . . . .	900
<b>EUM</b>	Endekennung zum Umdrehen . . . . .	904
<b>FNN</b>	Erste Fußnotennummer . . . . .	902
<b>FT</b>	Fußtext . . . . .	902
<b>KT</b>	Kopftext . . . . .	900
<b>KTZ</b>	Kopftext – zusätzliche Angaben . . . . .	901
<b>MAX</b>	Maximum für Testzwecke . . . . .	899
<b>MKR</b>	Makro-Definitionen . . . . .	903
<b>MKZ</b>	Makro-Kennzeichnung . . . . .	903
<b>NR</b>	Erste Seitennummer . . . . .	902
<b>NUM</b>	Nichtumzudrehende Zeichenfolgen . . . . .	904
<b>PAR</b>	Parameter-Konvention . . . . .	898
<b>REZ</b>	Randausgleich von Einzelzeilen . . . . .	903
<b>ROM</b>	Römische Seitenzahlen . . . . .	902
<b>STR</b>	Angaben zur Silbentrennung . . . . .	898
<b>X</b>	Austauschen bei der Eingabe . . . . .	902
<b>XPR</b>	Austauschen für das Protokoll . . . . .	902
<b>XUM</b>	Austauschen beim Umdrehen . . . . .	904
<b>XXX</b>	Austauschen bei der Ausgabe . . . . .	903

## Makros

FORMATIERE-Makros sind Kürzel, die stellvertretend für Steueranweisungen und/oder Textteile stehen. Diese Makros ermöglichen einerseits eine kürzere und übersichtlichere Schreibweise wiederkehrender Steueranweisungen und Textteile; sie erlauben andererseits eine inhaltliche Auszeichnung (statt einer typographischen Auszeichnung) eines Textes.

Makros müssen in den Daten entweder mit einem Dollarzeichen oder mit spitzen Klammern gekennzeichnet sein. Wird das Dollarzeichen als Kennzeichen verwendet, gilt als »Name« des Makros die Zeichenfolge, die zwischen dem Dollarzeichen und dem darauffolgenden Leerzeichen bzw. der darauffolgenden (mit \$, & oder @ beginnenden) Steueranweisung steht. Werden die spitzen Klammern als Kennzeichen verwendet, gilt als »Name« die zwischen den Klammern stehende Zeichenfolge; unpaarige spitze Klammern werden wie andere zu druckende Zeichen behandelt. Die »Namen« der Makros können aus beliebigen Zeichen bestehen. Sie müssen jedoch vollständig (einschließlich der Kennzeichen) in einer Zeile stehen. Groß- und Kleinbuchstaben werden unterschieden.

Für jedes in den Eingabedaten vorkommende Makro muss mit dem Parameter MKR (siehe Seite 903) die Auflösung definiert werden, d. h. es muss angegeben werden, für welche Steueranweisungen und/oder Textteile es steht. In der Auflösung dürfen auch andere Makros vorkommen. Mit dem Parameter MKZ (siehe Seite 903) muss angegeben werden, in welcher Weise die Makros in den Daten gekennzeichnet sind.

Unabhängig davon, ob unmittelbar vor oder hinter einem Makro Leerzeichen in den Eingabedaten stehen, arbeitet das Programm so, als stünden vor und hinter dem Makro Leerzeichen. Dies ist meist bedeutungslos. Wenn die Auflösung des Makros jedoch mit einem Textteil beginnt, der an das vor dem Makro stehende Wort ohne Zwischenraum angehängt werden soll, so muss die Steueranweisung 6 (& ,) vor dem Textteil in die Auflösung eingefügt werden. Entsprechend muss diese Steueranweisung auch nach einem Textteil am Ende der Auflösung eingefügt werden, wenn das nach dem Makro stehende Wort an diesen Textteil ohne Zwischenraum angehängt werden soll. Dies ist insbesondere immer dann der Fall, wenn die Auflösung eines Makros mit einem Code für Schriftumschaltung (z. B. #G+ und #G-) oder Unterstreichung (z. B. #1+ und #1-, vgl. Beispiel auf Seite 897) endet bzw. beginnt.

Makros werden nur logisch aufgelöst; die Auflösungen werden für die Ausgabe in die PROTOKOLL-Datei ausgewertet. Falls eine ZIEL-Datei angegeben ist, wird jedes Makro in unveränderter Form in diese ausgegeben. Dies geschieht, nachdem die Auflösung des Makros vollständig ausgewertet ist. Wenn die Auflösung Steueranweisungen und/oder Textteile enthält, die einen Zeilenwechsel bewirken, kann es sinnvoll sein, das Makro schon während der Auswertung in die ZIEL-Datei auszugeben, damit es dort am Ende der vorangehenden Zeile steht anstatt am Anfang der neuen Zeile. Dazu kann an der Stelle, an der das Makro während der Auswertung ausgegeben werden soll, ein Dollarzeichen in die Auflösung eingefügt werden. Dies ist in der Regel bei Makros sinnvoll, die am Ende einer logischen Einheit (z. B. an einem Überschriftsende, vgl. Beispiel auf Seite 897) stehen.

## Steueranweisungen

Jede Steueranweisung gilt zugleich als Trennzeichen zwischen zwei Wörtern. Es ist also gleichgültig, ob eine Anweisung in Leerzeichen eingeschlossen ist oder nicht. Falls jedoch bei einer mit »&« beginnenden Anweisungen das unmittelbar nachfolgende Wort mit einer oder mehreren Ziffern beginnt, muss zwischen der Anweisung und dem Wort mindestens ein Leerzeichen stehen, da sonst angenommen wird, dass die Ziffern zur Anweisung gehören. Ebenso muss nach einem Makrokürzel, das mit einem Dollarzeichen gekennzeichnet ist, mindestens ein Leerzeichen folgen, falls unmittelbar danach keine Anweisung sondern Text folgt.

Die Buchstaben in den Steueranweisungen können Groß- oder Kleinbuchstaben sein.

Die Angabe  $n$  steht für eine vorzeichenlose ganze Zahl. Sie wird nach rechts durch ein beliebiges Zeichen (außer Ziffer) begrenzt. Die Angabe » $n =$  « in der Beschreibung meint den Fall, in dem  $n$  fehlt, wenn also keine Zahl angegeben ist.

Die Angaben bzw. Definitionen der Anweisungen, die in dieser Beschreibung mit einem »+« oder »-« gekennzeichnet sind, gelten jeweils so lange, bis sie durch eine entsprechende Anweisung aufgehoben oder geändert werden. Die Angaben bzw. Definitionen der Anweisungen, die mit »+« gekennzeichnet sind, werden zusätzlich im jeweils gültigen »Format« gemerkt und werden beim Umschalten auf ein anderes Format entsprechend neu eingestellt. Alle nicht gekennzeichneten Anweisungen haben nur lokale Auswirkungen.

## Wortzwischenraum

Als Wortzwischenraum gilt neben Leerzeichen auch jede Steueranweisung. Mehrere unmittelbar aufeinander folgende Leerzeichen und/oder Steueranweisungen werden als 1 Wortzwischenraum behandelt, falls dies durch die Anweisung 4 (@-) nicht anders festgelegt wird.

## Blocksatz

Das Programm gleicht automatisch den Rand aus, wenn dies nicht durch eine der folgenden Anweisungen anders festgelegt wird. Dabei werden an vorhandenen Wortzwischenräumen zusätzliche Leerstellen in die Zeile eingefügt, bis der Randausgleich erreicht ist.

Wenn ein Wortzwischenraum beim Randausgleich nicht erweitert werden soll, so muss dafür »\_« (Unterstreichungsstrich) geschrieben werden. Ein so geschriebener Wortzwischenraum wirkt nicht als Trennzeichen zwischen zwei Wörtern.

@R- 1 + Randausgleich aus

@R+ 2 + Randausgleich ein, jedoch bei Einzelzeilen Randausgleich unterlassen (Grundeinstellung)

- @R** 3 - Rand in der aktuellen Zeile ausgleichen. Die Anweisung 19 (@Z) hat jedoch Vorrang gegenüber dieser Anweisung.
- @-** 4 + Mehrere unmittelbar aufeinander folgende Leerzeichen beibehalten; Zeilenwechsel in den Eingabedaten wirkt wie die Anweisung &?.  
Das bedeutet, dass das Programm versucht, die Textenteilung der Eingabedatei (z. B. für fertige Tabellen) beizubehalten. Im Text enthaltene Anweisungen werden jedoch ausgeführt.
- @+** 5 Aufheben der Anweisung 4 (Grundeinstellung)
- &, n** 6 Wortzwischenraum vor dem nachfolgenden Wort auf n Leerzeichen festlegen.  
n = : Wortzwischenraum aufheben
- &Xn** 7 + Falls zum Randausgleich insgesamt mehr als n Leerzeichen in eine Zeile eingeschoben werden müssten, Rand in dieser nicht ausgleichen (Grundeinstellung n=999).  
n = 0 : (noch nicht definiert)  
n = : (noch nicht definiert)
- &Yn** 8 + Falls zum Randausgleich irgendwo in der Zeile mehr als n Leerzeichen eingeschoben werden müssten, Rand in dieser nicht ausgleichen (Grundeinstellung n=999).  
n = 0 : (noch nicht definiert)  
n = : (noch nicht definiert)

## Abschnitte, Zeilen- und Seitenumbruch, Leerzeilen

Abschnittsanfang beinhaltet auch Abschnittsende für den vorangehenden Text. Das Programm achtet darauf, dass weder die erste Zeile eines Abschnitts als letzte unten, noch die letzte Zeile eines Abschnitts als erste oben auf einer Seite (bzw. in einer Spalte) steht. Das hat zur Folge, dass ggf. eine Seite (bzw. eine Spalte) nicht bis zur letzten Zeile bedruckt wird, sondern notfalls ein oder zwei Zeilen nach oben auf die nächste Seite (bzw. Spalte) gebracht werden, um diese Forderung zu erfüllen.

Enthält jedoch die letzte Zeile eines Abschnitts einen Verweis auf eine Fußnote, so kann es in ungünstigen Fällen vorkommen, dass diese Zeile oben auf eine neue Seite gedruckt wird.

Folgen in den Eingabedaten n Leerzeilen unmittelbar aufeinander, haben diese die gleiche Wirkung wie die Anweisung &!n (n = Anzahl der Leerzeilen).

- &!n** 9 Abschnittsanfang, davor n Leerzeilen. Diese werden ggf. am Spaltenanfang unterdrückt.  
Werden mit dieser Anweisung mehrmals Leerzeilen (ohne dazwischen stehenden Text) verlangt, so gilt nur die Anforderung mit der größten Anzahl Leerzeilen.

n = 0 : 1/2 Leerzeile  
 n = : keine Leerzeile

Der Zeilenabstand der Leerzeilen ist gleich groß wie bei zu druckenden Zeilen. Er ist 1/6 Zoll (das entspricht 6 Zeilen je Zoll), wenn mit der Anweisung 14 (&v<sub>n</sub>) nichts anderes eingestellt wurde.

Einschränkung: 1 1/2-zeiliger Vorschub ist nur bei einspaltigem Ausdruck (1. Zahlenwert des Parameters DR = 1) möglich. Außerdem ist er nur bei Matrix- und Laserdruckern, nicht aber bei Zeilendruckern möglich. Falls 1 1/2-zeiliger Vorschub nicht möglich ist, wird die Anweisung &!0 wie &! gewertet.

- &?n** 10 Abschnittsanfang und neue Spalte (d. i. neue Seite bei einspaltigem Druck), falls weniger als n Zeilen frei sind. Dabei werden die Leerzeilen, die eventuell vor der nächsten zu druckenden Zeile auszugeben sind, nicht mitgezählt.

Wird diese Anweisung mehrmals (ohne dazwischen stehenden Text) gegeben, so gilt nur die Anweisung mit dem größten Wert von n; die Anweisung für Seitenwechsel (&?0) hat jedoch in jedem Fall Vorrang.

n = 1 : wie &?, falls der nachfolgende Text bis zur nächsten Zeilenwechselanweisung in 1 Zeile passt, sonst wie &?2.

n = 0 : neue Seite

n = : nur neue Zeile, kein Abschnittsanfang

Wieviel Platz für die n Zeilen, die noch frei sein müssen, berücksichtigt wird, ist vom eingestellten Zeilenabstand abhängig. Er ist 1/6 Zoll (das entspricht 6 Zeilen je Zoll) je Zeile, wenn mit der Anweisung 14 (&v<sub>n</sub>) nichts anderes eingestellt wurde.

- @\$** 11 Neue Zeile, falls die nachfolgende (absolute) Positionierung nicht wie gewünscht möglich ist.
- @\$+** 12 – Wie Anweisung 11, jedoch für alle nachfolgenden Positionierungen. Anweisung 11 hebt in diesem Fall die Anweisung 12 für die nächste Positionierung auf.
- @\$-** 13 – Aufheben der Anweisung 12 (Grundeinstellung)

## Zeilenabstand

- &v<sub>n</sub>** 14 + Zeilenabstand auf n (max. 4) Zeilen einstellen (Grundeinstellung n=1).

n = 0 : 1 1/2-zeiliger Druck (4 Zeilen je Zoll)

n = 1 : 1-zeiliger Druck (6 Zeilen je Zoll)

n = 2 : 2-zeiliger Druck (3 Zeilen je Zoll)

n = 3 : 3-zeiliger Druck (2 Zeilen je Zoll)

n = 4 : 4-zeiliger Druck (1 1/2 Zeilen je Zoll)

n = : (noch nicht definiert)

Einschränkung: 1 1/2-zeiliger Vorschub ist nur bei einspaltigem Ausdruck (1. Zahlenwert des Parameters DR = 1) möglich. Außerdem ist er nur bei Matrix- und Laserdruckern, nicht aber bei Zeilendruckern möglich. Falls 1 1/2-zeiliger Vorschub nicht möglich ist, wird die Anweisung &V0 wie &V1 gewertet.

## Einrücken

Vorbemerkung: Folgezeilen sind Zeilen, die durch automatischen Zeilenwechsel, also nicht durch eine Zeilenwechsellanweisung (Anweisung 9 und 10), begonnen werden.

- &\$n** 15 + Am Beginn eines durch Anweisung 9 (&!n) eingeleiteten Abschnitts n Zeichen vom linken Rand einrücken (Grundeinstellung n=0).  
n = : einrücken wie am Beginn eines Abschnitts
- &Fn** 16 + Folgezeilen n Zeichen vom linken Rand einrücken (die Anweisungen 17 und 18 haben Vorrang; Grundeinstellung n=0).  
n = : einrücken wie bei Folgezeilen
- &:n** 17 Rest der Zeile und Folgezeilen (bis zur nächsten Zeilenwechsellanweisung) n Zeichen vom linken Rand einrücken (hat Vorrang gegenüber Anweisung 16).  
n = : (noch nicht definiert)
- &;n** 18 Folgezeilen (bis zur nächsten Zeilenwechsellanweisung) n Zeichen vom linken Rand einrücken (hat Vorrang gegenüber Anweisung 16).  
n = : Folgezeilen bis zur aktuellen Position (relativ zum linken Rand) einrücken

## Zentrieren

- @Z** 19 Nachfolgenden Text (bis zur nächsten Positionierung oder nächsten Zeilenwechsellanweisung oder zu einer der Anweisungen 19 bis 21) auf Mitte zentrieren
- @/** 20 Nachfolgenden Text (bis zur nächsten Positionierung oder nächsten Zeilenwechsellanweisung oder zu einer der Anweisungen 19 bis 21) nach rechts schieben
- @.** 21 Wie Anweisung 20, jedoch den Zwischenraum auspunktieren
- &Zn** 22 + Feld vor dem Tabulator n (max. 20) auf Mitte zentrieren (wirkt dann, wenn auf den Tabulator n positioniert wird)  
n = 0 : keine Felder auf Mitte zentrieren (Grundeinstellung)  
n = : alle Felder auf Mitte zentrieren

- &/n** 23 + Feld vor dem Tabulator  $n$  (max. 20) rechtsbündig drucken (wirkt dann, wenn auf den Tabulator  $n$  positioniert wird)
- $n = 0$  : keine Felder rechtsbündig drucken (Grundeinstellung)  
 $n =$  : alle Felder rechtsbündig drucken
- &.n** 24 Wie Anweisung 23, jedoch den Zwischenraum auspunktieren

## Positionieren

Falls positioniert werden soll (Anweisungen 17, 25-27, 29, 32-33) und die aktuelle Position weiter rechts ist als die gewünschte Position, so wird statt der Positionierung ein Leerzeichen eingesetzt, falls mit Anweisung 11 oder 12 nicht verlangt wurde, dass in diesem Fall eine neue Zeile begonnen und in dieser positioniert werden soll. Stehen jedoch unmittelbar links von der aktuellen Position Leerzeichen, so wird bis zum letzten Nicht-Leerzeichen zurückgegangen und die Positionierung nochmals versucht.

- &=n** 25 Positionieren auf Position  $n$
- $n = 0$  : aktuelle Position (abs.) merken  
 $n =$  : auf die mit der Anweisung  $\&=0$  gemerkte Stelle positionieren
- &+n** 26 Um  $n$  Stellen nach rechts positionieren
- $n = 0$  : Position (relativ zum linken Rand) merken  
 $n =$  : auf die mit der Anweisung  $\&+0$  gemerkte Stelle positionieren
- &-n** 27 Um  $n$  Stellen nach links positionieren
- $n = 0$  : Position (relativ zum rechten Rand) merken  
 $n =$  : auf die mit der Anweisung  $\&-0$  gemerkte Stelle positionieren

## Tabulatoren

- &Mn** 28 + Tabulator  $n$  (max. 20) auf die aktuelle Position setzen
- $n = 0$  : alle Tabulatoren löschen (Grundeinstellung)  
 $n =$  : nächsten Tabulator (d. i. der Tabulator mit der Nummer 1, falls zum ersten Mal in der Zeile ein Tabulator angesprochen wird) auf die aktuelle Position setzen
- &Tn** 29 Auf Tabulator  $n$  (max. 20) positionieren.
- $n = 0$  : Auf den mechanisch nächsten Tabulator positionieren (d. h. auf den Tabulator, der auf die aktuelle Position oder weiter rechts gesetzt ist und der der Nummer nach als nächster auf den Tabulator folgt, der zuletzt in der gleichen Zeile angesprochen wurde bzw. die niedrigste Nummer hat, falls zum ersten Mal in der Zeile ein Tabulator angesprochen wird). Falls es keinen solchen gibt, wird auf den rechten Rand positioniert. Falls die Tabulatorpositionen alle aufsteigend sind, entspricht die Funktionsweise dieser Anweisung also der Tabulatortaste auf einer Schreibmaschine.

$n =$  : Auf den logisch nächsten Tabulator positionieren (d. h. auf den Tabulator, dessen Nummer um 1 größer ist als die des zuletzt in der gleichen Zeile angesprochenen Tabulators bzw. auf den Tabulator mit der Nummer 1, falls zum ersten Mal in der Zeile ein Tabulator angesprochen wird). Falls dieser nicht (mit der Anweisung 28) definiert wurde, wird die aktuelle Position nicht verändert.

## Ränder

Die Abstände zum Papierrand werden mit den Parametern DR (linker und rechter Rand) und DRZ (oberer und unterer Rand) festgelegt. Die folgenden Steuerzeichen ziehen den Rand gegenüber den mit dem Parameter DR festgelegten Rändern nach innen ein.

- &Ln** 30 + Linken Rand auf  $n$  Zeichen festlegen (Grundeinstellung  $n=0$ ).  
 $n =$  : linken Rand auf aktuelle Position festlegen
- &Rn** 31 + Rechten Rand auf  $n$  Zeichen festlegen (Grundeinstellung  $n=0$ ).  
 $n =$  : rechten Rand auf aktuelle Position festlegen
- &An** 32 Wie Anweisung 30 und zusätzlich auf den linken Rand positionieren.  
 $n =$  : nur auf den linken Rand positionieren
- &En** 33 Wie Anweisung 31 und zusätzlich auf den rechten Rand positionieren.  
 $n =$  : nur auf den rechten Rand positionieren

## Fußnoten

Fußnoten müssen zwischen @F+ und @F- eingeschlossen werden. Sie werden beim Formatieren automatisch durchnummeriert und ans Ende der jeweiligen Seite gestellt. Der Fußnotenverweis (die Fußnotennummer) wird unmittelbar ans vorangehende Wort angehängt.

Soll der Verweis nicht am Ende des Wortes eingefügt werden, sondern weiter links im Wort (z. B. vor dem Komma am Ende des Wortes), so kann mit der Anweisung 37 (z. B. &"1) vor der Anweisung 34 angegeben werden, um wieviele Zeichen (Druckpositionen, nicht Zeichen der Eingabe-Codierung) der Verweis weiter links eingefügt werden soll.

Einschränkung: Fußnoten sind nur bei einspaltigem Ausdruck (1. Zahlenwert des Parameters DR = 1) möglich.

- @F+** 34 Fußnotenanzug
- @F-** 35 Fußnotenende
- @F** 36 Nummerierung der Fußnoten (bei der nächsten Anweisung 34) wieder mit 1 beginnen.



- &"n** 37 Verweis auf nachfolgende Fußnote nicht ans Ende des vorangehenden Wortes, sondern um *n* Zeichen (Druckpositionen, nicht Zeichen der Eingabe-Codierung) weiter links einfügen.
- n* = 0 : Fußnotenverweis unmittelbar nach dem letzten zu druckenden Zeichen des vorangehenden Wortes (z. B. zwischen »t« und »#1-« bei »#1+Wort#1-«) einfügen.
- n* = : (noch nicht definiert)

## Datum und Uhrzeit einfügen

- &Dn** 38 Einfügen des aktuellen Datums.
- n* = : (noch nicht definiert)
- n* = 0 : Wochentag (z. B. Sonntag)
- n* = 1 : Datum xx.xx.xx (z. B. 02.04.96)
- n* = 2 : Datum xx. xxx. xxxx (z. B. 2. Apr. 2008)
- n* = 3 : Datum xx. xxxxxxxxxxxx xxxx (z. B. 2. April 2008)
- &Un** 39 Einfügen der aktuellen Uhrzeit.
- n* = 1 : Uhrzeit xx.xx (z. B. 12.00)
- n* = 2 : Uhrzeit xx:xx (z. B. 12:00)

## Auffüllen, Wiederholen

- @\*** 40 Nachfolgenden Text (bis zur nächsten Positionierung oder nächsten Zeilenwechsellanweisung oder zu einer der Anweisungen 19 bis 21) so oft wiederholen, bis der Zwischenraum (bis zur angegebenen Position bzw. bis zum rechten Rand) aufgefüllt ist.
- Achtung: Wenn der zu wiederholende Text mit einer der Anweisungen 19 bis 21 abgeschlossen wird, wird bis zum rechten Rand aufgefüllt und eine neue Zeile begonnen.

## Druck unterdrücken

- @D** 41 Nächstes Wort nicht drucken; diese Anweisung hebt ggf. die Anweisung 42 auf.
- @D-** 42 - Druck aus (d. h. der nachfolgende Text wird nicht gedruckt, sondern es wird entsprechend viel Raum freigelassen)
- @D+** 43 - Druck ein (Grundeinstellung)

## Text und Steueranweisungen ignorieren

- @I** 44 Nächstes Wort bzw. nächste Anweisung ignorieren.

Hinweis: Damit können z. B. Kennzeichnungen für Registereinträge unterdrückt werden.

**@I+** 45 - Nachfolgende Wörter bzw. Anweisungen ignorieren.

Hinweis: Damit können z. B. Registereinträge, die zusätzlich im Text enthalten sind, übergangen werden.

**@I-** 46 - Aufheben der Anweisung 45 (Grundeinstellung)

## Silbentrennung

Silbentrennung in den Eingabedaten wird aufgehoben, wenn mit dem Parameter STR nichts anderes angegeben ist. Als Silbentrennzeichen gilt ein »-«, das als letztes Zeichen in einer Zeile (= Eingabesatz) steht, wenn das zweitletzte Zeichen ebenfalls ein »-« ist, oder wenn das zweitletzte Zeichen ein Buchstabe und das drittletzte Zeichen kein Steuerzeichen (\$, &, @, \, \_, #, %) ist.

Soll ein »-« am Zeilenende nicht als Silbentrennzeichen interpretiert werden, so kann unmittelbar nach dem »-« ein »\« ergänzt werden. Das Zeichen »\« wird beim Ausdrucken ignoriert.

Beim Aufheben der Silbentrennung wird ein getrenntes »ck«, das als »k-« und »k« geschrieben ist, nicht wieder zu »ck«.

Das Programm trennt bei Bedarf Wörter nach den Regeln der deutschen Sprache. Zusätzlich können innerhalb eines Wortes Stellen mit »\« gekennzeichnet werden, an denen vorzugsweise getrennt werden soll, und mit »\\«, an denen keinesfalls getrennt werden darf. Beginnt ein Wort mit »\«, so wird es bei Bedarf nur an den mit »\« gekennzeichneten Stellen getrennt.

Bei der Ausgabe in die ZIEL-Datei bleibt eine eventuell durchgeführte Silbentrennung unberücksichtigt. Ein im Protokoll getrenntes Wort wird in ungetrennter Form in die ZIEL-Datei ausgegeben.

**@S-** 47 + Silbentrennung aus (d. h. nur noch an mit »\« gekennzeichneten Kann-Trennstellen trennen)

**@S+** 48 + Silbentrennung ein (Grundeinstellung)

**&Sn** 49 + Silbentrennung ein. Aber Wörter nur trennen, falls noch mehr als n Zeichen für das betreffende Wort in der Zeile frei sind (Grundeinstellung n=2) oder an Stellen, die mit »\« gekennzeichnet sind.

n = 1 : Wort nur trennen, wenn zum Randausgleich irgendwo mehr als 1 Leerzeichen eingeschoben werden müsste

n = 0 : Wort nur an den Stellen trennen, die durch »\« gekennzeichnet sind

n = : (noch nicht definiert)

**@\+** 50 + Enthält ein Wort ein Leerzeichen (dies kann ein mit »\_« codiertes Leerzeichen sein oder eines, das beim Ersetzen auf Grund des Parameters XPR eingefügt wurde und deshalb nicht als Worttrennzeichen wirkt), darf an dieser Stelle eine neue Zeile begonnen werden, falls das Wort nicht mehr in die Zeile passt.

- @\-** 51 + Enthält ein Wort ein Leerzeichen (dies kann ein mit »\_« codiertes Leerzeichen sein oder eines, das beim Ersetzen auf Grund des Parameters XPR eingefügt wurde und deshalb nicht als Worttrennzeichen wirkt), darf an dieser Stelle keine neue Zeile begonnen werden. (Grundeinstellung)
- @\** 52 Nachfolgendes Wort nicht trennen (außer an einer mit »\« gekennzeichneten Kann-Trennstelle)

## Markierungen

Einschränkung: Das Markieren der Zeilen ist nur bei einspaltigem Ausdruck (1. Zahlenwert des Parameters DR = 1) möglich.

- @M** 53 Die Zeile, in der das folgende Wort steht, markieren; diese Anweisung hebt ggf. die Anweisung 54 auf.
- @M+** 54 – Markierung der Zeilen ein. Die Markierung (senkrechter Strich am rechten Rand) kann z. B. verwendet werden, um bei Neufassungen die geänderten Stellen zu kennzeichnen.
- @M-** 55 – Markierung der Zeilen aus (Grundeinstellung)

## Formate

Zu einem »Format« gehören alle Einstellungen (z. B. linker und rechter Rand, Tabulatoren), die mit Anweisungen geändert werden können, die mit einem »+« gekennzeichnet sind. Es gibt 9 Formate, die von 1 bis 9 nummeriert sind.

Beim Start des Programms werden alle Formate mit den Grundeinstellungen definiert und es wird Format 1 eingestellt. Die Grundeinstellungen sind jeweils bei den mit »+« gekennzeichneten Anweisungen angegeben. Beim Wechsel in ein anderes Format (mit einer der Anweisungen 56–58) gelten dann jeweils die Einstellungen, die in diesem Format zuletzt definiert waren.

Formate können z. B. benutzt werden, wenn verschiedene Tabellenformen vorkommen, die sich wiederholen. Wird für jede Tabellenform ein Format verwendet, brauchen die Tabulatoren jeweils nur einmal gesetzt zu werden.

- &Wn** 56 – Wechseln auf Format mit der Nummer  $n$  (max. 9).
- $n = 0$  : Werte von der Grundeinstellung übernehmen  
 $n =$  : Werte vom zuvor eingestellten Format übernehmen.
- @W+** 57 + Auf Format mit der nächsthöheren Nummer wechseln
- @W-** 58 + Auf das zuvor eingestellte Format wechseln

## Zeichenvorrat

Der Zeichenvorrat (mit den Eingabe-Codierungen) ist der Beschreibung »TUSTEP-Grundlagen« zu entnehmen, ebenso die verschiedenen Möglichkeiten der Hoch- und Tiefstellung von Zeichen sowie der Auszeichnungen (Unterstreichung, Sperrung, Fettdruck usw.).

Sollen die Zeichen #, \$, %, &, @, \, \_ gedruckt werden, so muss dafür ^#, ^\$, ^%, ^&, ^@, ^\, ^\_ geschrieben werden. Die Codierung weiterer Sonderzeichen ist der Beschreibung »TUSTEP-Grundlagen« zu entnehmen.

\* \* \* \* \*

**#KAUSFUEHRE**

---

Kommando . . . . .	919
Leistung . . . . .	920
Korrekturanweisungen . . . . .	921
Angaben der zu korrigierenden Textbereiche . . . . .	922
Abkürzende Schreibweisen . . . . .	922
Anmerkungen . . . . .	923
Korrekturprotokoll . . . . .	924
Parameter . . . . .	925
Einstellen der Parameter-Konvention . . . . .	925
Auswahl der Daten . . . . .	925
Angaben zum Protokoll . . . . .	925
Angaben zu den Ausgabesätzen . . . . .	927
Alphabetisches Verzeichnis der Parameter . . . . .	928

## Kommando

### #KAUSFUEHRE

QUELLE	= datei	Name der Datei mit den Daten, die korrigiert werden sollen.
	= -STD-	Die Daten, die korrigiert werden sollen, stehen in der Standard-Text-Datei.
ZIEL	= datei	Name der Datei für die korrigierten Daten.
	= -STD-	Die korrigierten Daten in die Standard-Text-Datei ausgeben.
MODUS	= -STD-	* Nummerierung der Sätze nach Möglichkeit beibehalten.
	= +	Sätze (im Textmodus) neu nummerieren.
LOESCHEN	= -	* Daten in der ZIEL- und in der PROTOKOLL-Datei nicht löschen.
	= +	Daten in der ZIEL- und in der PROTOKOLL-Datei zuerst löschen.
PARAMETER	= -	* Keine Parameter.
	= datei	Name der Datei mit den Parametern.
	= *	Die Parameter folgen auf das Kommando und sind mit *EOF abgeschlossen.
KORREKTUR	= datei	Name der Datei mit den Korrekturanweisungen.
PROTOKOLL	= -	* Kein Korrekturprotokoll erstellen, nur Fehlermeldungen ins Ablaufprotokoll ausgeben.
	= -STD-	Korrekturprotokoll in die Standard-Protokoll-Datei ausgegeben.
	= datei	Name der Datei für das Korrekturprotokoll.

## Leistung

Mit diesem Programm können Texte (ohne Verwendung eines Editors) korrigiert werden. Mit Korrekturanweisungen, die in einer Datei stehen, können

- Zeilen (z. B. Zeile 2 auf Seite 3)
- Wörter (z. B. 2. Wort in Zeile 3 auf Seite 4)
- Zeichen (z. B. 2. Zeichen in Zeile 3 auf Seite 4  
oder 2. Zeichen im 3. Wort in Zeile 4 auf Seite 5)

ersetzt, gelöscht oder eingefügt werden. Auf Wunsch wird ein Protokoll der ausgeführten Korrekturen erstellt.

## Hinweis

Die Korrekturanweisungen für dieses Programm werden meist mit dem Kommando #VERGLEICHE erstellt und dann überarbeitet.



## Korrekturanweisungen

Eine Korrekturanweisung hat folgende Bestandteile:

- Angabe des zu korrigierenden Textbereichs
- Angabe der Korrekturart durch die Korrekturzeichen »-«, »+«, »=«, »\*« für »Löschen«, »Einfügen«, »Ersetzen«, »Kommentar«
- Angabe des Korrekturtextes (bei Korrekturart »Einfügen« und »Ersetzen«).

Der Korrekturtext beginnt jeweils unmittelbar (also ohne zusätzlichen Zwischenraum) hinter dem Korrekturzeichen.

Die Korrekturanweisungen müssen aufsteigend nach den zu korrigierenden Textbereichen sortiert sein; nachzutragende Korrekturanweisungen müssen an der richtigen Stelle eingeordnet werden.

Jede Korrekturanweisung beginnt mit einer neuen Zeile. Ist der Korrekturtext zu lang für eine Eingabezeile, so kann er in der nächsten Zeile fortgesetzt werden. Solche Fortsetzungszeilen sind durch ein »+« am Zeilenanfang zu kennzeichnen. Silbentrennung darf in den auf mehrere Zeilen verteilten Korrekturanweisungen nicht vorgenommen werden.

Bei der Korrekturart »Einfügen« wird anstatt der Bereichsangabe die Angabe der Stelle (Zeile oder Wort oder Zeichen) erwartet, hinter der die Einfügung vorgenommen werden soll. Diese Stellenangabe hat die gleiche Form wie der vor dem »bis«-Strich stehende Teil der Bereichsangabe.

Zwischen der Angabe des zu korrigierenden Textbereichs und der Korrekturart kann noch ein in eckige Klammern eingeschlossener Originalwortlaut und eine in spitzen Klammern eingeschlossene Stellenangabe stehen. Näheres dazu siehe Beschreibung des Kommandos #VERGLEICHE unter »Aufbau eines Satzes in der KORREKTUR-Datei«. Diese zusätzlichen Angaben sind für #KAUSFUEHRE nicht von Bedeutung.

## Angabe des zu korrigierenden Textbereichs

Der zu korrigierende Textbereich wird durch Seiten- und Zeilennummern angegeben und kann durch Wort- und/oder Zeichennummern noch weiter eingegrenzt werden. Eine Bereichsangabe hat die Form:

$$s1.z1[/u1][,w1][:b1]-s2.z2[/u2][,w2][:b2]$$

Dabei steht:

- s1 bzw. s2 für die Seitennummer
- z1 bzw. z2 für die Zeilennummer
- u1 bzw. u2 für die Unterscheidungsnummer
- w1 bzw. w2 für die Wortnummer
- b1 bzw. b2 für die Zeichnummer.

Die in eckigen Klammern stehenden Teile können fehlen, wenn sie zur eindeutigen Bezeichnung des zu korrigierenden Textbereichs nicht notwendig sind. Die beiden Teile einer Bereichsangabe müssen den Textbereich jedoch in jeweils gleicher Weise eingrenzen (also durch »Zeile« oder »Wort« oder »Zeile:Zeichen« oder »Wort:Zeichen«).

- Ist hinter einer Zeilennummer keine Unterscheidungsnummer angegeben, so wird die Unterscheidungsnummer 0 angenommen.
- Sind eine oder mehrere *ganze Zeilen* von der Korrekturanweisung betroffen, so fehlen die Bestandteile Wortnummer und Zeichnummer.
- Wird der Korrekturbereich durch *ganze Wörter* eingegrenzt, so fehlt der Bestandteil Zeichnummer.
- Wird der Korrekturbereich durch *einzelne Zeichen* eingegrenzt und sind diese Zeichen jeweils vom *Zeilenanfang* bzw. *Zeilenende* an gezählt, so fehlt der Bestandteil Wortnummer. Die Zeichen können aber auch vom *Anfang* bzw. *Ende eines Wortes* an gezählt werden; nur in diesem Fall können alle Bestandteile der Bereichsangabe vorkommen.

## Abkürzende Schreibweisen für die Angabe der Textbereiche

Die Seitennummer kann weggelassen werden, wenn sie mit der zuletzt genannten Seitennummer übereinstimmt.

Die Seitennummer kann auch in eine eigene, durch »=« eingeleitete Zeile geschrieben werden. Beispiel:

=1256

2-3=Ersatz für Zeile 2 und 3 der Seite 1256

5,2+Einfügung hinter Zeile 5, 2. Wort der Seite 1256

Beim zweiten Teil der Bereichsangabe können, von links beginnend, alle Bestandteile, die mit den entsprechenden Bestandteilen des ersten Teils übereinstimmen, einschließlich der Trennzeichen weggelassen werden (Ausnahme: »/« vor der Unterscheidungsnummer muss stehen bleiben), wie in folgendem Schema angegeben:

s1.z1/u1,w1:b1[-[[[ [s2.]z2]/u2],]w2:]b2]

Insbesondere genügt der vor dem »bis«-Strich stehende Teil der Bereichsangabe, falls eine einzelne Zeile, ein einzelnes Wort oder ein einzelnes Zeichen gelöscht bzw. ersetzt werden soll.

Beispiele für gleichwertige Schreibweisen:

1.2-1.2                    und    1.2  
 1.2-1.3                    und    1.2-3  
 1.2-1.2/3                  und    1.2-/3  
 1.2,3-1.2,3                und    1.2,3  
 1.2,3-1.2,4                und    1.2,3-4  
 1.2:3-1.2:4                und    1.2:3-4  
 1.2,3:4-1.2,4:6          und    1.2,3:4-4:6

Bei den Bestandteilen »Wort« und »Zeichen« ist es möglich, die Wörter bzw. Zeichen von rechts beginnend zu zählen. Dazu wählt man aus den Zehnerpotenzen 10, 100, 1000, 10000, 100000 eine aus, die größer ist als die Summe aus der Anzahl der Wörter bzw. Zeichen in der Zeile (bzw. der Zeichen im Wort) und der von rechts gezählten Nummer des betreffenden Wortes bzw. Zeichens, und subtrahiert von dieser Zehnerpotenz die von rechts gezählte Wort- bzw. Zeichennummer (z. B.:  $100-2=98$  für das vorletzte Wort oder Zeichen einer Zeile mit weniger als 98 Wörtern bzw. Zeichen; gleichbedeutend kann 998 oder 9998 oder 99998 geschrieben werden).

## Anmerkungen

Beim Zählen der Wörter gilt jede Zeichenfolge als ein Wort, die von Zeilenanfang und/oder Zeilenende und/oder einem oder mehreren Leerzeichen eingeschlossen ist. Satzzeichen und Sonderzeichen gehören also entweder zum Wort oder bilden, wenn sie durch Leerzeichen davon getrennt sind, ein eigenes Wort. Leerzeichen gehören jeweils zum vorherigen Wort. Fester Ausschluss (»\_«) gilt nicht als Leerzeichen.

Beim Zählen der Zeichen in einer Zeile müssen Leerzeichen am Zeilenanfang mitgezählt werden.

Sollen am Zeilenanfang ein oder mehrere Wörter bzw. Zeichen eingefügt werden, so kann der Zeilenanfang durch Wortnummer 0 bzw. Zeichennummer 0 angegeben werden.

Wird bei Korrekturart »Ersetzen« und »Einfügen« der Korrekturbereich durch Wortnummern (ohne zusätzliche Angabe von Zeichennummern) eingegrenzt, so wird am Ende des Korrekturtextes ein Leerzeichen ergänzt; wird der Korrekturtext ans Ende der Zeile (durch »Einfügen nach dem letzten Wort«) angehängt, so wird zuvor am Zeilenende ein Leerzeichen ergänzt.

Wird bei Korrekturart »Ersetzen« und »Einfügen« der Korrekturbereich durch Zeichennummern eingegrenzt und stehen hinter den Korrekturzeichen »=« bzw. »+« keine anderen Zeichen, so wird bei der Ausführung der Korrektur dort ein Leerzeichen eingesetzt.

Beim Einlesen des Textes, beim Einlesen der Korrekturanweisungen und beim Ausgeben des Textes werden am Zeilenende stehende Leerzeichen entfernt.

## Korrekturprotokoll

Das Protokoll weist alle durch die Korrektur vorgenommenen Änderungen nach. Dabei wird jede Zeile, die verändert wurde, in der ursprünglichen und in der korrigierten Fassung untereinander ausgedruckt; zwischen diesen beiden Zeilen werden die vorgenommenen Änderungen wie folgt markiert:

- eine Löschung wird durch »-« unter den gelöschten Zeichen der ursprünglichen (im Protokoll oberen) Zeile markiert;
- eine Einfügung wird durch »+« über den eingefügten Zeichen der korrigierten (im Protokoll unteren) Zeile markiert;
- eine Ersetzung wird wie eine Löschung und anschließende Einfügung markiert.

Treffen in der Zeile mit den Markierungen »-« und »+« zusammen (z. B. beim Protokollieren einer Ersetzung), so wird an dieser Stelle ein »\*« gedruckt.

## Parameter

Jeder Parameter ist einer Parameterart zugeordnet. Sie ist jeweils als römische Zahl in [ ] angegeben. Damit ist festgelegt, in welcher Form die Angaben vom Programm erwartet werden. Die einzelnen Parameterarten sind in der Beschreibung »TUSTEP-Grundlagen« im Kapitel »{}-Parameter« bzw. »<>-Parameter« definiert.

Für manche Parameter gibt es voreingestellte Werte, die angenommen werden, falls die entsprechenden Werte nicht angegeben sind. Diese Voreinstellungen sind in < > angegeben.

### Einstellen der Parameter-Konvention

Ob die Parameter nach der »{}-Parameter-Konvention« oder nach der »<>-Parameter-Konvention« interpretiert werden, kann mit dem Kommando #PARAMETER, {} bzw. #PARAMETER, <> eingestellt werden. Darüber hinaus kann die Konvention auch mit folgendem Parameter eingestellt werden.

- PAR**            Parameter-Konvention einstellen.
- { }    Nachfolgende Parameter nach der »{}-Parameter-Konvention« interpretieren.
  - <>    Nachfolgende Parameter nach der »<>-Parameter-Konvention« interpretieren.

Die Einstellung mit dem Parameter **PAR** hat Vorrang vor der mit dem Kommando #PARAMETER gewählten Einstellung; sie gilt jeweils nur für die nachfolgenden Parameter bis zum nächsten Parameter **PAR** bzw. bis zum Ende der Parameter dieses Programms.

### Auswahl der Daten

Soll die gesamte Datei verarbeitet werden, braucht der folgende Parameter nicht angegeben zu werden.

- BER**            Angabe eines Bereichs (»Seite.Zeile–Seite.Zeile«, falls die Datei im Textmodus nummeriert ist; Zeile–Zeile, falls die Datei im Programmmodus nummeriert ist) oder einer Anfangsstelle (»Seite.Zeile« bzw. »Zeile«), falls nicht die ganze Datei verarbeitet werden soll. [ XI ]

### Angaben zum Protokoll

- DR**            Angaben zur Druckausgabesteuerung. [ I ]
- Es können vier Zahlenwerte angegeben werden:
1. Zahl: Spalten <1>

Anzahl der Spalten, die auf jeder Seite nebeneinander gedruckt werden sollen

2. Zahl: Rand <0>

Anzahl der Leerstellen links der ersten Spalte

3. Zahl: Breite <132>

Anzahl der Zeichen je Spalte

4. Zahl: Zwischenraum <0>

Anzahl der Leerstellen zwischen den Spalten

**DRZ**

Zusätzliche Angaben zur Druckausgabesteuerung. [ I ]

Es können drei Zahlenwerte angegeben werden:

1. Zahl: Kopftext <3>

Anzahl der Zeilen für den Kopftext, einschließlich der Leerzeilen

2. Zahl: Höhe <60>

Anzahl der Zeilen je Reihe, ohne die Zeilen für den Kopf- und Fußtext

3. Zahl: Fußtext <0>

Anzahl der Zeilen für den Fußtext, einschließlich der Leerzeilen

**KT**

Textteile, die als Kopftext oben auf jeder Seite gedruckt werden sollen.

[ II ] <: &Q3 @/ &D2 &U2 &#6:>

In den Textteilen werden folgende Steueranweisungen durch die entsprechenden aktuellen Werte ersetzt:

&Q1 Projektname der QUELL-Datei (ohne Dateiname)

&Q2 Dateiname der QUELL-Datei (ohne Projektname)

&Q3 Projekt- und Dateiname der QUELL-Datei

&D0 Wochentag (z. B. Sonntag)

&D1 Datum xx.xx.xx (z. B. 02.04.96)

&D2 Datum xx. xxx. xxxx (z. B. 2. Apr. 2008)

&D3 Datum xx. xxxxxxxxxxxx xxxx (z. B. 2. April 2008)

&U1 Uhrzeit xx.xx (z. B. 12.00)

&U2 Uhrzeit xx:xx (z. B. 12:00)

&#n Seitennummer mit maximal n (n=1 bis 6) Stellen

Die Seitennummer kann nur einmal eingesetzt werden. Wird für die Seitennummer »- &#n -« (n=1 bis 6) angegeben, so wird die Seitennummer in die Mitte zwischen die Minuszeichen eingesetzt; die Minuszeichen werden bis auf ein Leerzeichen als Zwischenraum nach rechts bzw. links zur Seitennummer hin verschoben.

Werden pro Seite mehrere Reihen (4. Zahlwert des Parameters DRZ) ausgegeben und wird &#n in einer Zeile angegeben, die für jede Reihe

wiederholt wird (4. bzw. 7. Zahlwert des Parameters DRZ), so wird statt der Seitennummer die laufende Nummer der jeweiligen Reihe eingesetzt.

Jeder der Textteile kann durch die Formatieranweisungen »@z« und »@/« in drei Teile gegliedert sein:

linksbündig @z auf Mitte zentriert @/ rechtsbündig

Diese einzelnen Teile werden linksbündig, auf Mitte zentriert und rechtsbündig eingesetzt. Jeder einzelne Teil kann fehlen; dabei kann bei Teil zwei und drei die davor stehende Formatieranweisung ebenfalls entfallen.

Jeder Textteil wird in eine neue Zeile des Kopftextes gedruckt. Bei mehrspaltigem Druck können auch Textteile für die einzelnen Spalten angegeben werden. Dazu gibt es folgende Regelung:

Beginnt ein Textteil mit »\* :«, so wird der Rest des Textteils über jede Spalte in den Kopftext eingetragen. Steht anstelle des Sterns eine Zahl, so wird der Rest des Textteils über die durch die Zahl bezeichnete Spalte eingetragen. Hat die Zahl den Wert 0, so gilt der Rest des Textteils für die ganze Zeile. Beginnt ein Textteil nicht in der beschriebenen Weise, so wird »0 :« angenommen (Normalfall).

Mit einem Textteil, der für eine ganze Zeile des Kopftextes gilt, wird immer eine neue Zeile begonnen. Ein Textteil, der über einer bestimmten Spalte stehen soll, wird in die gleiche Zeile wie der vorangehende Textteil eingetragen, falls diese Zeile nicht schon einen Text für die ganze Zeile oder für diese oder eine weiter rechts stehende Spalte enthält; andernfalls wird mit diesem Textteil eine neue Zeile begonnen.

**FT** Textteile, die als Fußtext unten auf jeder Seite gedruckt werden sollen. [ II ] <>

Es gelten die gleichen Regelungen wie für den Kopftext (Parameter KT). Die Seitennummer kann jedoch nicht in den Fußtext eingesetzt werden, wenn sie schon in den Kopftext eingesetzt worden ist.

**DRT** Druckertyp, für den die Daten aufbereitet werden. [ XI ]

Dieser Parameter ist obligat, wenn ein Protokoll erstellt wird.

Hinweis: Mit dem Kommando #LISTE, DRUCKERTYPEN werden die definierten Druckertypen aufgelistet.

## Angaben zu den Ausgabesätzen

**NR** Angaben zur Nummerierung der Ausgabesätze. [ I ]

Es können drei Zahlenwerte angegeben werden:

1. Zahl: Seitennummer, mit der bei der Ausgabe begonnen werden soll bzw. 999999, falls mit der nächsten freien Seite begonnen werden soll. <MODUS=-STD-: 999999; MODUS+=: 1>

- 2. Zahl: Maximale Anzahl der Sätze je Seite. <MODUS=-STD-: 1000000; MODUS=+: 60>
- 3. Zahl: Schrittweite der Nummerierung, wenn eine neue Satznummer vergeben werden muss. <MODUS=-STD-: 10; MODUS=+: 1000>

**SL** Angaben zur Satzlänge der Ausgabesätze. [ I ]

Es können zwei Zahlenwerte angegeben werden:

- 1. Zahl: Maximallänge für Sätze, in die diejenigen Zeilen unterteilt werden sollen, die länger sind, als mit der zweiten Zahl dieses Parameters angegeben ist. <100>
- 2. Zahl: Maximallänge für Zeilen, die nicht unterteilt werden sollen. <120 bzw. Wert der 1. Zahl, falls diese größer als 120 ist>

Vor der Ausgabe wird eine Zeile, die mehr Zeichen enthält, als mit dem 2. Zahlenwert angegeben ist, in mehrere Ausgabesätze unterteilt. Die Zeile wird so unterteilt, dass die Ausgabesätze maximal so lang sind, wie mit dem 1. Zahlenwert angegeben ist. Als Trennstelle wird ein Leerzeichen gewählt, vor dem kein »-« steht, das mit einem Silbentrennzeichen verwechselt werden kann. Falls innerhalb der mit dem 1. Zahlenwert angegebenen Anzahl von Zeichen keine solche Trennstelle vorhanden ist, wird die nächstmögliche Trennstelle gesucht.

**Alphabetisches Verzeichnis der Parameter**

<b>BER</b>	Auswählen eines Bereichs aus der QUELL-Datei . . . . .	925
<b>DR</b>	Druckausgabesteuerung . . . . .	925
<b>DRT</b>	Druckertyp . . . . .	927
<b>DRZ</b>	Druckausgabesteuerung - zusätzliche Angaben . . . . .	926
<b>FT</b>	Fußtext . . . . .	927
<b>KT</b>	Kopf text . . . . .	926
<b>NR</b>	Nummerierung der Ausgabesätze . . . . .	927
<b>PAR</b>	Parameter-Konvention . . . . .	925
<b>SL</b>	Satzlänge der Ausgabesätze . . . . .	928

\*\*\*\*\*



**#KOPIERE**

KOPIERE

---

Kommando . . . . .	932
Leistung . . . . .	933
Parameter . . . . .	934
Einstellen der Parameter-Konvention . . . . .	934
– INITIALISIERUNG . . . . .	935
Definieren der Sonderwahlschalter . . . . .	935
Bestimmen der Wahlschalter-Grundstellung . . . . .	936
Holen der Wahlschalter von der Kommandoebene . . . . .	936
Bestimmen des Anfangswertes der laufenden Nummer . . . . .	936
Bestimmen der Anfangswerte der Merk-Vergleichstexte . . . . .	937
Bestimmen der Anfangswerte der Ersetzungstexte . . . . .	937
Bestimmen der Anfangswerte der Merktexte . . . . .	937
Bestimmen der Anfangswerte der Variablen . . . . .	937
Umdefinieren des Programmablaufs . . . . .	937
Bestimmen des Schwellenwerts für Endlosschleifen . . . . .	938
Ausgeben eines Anfangstextes . . . . .	939
– PROGRAMMTEIL 0 (Eingabe) . . . . .	939
Auswahl der Daten . . . . .	939
Zusammenfassen mehrerer Sätze zu einer Texteinheit . . . . .	940
Austauschen von Zeichenfolgen . . . . .	943
Unterteilen des Grundtextes . . . . .	943
– PROGRAMMTEIL 1 (Abfrage + Vergleich) . . . . .	944
Verzweigen in Abhängigkeit von Wahlschaltern . . . . .	944
Verzweigen in Abhängigkeit von einer Variablen/Zeichenfolge . . . . .	944
Verzweigen in Abhängigkeit von einer Kennung . . . . .	945
Erstellen des Vergleichstextes . . . . .	945
Verzweigen in Abhängigkeit von e. Zeichenfolgenhäufigkeit . . . . .	946
Auswählen der Texteinheiten . . . . .	946
– PROGRAMMTEIL 2 (Verarbeitung) . . . . .	953
Setzen und Löschen von Wahlschaltern . . . . .	953
Bestimmen des Arbeitstextes . . . . .	954
Austauschen von Textteilen . . . . .	956
Umdrehen von Textteilen . . . . .	957
Berechnen der Druckpositionen . . . . .	958
Lesen von Zahlenwerten aus dem Arbeitstext . . . . .	958
Ausführen von Rechenanweisungen . . . . .	959
Einsetzen von Zahlenwerten in den Arbeitstext . . . . .	959
Austauschen von Zeichenfolgen . . . . .	961
Austauschen von Zeichenfolgen mit Bedingungen . . . . .	962
Einsetzen von zuvor definierten Textteilen . . . . .	963
Einsetzen einer laufenden Nummer . . . . .	965
Einsetzen der Seiten-Zeilen-Nummer . . . . .	966
Definieren von Positionen (Spalten) . . . . .	966

– Programmteil 3 (Ausgabe) . . . . .	967
Ausdrucken von Meldungen . . . . .	967
Bestimmen der ZIEL-Datei . . . . .	968
Bestimmen von Zeilenanfang und -ende . . . . .	968
Bestimmen von Seitenanfang und -ende . . . . .	969
Einfügen von Leerzeilen . . . . .	969
Austauschen von Zeichenfolgen . . . . .	970
Unterdrücken von Leerzeichen am Satzanfang und -ende . . . . .	970
Unterdrücken von Leerzeilen . . . . .	970
Bestimmen der Satzlänge . . . . .	970
Bestimmen der Nummerierung der Sätze . . . . .	971
– PROGRAMMTEIL 4 . . . . .	972
Umdefinieren des Grundtextes . . . . .	972
– PROGRAMMTEIL 5 . . . . .	972
Definieren von Merk-Vergleichstexten . . . . .	972
– PROGRAMMTEIL 6 . . . . .	972
Definieren von Ersetzungstexten . . . . .	972
– PROGRAMMTEIL 7 . . . . .	972
Ersetzen/Ergänzen des Merktextes durch den Arbeitstext . . . . .	972
– PROGRAMMTEIL 8 . . . . .	973
Austauschen von Merktext und Arbeitstext . . . . .	973
– PROGRAMMTEIL 9 . . . . .	974
Ersetzen/Ergänzen des Arbeitstextes durch den Merktext . . . . .	974
– ABSCHLUSS . . . . .	975
Ausgeben eines Endetextes und einer Endmeldung . . . . .	975
Retten der Endwerte von Variablen . . . . .	976
Retten der Wahlschalter auf die Kommandoebene . . . . .	976
– Alphabetisches Verzeichnis der Parameter . . . . .	977
Prüfziffern . . . . .	981
Vergleichsalgorithmus . . . . .	982
Rechenanweisungen . . . . .	983
Variablen . . . . .	983
Funktionen . . . . .	986
Arithmetischer Ausdruck . . . . .	989
Vergleichsbedingung . . . . .	989
Logischer Ausdruck . . . . .	990
Wertzuzuweisung . . . . .	990
Sprunganweisung . . . . .	991
Bedingungsanweisung . . . . .	991
Schleifenanweisung . . . . .	992
Logischer Programmaufbau . . . . .	994

## Kommando

#KOPIERE

QUELLE	= <code>datei</code>	Name der Datei mit den Daten, die kopiert werden sollen.
	= <code>-STD-</code>	Die Daten, die kopiert werden sollen, stehen in der Standard-Text-Datei.
ZIEL	= <code>datei</code>	Name der Datei für die kopierten Daten. Es können bis zu zehn Dateinamen angegeben werden; sie müssen durch Apostroph getrennt sein.
	= <code>-STD-</code>	Die Daten in die Standard-Text-Datei kopieren.
MODUS	= <code>-STD-</code>	* Satznummern nur ändern, wenn sie durch Parameter geändert werden. Nicht aufsteigende Satznummern automatisch ändern, damit sie aufsteigend sind.
	= <code>+</code>	Sätze neu nummerieren.
	= <code>-</code>	Satznummern nur ändern, wenn sie durch Parameter geändert werden. Nicht aufsteigende Satznummern zulassen.
	= <code>S</code>	DATEN-Datei enthält Sortiereinheiten; Sätze neu nummerieren.
	= <code>R</code>	QUELL-Datei satzweise rückwärts kopieren; Satznummern nur ändern, wenn sie durch Parameter geändert werden.
LOESCHEN	= <code>-</code>	* Daten in der ZIEL- und in der PROTOKOLL-Datei nicht löschen.
	= <code>+</code>	Daten in der ZIEL- und in der PROTOKOLL-Datei zuerst löschen.
PARAMETER	= <code>-</code>	* Keine Parameter.
	= <code>datei</code>	Name der Datei mit den Parametern.
	= <code>*</code>	Die Parameter folgen auf das Kommando und sind mit *EOF abgeschlossen.
DATEN	= <code>-</code>	* Daten sind vollständig in der QUELL-Datei.
	= <code>datei</code>	Daten teilweise in der angegebenen Datei.
	= <code>-STD-</code>	Daten teilweise in der Standard-Daten-Datei.
PROTOKOLL	= <code>-</code>	* Kein Testprotokoll erstellen.
	= <code>+</code>	Testprotokoll ins Ablaufprotokoll ausgeben.

---

= -STD-	Testprotokoll in die Standard-Protokoll-Datei ausgegeben.
= datei	Name der Datei für das Testprotokoll.

## Leistung

Mit diesem Programm können Dateien nicht nur kopiert, sondern gleichzeitig auch in vielfältiger Weise bearbeitet werden.

Über Parameter kann u. a. gesteuert werden

- die Überprüfung der Daten
- die Auswahl der Daten
- die Umstellung und Ergänzung von einzelnen Textteilen
- die Ersetzung von Zeichenfolgen
- die Bearbeitung von Kalenderdaten
- das Rechnen mit im Text enthaltenen Zahlen
- die Form der Ausgabe

## Parameter

Jeder Parameter ist einer Parameterart zugeordnet. Sie ist jeweils als römische Zahl in [ ] angegeben. Damit ist festgelegt, in welcher Form die Angaben vom Programm erwartet werden. Die einzelnen Parameterarten sind in der Beschreibung »TUSTEP-Grundlagen« im Kapitel »{}-Parameter« bzw. »<>-Parameter« definiert.

Für manche Parameter gibt es voreingestellte Werte, die angenommen werden, falls die entsprechenden Werte nicht angegeben sind. Diese Voreinstellungen sind in < > angegeben.

Außer den im folgenden beschriebenen Parametern sind auch Parameter zur Definition von Zeichen- und Stringgruppen möglich. [ v ]

Mit bestimmten Parametern können über Anfangs- und/oder Endekennungen sowie über Kennungen, die als öffnende bzw. schließende Klammern dienen, Textteile für die weitere Bearbeitung ausgewählt werden. Die Funktionsweise dieser Parameter ist auch in den »TUSTEP-Grundlagen« am Ende des Kapitels »Parameter« beschrieben.

Bei Parametern, die für jeden Durchgang eigens angegeben werden können, muss die Nummer des Durchgangs rechtsbündig in den Spalten 6 und 7 angegeben werden; fehlt die Angabe bei einem solchen Parameter, so wird Durchgang 1 angenommen.

Die Parameter werden, falls nicht ausdrücklich etwas anderes angegeben ist, in der Reihenfolge abgearbeitet, in der sie im folgenden beschrieben sind. Bei mehreren Durchgängen bestimmt die Durchgangsnummer vorrangig die Reihenfolge, wenn der Programmablauf nicht durch entsprechende Parameter anders definiert wurde. Die Reihenfolge, in der die Parameter bei Aufruf des Programms angegeben werden, ist also belanglos. Werden für die Angaben eines Parameters Folgezeilen benutzt, so müssen diese jedoch in der entsprechenden Reihenfolge unmittelbar nacheinander angegeben werden. Um Fehler zu vermeiden, empfiehlt es sich, die Parameter in der Reihenfolge anzugeben, in der sie abgearbeitet werden.

### Einstellen der Parameter-Konvention

Ob die Parameter nach der »{}-Parameter-Konvention« oder nach der »<>-Parameter-Konvention« interpretiert werden, kann mit dem Kommando #PARAMETER, {} bzw. #PARAMETER, <> eingestellt werden. Darüber hinaus kann die Konvention auch mit folgendem Parameter eingestellt werden.

**PAR** Parameter-Konvention einstellen.

{ } Nachfolgende Parameter nach der »{}-Parameter-Konvention« interpretieren.

<> Nachfolgende Parameter nach der »<>-Parameter-Konvention« interpretieren.

Die Einstellung mit dem Parameter PAR hat Vorrang vor der mit dem Kommando #PARAMETER gewählten Einstellung; sie gilt jeweils nur für die nachfolgenden Parameter bis zum nächsten Parameter PAR bzw. bis zum Ende der Parameter dieses Programms.

## INITIALISIERUNG

### Definieren der Sonderwahlschalter

**SWS** Sonderwahlschalter. [ I ]

Es können fünf Zahlenwerte (Nummern der Wahlschalter 1 bis 16) angegeben werden. Die Angabe 0 (Null, Voreinstellung) bedeutet, dass für den jeweiligen Zweck kein Wahlschalter verwendet werden soll.

1. Zahl: Wahlschalter für Eingabedaten <0>

Ist kein erster und kein fünfter Sonderwahlschalter angegeben, so wird (nach Programmteil 9 gesprungen und) das Programm beendet, sobald die Eingabedaten erschöpft sind.

Ist der erste Sonderwahlschalter angegeben, so wird er bei jeder Eingabe gelöscht. Sind die Eingabedaten erschöpft, wird er gesetzt und fortgefahren, als wäre eine leere Texteinheit eingelesen worden. Beim Versuch, eine weitere Texteinheit einzulesen, wird das Programm mit einer entsprechenden Fehlermeldung beendet. Ist jedoch zu diesem Zeitpunkt der fünfte Sonderwahlschalter gesetzt, so werden die Daten nochmals eingelesen.

2. Zahl: Wahlschalter für Grundtext-Unterteilung <0>

Der angegebene Wahlschalter wird im Programmteil 0 vor jeder Grundtext-Unterteilung gelöscht und gesetzt, wenn der Grundtext erschöpft ist.

3. Zahl: Wahlschalter für Testprotokoll <0>

Die Ausgabe des Testprotokolls beim Durchlauf der einzelnen Programmteile erfolgt nur, wenn der angegebene Wahlschalter gesetzt ist.

4. Zahl: Wahlschalter für Grundtext-Wiederholung <0>

Wenn beim Sprung in den Programmteil 0 der angegebene Wahlschalter gesetzt ist und der Parameter GTU angegeben ist, wird nochmals der gleiche Teil des Grundtextes zum Arbeitstext wie beim vorhergehenden Mal, als der Grundtext der gleichen Texteinheit auf Grund des Parameters GTU unterteilt wurde. Der Wahlschalter wird dabei automatisch wieder gelöscht. Wenn der Grundtext soeben von der Datei eingelesen wurde, bleibt der Wahlschalter unberücksichtigt.

5. Zahl: Wahlschalter für doppelten Durchlauf <0>

Ist kein erster und kein fünfter Sonderwahlschalter angegeben, so wird (nach Programmteil 9 gesprungen und) das Programm beendet, sobald die Eingabedaten erschöpft sind.

Ist der fünfte Sonderwahlschalter angegeben und gesetzt, wenn die Eingabedaten erschöpft sind, so wird er gelöscht und die Daten werden nochmals eingelesen; andernfalls wird das Programm beendet.

Ist jedoch auch der erste Sonderwahlschalter angegeben, so wird verfahren wie oben beim ersten Sonderwahlschalter beschrieben. Erst beim Versuch, eine weitere Texteinheit einzulesen, wird der fünfte Sonderwahlschalter geprüft und entweder der Wahlschalter gelöscht und die Daten nochmals eingelesen oder das Programm mit einer entsprechenden Fehlermeldung beendet.

Hinweis: Damit kann z. B. im ersten Durchlauf (ohne dabei die Daten in die ZIEL-Datei auszugeben) durch Aufaddieren einzelner im Text stehender Zahlenwerte die Gesamtzahl festgestellt werden, um im zweiten Durchlauf für die einzelnen Zahlenwerte die entsprechenden Prozentangaben auszurechnen und im Text zu ergänzen.

## Bestimmen der Wahlschalter-Grundstellung

**WSG** Wahlschalter, die gesetzt werden sollen. [ 1 ]

Es können bis zu 16 Zahlenwerte (Nummern der Wahlschalter 1 bis 16) angegeben werden. Wahlschalter, die nicht angegeben sind, werden gelöscht.

## Holen der Wahlschalter von der Kommandoebene

**WSH** Wahlschalter, deren Grundstellung von den Wahlschaltern der Kommandoebene (das sind die Wahlschalter, die mit dem Kommando #WAHLSCHALTER gesetzt bzw. gelöscht werden können) übernommen werden soll (Wahlschalter holen). [ 1 ]

Es können bis zu 7 Zahlenwerte (Nummern der Wahlschalter 1 bis 7) angegeben werden. Wahlschalter, die nicht angegeben sind, werden entsprechend den Angaben zum Parameter WSG gesetzt bzw. gelöscht.

## Bestimmen des Anfangswertes der laufenden Nummer

**LNB** Angabe, falls die laufende Nummer, die durch Angabe des Parameters LNR eingesetzt werden kann, nicht mit 1 beginnen soll. [ 1 ]

Es können zwei Zahlenwerte angegeben werden:

1. Zahl: Wert, der auf die Ausgabennummer (= 1 + Anzahl der bereits ausgegebenen Texteinheiten), aufaddiert werden soll. <0>



2. Zahl: Wert, der auf die laufende Nummer aufaddiert werden soll.  
<0>

### Bestimmen der Anfangswerte der Merk-Vergleichstexte

**VTV** Vergleichstext-Vorbelegung. Der hier angegebene Textteil wird als Merk-Vergleichstext für den jeweiligen Durchgang, für den dieser Parameter angegeben ist, beim Start des Programms eingespeichert. [ II ]

### Bestimmen der Anfangswerte der Ersetzungstexte

**ETV** Ersetzungstext-Vorbelegung. Der hier angegebene Textteil wird als Ersetzungstext für den jeweiligen Durchgang, für den dieser Parameter angegeben ist, beim Start des Programms eingespeichert. [ II ]

### Bestimmen der Anfangswerte der Merkttexte

**MTV** Merkttext-Vorbelegung. Die hier angegebenen Textteile werden als Merkttexte beim Start des Programms eingespeichert. Es können bis zu 10 Merkttexte belegt werden; sie sind von 0 bis 9 durchnummeriert.  
[ II ]

### Bestimmen der Anfangswerte der Variablen

**HVH** Name einer TUSTEP-Variablen, die Zahlenwerte enthält, mit denen die H-Variablen vorbesetzt werden. [ XI ]

Die angegebene TUSTEP-Variable darf bis zu 10 Zahlenwerte enthalten. Sie werden den Variablen H0, H1, ..., H9 zugewiesen. Die einzelnen Zahlenwerte müssen durch Leerzeichen oder durch Apostroph getrennt sein.

**R** Rechenanweisungen zur Vorbesetzung der Variablen [ XI ]  
Beschreibung siehe »Rechenanweisungen« Seite 983 ff.

### Umdefinieren des Programmablaufs

Der logische Programmablauf ist in der Tabelle »Logischer Programmaufbau« auf Seite 994 schematisch dargestellt. Zum Umdefinieren des Programmablaufs können entweder die Parameter SPW, SPN, SPJ und SP<sub>n</sub> (n=0, 2, 3, . . . , 8, 9) oder der Parameter SPR verwendet werden. Wenn der Parameter SPR verwendet wird, darf keiner der anderen verwendet werden.

- SPW** Sprungtabelle für Sprung, falls ein unter *WS+* angegebener Wahlschalter gesetzt ist oder falls ein unter *WS-* angegebener nicht gesetzt ist. [ 1 ] <siehe »Logischer Programmaufbau»>
- SPN** Sprungtabelle für Sprung, falls die Vergleichsbedingungen nicht alle erfüllt sind (nein, nicht erfüllt). [ 1 ] <siehe »Logischer Programmaufbau»>
- SPJ** Sprungtabelle für Sprung, falls alle Vergleichsbedingungen erfüllt sind (ja, erfüllt). [ 1 ] <siehe »Logischer Programmaufbau»>
- SPn** Sprungtabelle für Sprung nach Beendigung des Programmteils *n* (*n* = 0, 2, 3, ..., 8, 9). [ 1 ] <siehe »Logischer Programmaufbau»>
- SPR** Sprungtabelle für alle vorgesehenen Sprünge eines Durchgangs. [ 1 ] <siehe »Logischer Programmaufbau»>

Mit diesem Parameter können im *n*-ten Durchgang (Angabe der Zahl *n* rechtsbündig in Spalte 6 und 7 des Parameters) die Werte angegeben werden, die mit den Parametern *SP0*, *SPW*, *SPN*, *SPJ*, *SP2*, *SP3*, ..., *SP9* jeweils als *n*-ter Wert angegeben werden könnten. Dabei ist auf die Reihenfolge und die Vollständigkeit zu achten.

## Bestimmen des Schwellenwerts für Endlosschleifen

Durch die Umdefinition des Programmablaufs (s. o.) kann es vorkommen, dass das Programm in eine Endlosschleife gerät. Damit eine Endlosschleife erkannt und abgebrochen werden kann, wird beim Einlesen einer Texteinheit in Programmteil 0 ein Zähler auf Null gesetzt und dann in jedem weiteren Programmteil, der durchlaufen wird, um 1 erhöht. Erreicht dieser Zähler einen bestimmten Schwellenwert, wird das Programm mit einer entsprechenden Fehlermeldung abgebrochen. Der Schwellenwert wird vom Programm auf Grund der jeweils angegebenen Parameter geschätzt. Dieser Wert reicht im Regelfall aus. In Spezialfällen kann es jedoch vorkommen, dass dieser Wert zu klein ist. In diesen Fällen kann mit dem folgenden Parameter der Schwellenwert höher gesetzt werden.

- ELS** Anzahl der Programmteile, die für jede eingelesene Texteinheit durchlaufen werden dürfen. [ 1 ]

Werden mehr Programmteile durchlaufen als angegeben, wird eine Endlosschleife vermutet und das Programm abgebrochen.

Hinweis: Erfahrungsgemäß reicht der vom Programm errechnete Wert (die Voreinstellung) nur in ganz seltenen Fällen nicht aus. Die Ursache für einen Programmabbruch wegen einer Endlosschleife ist meist entweder eine fehlerhafte Umdefinition des Programmablaufs oder ein »Datenfehler«, also Daten in einem in den Parametern nicht vorgesehenen Format.

## Ausgeben eines Anfangstextes

**z** Text, der in die ZIEL-Datei ausgegeben werden soll. Bei jedem Trennzeichen wird ein neuer Satz begonnen. [ II ]

Ist mehr als eine ZIEL-Datei vorhanden, wird die erste Zeichenfolge in die erste ZIEL-Datei, die zweite Zeichenfolge in die zweite ZIEL-Datei, usw., ausgegeben. Leere Zeichenfolgen werden nicht in die entsprechenden ZIEL-Dateien ausgegeben. Sind mehr Zeichenfolgen angegeben als ZIEL-Dateien vorhanden sind, werden die weiteren Zeichenfolgen reihum wieder in die erste, zweite, usw. ausgegeben.

**PROGRAMMTEIL 0** (Eingabe, keine Durchgangsangabe):

## Auswahl der Daten

Soll die gesamte Datei verarbeitet werden, braucht keiner der folgenden Parameter angegeben zu werden.

**BER** Angabe eines Bereichs (»Seite.Zeile–Seite.Zeile«, falls die Datei im Textmodus nummeriert ist; Zeile–Zeile, falls die Datei im Programmmodus nummeriert ist) oder einer Anfangsstelle (»Seite.Zeile« bzw. »Zeile«), falls nicht die ganze Datei verarbeitet werden soll. [ XI ]

Soll ein Segment einer Segment-Datei verarbeitet werden, kann anstelle des Bereichs der Name des Segments angegeben werden.

Dieser Parameter ist nur zugelassen, wenn die Satznummern in der Datei alle aufsteigend sind; außerdem schließen sich die Parameter BER, NR+ und NR– gegenseitig aus.

**NR+** Seiten-Zeilen-Nummern (auch Bereiche) der Sätze, die berücksichtigt werden sollen; die einzelnen Angaben müssen durch Apostroph getrennt werden. [ XI ]

Die Sätze werden in der Reihenfolge eingelesen, in der die Seiten-Zeilen-Nummern (bzw. Bereiche) angegeben sind. Mehrfachnennungen und Überlappungen sind möglich.

Dieser Parameter ist nur zugelassen, wenn die Satznummern in der Datei alle aufsteigend sind; außerdem schließen sich die Parameter BER, NR+ und NR– gegenseitig aus.

**NR–** Seiten-Zeilen-Nummern (auch Bereiche) der Sätze, die übergangen werden sollen; die einzelnen Angaben müssen durch Apostroph getrennt werden. [ XI ]

Die Seiten-Zeilen-Nummern (bzw. Bereiche) müssen in aufsteigender Reihenfolge angegeben werden. Bei Angabe von Bereichen dürfen diese sich nicht überlappen.

Dieser Parameter ist nur zugelassen, wenn die Satznummern in der Datei alle aufsteigend sind; außerdem schließen sich die Parameter BER, NR+ und NR- gegenseitig aus.

**MAX** Angaben für Testzwecke, wieviele Texteinheiten maximal eingelesen bzw. ausgegeben werden sollen. [ 1 ]

Es können zwei Zahlenwerte angegeben werden:

1. Zahl: einzulesende Texteinheiten <999999999>
2. Zahl: auszugebende Texteinheiten <999999999>

## Zusammenfassen mehrerer Sätze zu einer Texteinheit

Falls jeder Eingabesatz schon eine vollständige Texteinheit enthält, sollte keiner der folgenden Parameter angegeben werden. Andernfalls können mit den Parametern ANR, ALZ, AA und/oder AE jeweils mehrere Sätze zu einer Texteinheit zusammengefasst werden.

Ist einer der Parameter ANR, ALZ, AA und AE angegeben, werden vor der Auswertung dieser Parameter evtl. am Anfang und am Ende des Eingabesatzes stehende Leerzeichen entfernt.

Beim Zusammenfassen wird zwischen den einzelnen Eingabesätzen je ein Leerzeichen ergänzt, jedoch nicht an den Stellen, an denen eine Silbentrennung aufgehoben wird (siehe Parameter STR).

**ANR** Angaben, ob aufeinander folgende Sätze, deren Satznummern teilweise oder vollständig übereinstimmen, zu einer Texteinheit zusammengefasst werden sollen. (Abschnitt auf Grund der Nummerierung). [ 1 ] <0>

Es kann ein Zahlenwert angegeben werden:

- 0 = kein Zusammenfassen von Sätzen auf Grund der Satznummer
- 1 = alle aufeinander folgenden Sätze mit der gleichen Seitennummer zu einer Texteinheit zusammenfassen.
- 2 = alle aufeinander folgenden Sätze mit der gleichen Seiten- und der gleichen Zeilennummer (ohne Berücksichtigung der Unterscheidungsnummer) zu einer Texteinheit zusammenfassen.
- 3 = alle aufeinander folgenden Sätze mit der gleichen Satznummer zu einer Texteinheit zusammenfassen.

Ist 0 angegeben (Voreinstellung), so wird die Zusammenfassung nur auf Grund der Parameter ALZ, AA und AE vorgenommen; ist einer der Werte 1 bis 3 angegeben, so ist eine weitere Unterteilung der so gebildeten Texteinheiten auf Grund der genannten Parameter möglich.

**ALZ** Angaben, ob Leerzeilen als Kennzeichen für den Anfang bzw. für das Ende einer Texteinheit dienen sollen. [ 1 ]

Es können zwei Zahlenwerte angegeben werden:

## 1. Zahl: Anfang einer Texteinheit &lt;0&gt;

- 0 = Leerzeilen sind kein Kennzeichen für den Anfang einer Texteinheit.
- 1 = Leerzeilen kennzeichnen den Anfang einer Texteinheit. Folgen mehrere Leerzeilen unmittelbar aufeinander, so beginnt mit jeder einzelnen Leerzeile eine Texteinheit.
- 2 = Leerzeilen kennzeichnen den Anfang einer Texteinheit. Folgen mehrere Leerzeilen unmittelbar aufeinander, so beginnt nur mit der ersten Leerzeile eine Texteinheit.

## 2. Zahl: Ende einer Texteinheit &lt;0&gt;

- 0 = Leerzeilen sind kein Kennzeichen für das Ende einer Texteinheit.
- 1 = Leerzeilen kennzeichnen das Ende einer Texteinheit. Folgen mehrere Leerzeilen unmittelbar aufeinander, so endet mit jeder einzelnen Leerzeile eine Texteinheit.
- 2 = Leerzeilen kennzeichnen das Ende einer Texteinheit. Folgen mehrere Leerzeilen unmittelbar aufeinander, so endet nur mit der ersten Leerzeile eine Texteinheit.

Ist jeweils 0 angegeben (Voreinstellung), so wird die Zusammenfassung nur auf Grund der Parameter ANR, AA und AE vorgenommen; andernfalls ist eine weitere Unterteilung der so gebildeten Texteinheiten auf Grund der genannten Parameter möglich.

**AA**

Zeichenfolgen, die am Anfang des Eingabesatzes den Anfang einer Texteinheit (Abschnittsanfang) kennzeichnen. [ VIII-a ]

Falls mit dem Parameter AAZ nichts anderes angegeben ist, werden vor der Überprüfung, ob ein Satz mit einer der angegebenen Zeichenfolgen beginnt, führende Leerzeichen entfernt.

Beginnen mit dem Parameter AA angegebene Zeichenfolgen mit Leerzeichen, so können diese nur dann einen Abschnittsanfang kennzeichnen, wenn mit dem Parameter AAZ angegeben wird, dass die führenden Leerzeichen erst nach der Überprüfung entfernt werden sollen; andernfalls ist die Angabe solcher Zeichenfolgen wirkungslos.

**AAZ**

Zusatzangaben zum Parameter AA. [ 1 ] <0>

Es kann ein Zahlenwert angegeben werden:

- 0 = Leerzeichen am Anfang von Eingabesätzen vor der Auswertung des Parameters AA entfernen.
- 1 = Leerzeichen am Anfang von Eingabesätzen nach der Auswertung des Parameters AA entfernen.
- 2 = wie 0, jedoch zusätzlich auch die Zeichenfolgen »#[09]« und »#[0009]« entfernen.

**AE**

Zeichenfolgen, die am Ende des Eingabesatzes das Ende einer Texteinheit (Abschnittsende) kennzeichnen. [ VIII-b ]

Falls mit dem Parameter `AEZ` nichts anderes angegeben ist, werden vor der Überprüfung, ob ein Satz mit einer der angegebenen Zeichenfolgen endet, abschließende Leerzeichen entfernt.

Enden mit dem Parameter `AE` angegebene Zeichenfolgen mit Leerzeichen, so können diese nur dann ein Abschnittsende kennzeichnen, wenn mit dem Parameter `AEZ` angegeben wird, dass die abschließenden Leerzeichen erst nach der Überprüfung entfernt werden sollen; andernfalls ist die Angabe solcher Zeichenfolgen wirkungslos.

**AEZ** Zusatzangaben zum Parameter `AE`. [ 1 ] <0>

Es kann ein Zahlenwert angegeben werden:

- 0 = Leerzeichen am Ende von Eingabesätzen vor der Auswertung des Parameters `AE` entfernen.
- 1 = Leerzeichen am Ende von Eingabesätzen nach der Auswertung des Parameters `AE` entfernen.

**STR** Angabe zur Silbentrennung. [ 1 ] <0>

Es kann ein Zahlenwert angegeben werden:

- 0 = Silbentrennungen nicht aufheben.
- 1 = Silbentrennung bei der Eingabe aufheben; dabei wird angenommen, dass im nachfolgenden Eingabesatz alle Zeichen bis zum ersten Leerzeichen zum getrennten Wort gehören.
- 2 = wie 1, jedoch Silbentrennung bei der Eingabe nur aufheben, falls entweder der letzte Buchstabe vor und der erste Buchstabe nach dem Silbentrennzeichen (im nachfolgenden Eingabesatz) Kleinbuchstaben sind oder der letzte Buchstabe vor und die ersten beiden Buchstaben nach dem Silbentrennzeichen (im nachfolgenden Eingabesatz) Großbuchstaben sind.
- 3 = wie 2, jedoch wird angenommen, dass im nachfolgenden Eingabesatz alle Zeichen bis zum ersten Leerzeichen bzw. bis zur ersten öffnenden spitzen Klammer zum getrennten Wort gehören.
- 4 = wie 2, jedoch wird angenommen, dass im nachfolgenden Eingabesatz alle Zeichen bis zum ersten Leerzeichen, das außerhalb von spitzen Klammern steht, zum getrennten Wort gehören.

Falls angegeben ist, dass die Silbentrennung bei der Eingabe aufgehoben werden soll, werden in allen Eingabesätzen führende und abschließende Leerzeichen entfernt.

Sollen bei der Eingabe Zeichenfolgen ersetzt werden (Parameter `X`), so wird nach dem Ersetzen festgestellt, ob der Satz mit einem Silbentrennzeichen endet.

Als Silbentrennzeichen gilt ein »-«, das (nachdem abschließende Leerzeichen entfernt wurden) als letztes Zeichen in einem Eingabesatz steht, wenn das zweitletzte Zeichen ebenfalls ein »-« ist, oder wenn das zweitletzte Zeichen ein Buchstabe und das drittletzte Zeichen kein Steuerzeichen (`$`, `&`, `@`, `\`, `_`, `#`, `%`) ist.

Beim Zusammenfassen mehrerer Eingabesätze zu einer Texteinheit werden Silbentrennungen nur innerhalb einer Texteinheit aufgehoben. Steht ein Silbentrennzeichen am Ende des letzten Eingabesatzes einer Texteinheit, wird an dieser Stelle die Silbentrennung nicht aufgehoben.

Beim Aufheben der Silbentrennung wird ein getrenntes »ck«, das als »k-« und »k« geschrieben ist, nicht wieder zu »ck«.

**STE** Zeichenfolge, die beim Aufheben einer Silbentrennung anstelle des Silbentrennzeichens eingefügt werden soll. [ II ]

## Austauschen von Zeichenfolgen

**x** Zeichenfolgenpaare (und Ausnahmezeichenfolgen), deren jeweils erste Zeichenfolge durch die jeweils zweite Zeichenfolge bei der Eingabe ersetzt werden soll. [ X ]

Das Ersetzen erfolgt in jedem einzelnen Eingabesatz, auch wenn mehrere Sätze zu einer Texteinheit zusammengefasst werden (vgl. Parameter ANR, AA und AE).

Vor dem Ersetzen werden evtl. am Anfang und am Ende des Satzes stehende Leerzeichen entfernt und dann am Anfang und am Ende je ein Leerzeichen ergänzt. Nach dem Ersetzen werden nochmals evtl. am Anfang und am Ende des Satzes stehende Leerzeichen entfernt.

Die Überprüfung, ob ein Satz mit einer zum Parameter AA angegebenen Zeichenfolge beginnt bzw. mit einer zum Parameter AE angegebenen endet, erfolgt vor dem Ersetzen.

Soll die Silbentrennung aufgehoben werden (Parameter STR), so wird nach dem Ersetzen festgestellt, ob der Satz mit einem Silbentrennzeichen endet.

**PROGRAMMTEIL 0** (Grundtext ==> Arbeitstext):

## Unterteilen des Grundtextes

Soll nicht der gesamte Grundtext, sondern jeweils nur der nächste Teil des Grundtextes in den Arbeitstext übertragen werden, so kann mit dem folgenden Parameter angegeben werden, bei welchen Zeichenfolgen der Grundtext für diesen Zweck unterteilt werden soll. Die Zeichenfolgen gehören jeweils zum nachfolgenden Teil des Textes.

**GTU** Zeichenfolgen, die die Stellen kennzeichnen, an denen der Grundtext unterteilt werden soll. [ IX ]

Ist mit dem zweiten Zahlenwert des Parameters *SWS* kein Wahlschalter als Sonderwahlschalter bestimmt worden, gilt folgendes: Bei jeder Ausführung des Parameters *GTU* wird der nächste Teil des Grundtextes in den Arbeitstext übertragen. Sind schon alle Teile in den Arbeitstext übertragen worden, erfolgt automatisch ein Sprung in den Programmteil 0.

Ist mit dem zweiten Zahlenwert des Parameters *SWS* ein Wahlschalter als Sonderwahlschalter bestimmt worden, gilt folgendes: Bei jeder Ausführung des Parameters *GTU* wird der nächste Teil des Grundtextes in den Arbeitstext übertragen und der Sonderwahlschalter gelöscht. Sind schon alle Teile in den Arbeitstext übernommen worden, wird eine leere Zeichenfolge in den Arbeitstext gespeichert und der Sonderwahlschalter gesetzt.

**GTZ** Zusatzangaben zum Parameter *GTU*. [ I ] <0>

Es kann ein Zahlenwert angegeben werden:

0 = Bei Ausführung des Parameters *GTU* wird der erste, zweite, ..., letzte Teil des Grundtextes in den Arbeitstext übertragen.

1 = Bei Ausführung des Parameters *GTU* wird der letzte, zweitletzte, ..., erste Teil des Grundtextes in den Arbeitstext übertragen.

## PROGRAMMTEIL 1 (Abfrage + Vergleich):

### Verzweigen in Abhängigkeit von Wahlschaltern

**WS+** Die Sprungbedingung ist erfüllt, falls einer der angegebenen Wahlschalter gesetzt ist (siehe auch Parameter *SPW*). [ I ]

Es können bis zu 16 Zahlenwerte (Nummern der Wahlschalter 1 bis 16) angegeben werden.

**WS-** Die Sprungbedingung ist erfüllt, falls einer der angegebenen Wahlschalter nicht gesetzt ist (siehe auch Parameter *SPW*). [ I ]

Es können bis zu 16 Zahlenwerte (Nummern der Wahlschalter 1 bis 16) angegeben werden.

### Verzweigen in Abhängigkeit von einer Zeichenfolge/Variablen

**PV** Anfangsposition für die Parameter *ZFS* und *KEN* sowie zum Erstellen des Vergleichstextes (d. h. Position, ab der der Vergleichstext aus dem Arbeitstext ausgewählt werden soll), falls diese nicht 1 ist bzw. sich von der unter *POS* angegebenen Anfangsposition unterscheidet. [ I ]  
<1. Zahlenwert des Parameters *POS*>



- ZFS** Zeichenfolgen, die im Arbeitstext gesucht werden sollen. Von der Zeichenfolge, die als erste vorkommt, wird in der Variablen `S8` (vgl. Seite 984) abgespeichert, als wievielte sie zu diesem Parameter angegeben ist. [ IX ]
- VSP** Sprungmarken, die in Abhängigkeit vom Wert der Variablen `S8` (vgl. Seite 984) angesprungen werden sollen. Ist der Wert `n`, wird zur `n`-ten Sprungmarke gesprungen; ist der Wert kleiner 1 oder größer der Anzahl der angegebenen Sprungmarken, so wird kein Sprung ausgeführt. [ I ]

## Verzweigen in Abhängigkeit von einer Kennung

- PV** Anfangsposition für den Vergleichstext.  
(Beschreibung des Parameters `PV` auf Seite 944)
- KEN** Zeichenfolgen, die als (Anfangs)-Kennung von Texteinheiten dienen. Beginnt der Arbeitstext mit einer angegebenen Zeichenfolge, so wird auf die dazugehörige Sprungmarke, die unter `KSP` angegeben ist, gesprungen. Fehlt diese oder beginnt die Texteinheit anders, so wird kein Sprung ausgeführt. [ VIII-a ]
- KSP** Sprungmarken parallel zu den Zeichenfolgen des Parameters `KEN`. [ I ]
- Falls mit dem Parameter `KEN` nicht nur Suchzeichenfolgen, sondern auch Ausnahmezeichenfolgen angegeben sind, muss für jede Ausnahmezeichenfolge die Sprungmarke `8` angegeben werden; folgen nach Ausnahmezeichenfolgen keine Suchzeichenfolgen mehr, kann für dies Ausnahmezeichenfolgen die Angabe einer Sprungmarke entfallen.

## Erstellen des Vergleichstextes

Die folgenden Auswahlparameter (`PV` bis `VI`) sind nur notwendig, falls nicht der ganze Arbeitstext als Vergleichstext gelten soll.

- PV** Anfangsposition für den Vergleichstext.  
(Beschreibung des Parameters `PV` auf Seite 944)
- AV** Zeichenfolgen, die den Anfang des Textteils im Arbeitstext kennzeichnen, der als Vergleichstext herangezogen werden soll. [ IX ]
- EV** Zeichenfolgen, die das Ende des Textteils im Arbeitstext kennzeichnen, der als Vergleichstext herangezogen werden soll. [ IX ]
- (V** Zeichenfolgen, die bei der Auswahl des Vergleichstextes (falls `AV` und/oder `EV` nicht angegeben) bzw. die bei der Eliminierung von Textteilen aus dem bereits mit `AV` und/oder `EV` ausgewählten Teil als öffnende Klammer dienen sollen. [ IX ]

Die Wirkung der öffnenden Klammer wird mit dem zweiten Zahlenwert des Parameters  $\vee\text{I}$  bestimmt.

**)V** Zeichenfolgen, die bei der Auswahl des Vergleichstextes (falls  $\text{AV}$  und/oder  $\text{EV}$  nicht angegeben) bzw. die bei der Eliminierung von Textteilen aus dem bereits mit  $\text{AV}$  und/oder  $\text{EV}$  ausgewählten Teil als schließende Klammer dienen sollen. [ IX ]

Die Wirkung der schließenden Klammer wird mit dem zweiten Zahlenwert des Parameters  $\vee\text{I}$  bestimmt.

**VI** Index für die Parameter  $\text{AV}$ ,  $\text{EV}$ , ( $\vee$  und  $)\vee$ . [ I ]

Es können zwei Zahlenwerte angegeben werden:

1. Zahl: Angabe analog zum Parameter  $\text{AEI}$  (siehe Seite 954)  $\langle 1 \rangle$

2. Zahl: Angabe analog zum Parameter  $\text{KLI}$  (siehe Seite 955)  $\langle 0 \rangle$

**xv** Zeichenfolgenpaare (und Ausnahmezeichenfolgen), deren jeweils erste Zeichenfolge durch die jeweils zweite Zeichenfolge im Vergleichstext ersetzt werden soll. [ X ]

## Verzweigen in Abhängigkeit von einer Zeichenfrequenzhäufigkeit

**ZFZ** Zeichenfolgen, die im Vergleichstext gezählt werden sollen. Die Anzahl wird auf die Variable  $\text{s5}$  (vgl. Seite 984) abgespeichert. Von der Zeichenfolge, die als erste im Vergleichstext vorkommt, wird in der Variablen  $\text{s6}$  (vgl. Seite 984) abgespeichert, als wievielte sie zu diesem Parameter angegeben ist. [ IX ]

**ZSP** Sprungmarken, die in Abhängigkeit von der Anzahl der unter  $\text{ZFZ}$  gezählten Zeichenfolgen angesprungen werden sollen. Ist die Anzahl  $n$ , wird zur  $n$ -ten Sprungmarke gesprungen; ist die Anzahl 0 oder größer als die Anzahl der angegebenen Sprungmarken, so wird kein Sprung ausgeführt. [ I ]

## Auswählen der Texteinheiten

Falls unter den Texteinheiten für die weitere Verarbeitung eine Auswahl getroffen werden soll, so können mit den folgenden Parametern die Bedingungen für die Auswahl angegeben werden.

Die Auswahl erfolgt in fünf Schritten. Damit eine Texteinheit am Ende als ausgewählt gilt, muss sie in jedem Schritt, für den Parameter angegeben sind, ausgewählt werden.

### Schritt 1

In diesem Schritt kann überprüft werden, ob der Vergleichstext nur aus bestimmten Zeichen oder Zeichenfolgen besteht.

**ZFP** Zeichenfolgen, aus denen der Vergleichstext bestehen darf. [ IX ]  
 Enthält der Vergleichstext Zeichen oder Zeichenfolgen, die sich nicht aus den mit dem Parameter angegebenen zusammensetzen lassen, so wird die Texteinheit nicht ausgewählt.

## Schritt 2

In diesem Schritt kann überprüft werden, ob der Vergleichstext

- aus einer bestimmten Zeichenfolge besteht / nicht besteht
- eine bestimmte Zeichenfolge enthält / nicht enthält
- mit einer bestimmten Zeichenfolge beginnt / nicht beginnt
- mit einer bestimmten Zeichenfolge endet / nicht endet

Stimmt der Vergleichstext mit einem zum Parameter T+N, T+U oder T+ angegebenen Textteil überein, wird die Texteinheit ausgewählt. In diesem Fall unterbleiben weitere Überprüfungen auf Grund der übrigen Parameter dieses Schrittes. Sind keine anderen Parameter dieses Schrittes angegeben, so werden nur diese Texteinheiten ausgewählt.

**T+N** Textteile, von denen einer mit dem Vergleichstext übereinstimmen muss, damit die Texteinheit ausgewählt wird. [ III ]

Groß- und Kleinbuchstaben werden nicht unterschieden.

**T+U** Textteile, von denen einer mit dem Vergleichstext übereinstimmen muss, damit die Texteinheit ausgewählt wird. [ III ]

Groß- und Kleinbuchstaben werden unterschieden.

**T+** Textteile, von denen einer mit dem Vergleichstext übereinstimmen muss, damit die Texteinheit ausgewählt wird. [ III ]

Beim Vergleich wird die Sonder-Sortierfolge zugrunde gelegt; d. h. die Sonderzeichen, die mit x bzw. ^x codiert sind (z. B. ! und ^!, & und ^&) sowie die Groß- und Kleinbuchstaben werden nicht unterschieden.

Stimmt der Vergleichstext mit einem zum Parameter T-N, T-U oder T- angegebenen Textteil überein, wird die Texteinheit nicht ausgewählt. In diesem Fall unterbleiben weitere Überprüfungen auf Grund der übrigen Parameter dieses Schrittes.

**T-N** Textteile, von denen keiner mit dem Vergleichstext übereinstimmen darf, damit die Texteinheit ausgewählt wird. [ III ]

Groß- und Kleinbuchstaben werden nicht unterschieden.

**T-U** Textteile, von denen keiner mit dem Vergleichstext übereinstimmen darf, damit die Texteinheit ausgewählt wird. [ III ]

Groß- und Kleinbuchstaben werden unterschieden.

**T-** Textteile, von denen keiner mit dem Vergleichstext übereinstimmen darf, damit die Texteinheit ausgewählt wird. [ III ]

Beim Vergleich wird die Sonder-Sortierfolge zugrunde gelegt; d. h. die Sonderzeichen, die mit x bzw. ^x codiert sind (z. B. ! und ^!, & und ^&) sowie die Groß- und Kleinbuchstaben werden nicht unterschieden.

Mit den Parametern EZ+, AZ+, ZF+, ZB+, TA+ und TE+ können Bedingungen angegeben werden, unter denen eine Texteinheit ausgewählt werden soll. Falls von diesen Parametern einer oder mehrere angegeben sind, muss der Vergleichstext mindestens eine dieser Bedingungen erfüllen, damit die Texteinheit ausgewählt wird. Falls jedoch der Parameter EZ+ zusammen mit dem Parameter ZF+ oder der Parameter AZ+ zusammen mit dem Parameter ZF+ angegeben wird, müssen jeweils beide Bedingungen erfüllt sein, damit die Texteinheit ausgewählt wird.

**EZ+** Zeichenfolgen, von denen mindestens eine im Vergleichstext vorkommen muss, damit die Texteinheit ausgewählt wird. [ IX ]

Falls auch der Parameter ZF+ angegeben ist, muss auch eine der dort angegebenen Zeichenfolgen im Vergleichstext vorkommen, damit die Texteinheit ausgewählt wird.

**AZ+** Zeichenfolgen, die alle (in beliebiger Reihenfolge) im Vergleichstext vorkommen müssen, damit die Texteinheit ausgewählt wird. [ IX ]

Falls auch der Parameter ZF+ angegeben ist, muss auch eine der dort angegebenen Zeichenfolgen im Vergleichstext vorkommen, damit die Texteinheit ausgewählt wird.

**ZF+** Zeichenfolgen, von denen mindestens eine im Vergleichstext vorkommen muss, damit die Texteinheit ausgewählt wird. [ IX ]

Falls auch die Parameter EZ+ und/oder AZ+ angegeben sind, muss auch eine Bedingung dieser Parameter erfüllt sein, damit die Texteinheit ausgewählt wird.

**ZB+** Zeichenfolgen, die im Vergleichstext vorkommen müssen, damit die Texteinheit ausgewählt wird. Ob eine oder alle angegebenen Zeichenfolgen vorkommen müssen und welche weiteren Bedingungen gelten sollen, kann mit Optionen festgelegt werden. [ XII ]

Die Optionen werden in der ersten Zeile des Parameters angegeben, die Zeichenfolgen in den darauf folgenden Fortsetzungszeilen.

Die einzelnen Optionen müssen mit einem Schrägstrich getrennt werden; sie sind aus Gründen der Übersichtlichkeit weiter unten beschrieben.

**TA+** Zeichenfolgen, von denen mindestens eine mit dem Anfang des Vergleichstextes übereinstimmen muss, damit die Texteinheit ausgewählt wird. [ VIII-a ]

**TE+** Zeichenfolgen, von denen mindestens eine mit dem Ende des Vergleichstextes übereinstimmen muss, damit die Texteinheit ausgewählt wird. [ VIII-b ]

Mit den Parametern EZ-, AZ-, ZF-, ZB-, TA- und TE- können Bedingungen angegeben werden, unter denen eine Texteinheit nicht ausgewählt werden soll. Falls von diesen Parametern einer oder mehrere angegeben sind, genügt es, wenn der Vergleichstext eine dieser Bedingungen erfüllt, damit die Texteinheit nicht ausgewählt wird. Falls jedoch der Parameter EZ- zusammen mit dem Parameter ZF- oder der

Parameter **AZ-** zusammen mit dem Parameter **ZF-** angegeben wird, müssen jeweils beide Bedingungen erfüllt sein, damit die Texteinheit nicht ausgewählt wird.

**EZ-** Zeichenfolgen, von denen mindestens eine im Vergleichstext vorkommen muss, damit die Texteinheit nicht ausgewählt wird. [ IX ]

Falls auch der Parameter **ZF-** angegeben ist, muss auch eine der dort angegebenen Zeichenfolgen im Vergleichstext vorkommen, damit die Texteinheit nicht ausgewählt wird.

**AZ-** Zeichenfolgen, die alle (in beliebiger Reihenfolge) im Vergleichstext vorkommen müssen, damit die Texteinheit nicht ausgewählt wird. [ IX ]

Falls auch der Parameter **ZF-** angegeben ist, muss auch eine der dort angegebenen Zeichenfolgen im Vergleichstext vorkommen, damit die Texteinheit nicht ausgewählt wird.

**ZF-** Zeichenfolgen, von denen mindestens eine im Vergleichstext vorkommen muss, damit die Texteinheit nicht ausgewählt wird. [ IX ]

Falls auch die Parameter **EZ-** und/oder **AZ-** angegeben sind, muss auch eine Bedingung dieser Parameter erfüllt sein, damit die Texteinheit nicht ausgewählt wird.

**ZB-** Zeichenfolgen, die im Vergleichstext vorkommen müssen, damit die Texteinheit nicht ausgewählt wird. Ob eine oder alle angegebenen Zeichenfolgen vorkommen müssen und welche weiteren Bedingungen gelten sollen, kann mit Optionen festgelegt werden. [ XII ]

Die Optionen werden in der ersten Zeile des Parameters angegeben, die Zeichenfolgen in den darauf folgenden Fortsetzungszeilen.

Die einzelnen Optionen müssen mit einem Schrägstrich getrennt werden; sie sind aus Gründen der Übersichtlichkeit weiter unten beschrieben.

**TA-** Zeichenfolgen, von denen mindestens eine mit dem Anfang des Vergleichstextes übereinstimmen muss, damit die Texteinheit nicht ausgewählt wird. [ VIII-a ]

**TE-** Zeichenfolgen, von denen mindestens eine mit dem Ende des Vergleichstextes übereinstimmen muss, damit die Texteinheit nicht ausgewählt wird. [ VIII-b ]

Sind sowohl einer oder mehrere der Parameter **EZ+**, **AZ+**, **ZF+**, **ZB+**, **TA+** und **TE+** als auch einer oder mehrere der Parameter **EZ-**, **AZ-**, **ZF-**, **ZB-**, **TA-** und **TE-** angegeben, so muss der Vergleichstext mindestens eine der Bedingungen der Parameter **EZ+**, **AZ+**, **ZF+**, **ZB+**, **TA+** und **TE+** erfüllen und darf keine der Bedingungen der Parameter **EZ-**, **AZ-**, **ZF-**, **ZB-**, **TA-** und **TE-** erfüllen, damit die Texteinheit ausgewählt wird.

Optionen für die Parameter **ZB+** und **ZB-**:

Falls mehr als eine Zeichenfolge angegeben ist, so muss mindestens eine der drei Optionen **OR**, **AND** oder **USER** angegeben werden; die Optionen **OR** und **AND** dürfen

jedoch nicht zusammen angegeben werden. Ist die Option USER angegeben, sind die Besonderheiten der Parameterart [ XII-b ] zu beachten, andernfalls die der Parameterart [ XII-a ].

– EXACT

Groß- und Kleinbuchstaben werden als solche interpretiert, es sei denn, dass durch ein vor dem Buchstaben stehendes Zeichen »>« bzw. »<< ausdrücklich der entsprechende Klein- bzw. Großbuchstabe verlangt wird.

Ist diese Option nicht angegeben, so ist bei Buchstaben, die nicht mit »>« oder »<< gekennzeichnet sind, jeweils der entsprechende Groß- und Kleinbuchstabe gemeint (gleichgültig, ob der Buchstabe groß oder klein geschrieben ist).

– n

Um eine zu suchende Zeichenfolge als übereinstimmend mit einer gefundenen gelten zu lassen, darf sie bis zu n ( $n = 0$  bis 9) Unterschiede (Ersetzungen, Auslassungen oder Einfügungen von je 1 Zeichen) aufweisen.

– n:m

Wie Option n, jedoch ist die Anzahl der erlaubten Unterschiede von der Länge der zu suchenden Zeichenfolge abhängig. Sie beträgt bei bis zu n Zeichen 0 Unterschiede, bis zu  $n+m$  Zeichen 1 Unterschied, bis zu  $n+m+m$  Zeichen 2 Unterschiede usw.

– WORD

Die zu suchenden Zeichenfolgen müssen im Vergleichstext an einer Wortgrenze beginnen und enden. Als Wortgrenze gilt jedes Sonderzeichen (einschließlich Leerzeichen) sowie der Anfang und das Ende des Vergleichstextes.

Hinweis: Da Akzente mit Sonderzeichen codiert werden (z. B. %/e für ein e mit Akut), müssen Akzent-Codierungen zuvor (z. B. mit dem Parameter xv) eliminiert werden. Entsprechend müssen Sonderbuchstaben, die mit »#.« codiert sind, ersetzt werden (z. B. #. i durch i).

– TEXT

Die zu suchenden Zeichenfolgen müssen im Vergleichstext an der Textgrenze (= Anfang und Ende des Vergleichstextes) beginnen und enden.

– OR

Falls mehr als eine Zeichenfolge angegeben ist, genügt es, wenn eine davon im Vergleichstext vorkommt.

– AND

Falls mehr als eine Zeichenfolge angegeben ist, müssen sie alle im Vergleichstext vorkommen.

## – USER

Falls mehr als eine Zeichenfolge angegeben ist, können dazwischen logische Operatoren angegeben werden. Wird zwischen zwei Zeichenfolgen kein logischer Operator angegeben, so wird ein logisches ODER angenommen; dieses logische ODER hat Vorrang vor allen angegebenen logischen Operatoren.

Wenn die Option OR zusätzlich angegeben ist, so wird an Stellen, an denen zwischen zwei Zeichenfolgen kein logischer Operator angegeben ist, ein logisches ODER ohne Vorrang angenommen.

Wenn die Option AND zusätzlich angegeben ist, so wird an Stellen, an denen zwischen zwei Zeichenfolgen kein logischer Operator angegeben ist, ein logisches UND angenommen.

## Schritt 3

In diesem Schritt kann die Reihenfolge und die Häufigkeit von vorgegebenen Zeichenfolgen im Vergleichstext überprüft werden.

**ZF** Zeichenfolgen, deren Vorkommen im Vergleichstext auf Reihenfolge und Häufigkeit überprüft werden soll. [ IX ]

Die hier angegebenen Zeichenfolgen dürfen im Vergleichstext nur in der gleichen Reihenfolge vorkommen, in der sie mit diesem Parameter angegeben sind. Andernfalls wird die Texteinheit nicht ausgewählt.

Mit den Parametern ZFM und/oder ZFH kann angegeben werden, wie oft Zeichenfolgen, die mit dem Parameter ZF angegeben sind, im Vergleichstext vorkommen müssen bzw. dürfen. Zwischen solchen Zeichenfolgen, die im Vergleichstext mehrmals vorkommen, darf im Vergleichstext beliebiger Text, aber keine andere mit dem Parameter ZF angegebene Zeichenfolge stehen.

Von der Zeichenfolge, die als erste im Vergleichstext vorkommt, wird in der Variablen S4 (vgl. Seite 960) abgespeichert, als wievielte sie zum Parameter ZF angegeben ist. Ist jedoch eine der Bedingungen der Parameter ZF, ZFM und ZFH nicht erfüllt, so wird in der Variablen S4 abgespeichert, als wievielte die Zeichenfolge angegeben ist, die als erste eine dieser Bedingungen nicht erfüllt.

**ZFM** Angabe (parallel zu den Zeichenfolgen des Parameters ZF), wie oft die jeweilige Zeichenfolge im Vergleichstext mindestens vorkommen muss. [ I ] <0, 0, 0, . . . >

Erfüllt eine der Zeichenfolgen diese Bedingung nicht, so wird die Texteinheit nicht ausgewählt.

Falls mit dem Parameter ZF nicht nur Suchzeichenfolgen, sondern auch Ausnahmezeichenfolgen angegeben sind, muss für jede Ausnahmezeichenfolge eine Null angegeben werden; folgen nach Ausnahmezeichenfolgen keine Suchzeichenfolgen mehr, kann für diese Ausnahmezeichenfolgen die Angabe einer Null entfallen.

**ZFH** Angabe (parallel zu den Zeichenfolgen des Parameters ZF), wie oft die jeweilige Zeichenfolge im Vergleichstext höchstens vorkommen darf. [ 1 ] <1, 1, 1, . . . >

Erfüllt eine der Zeichenfolgen diese Bedingung nicht, so wird die Texteinheit nicht ausgewählt.

Falls mit dem Parameter ZF nicht nur Suchzeichenfolgen, sondern auch Ausnahmezeichenfolgen angegeben sind, muss für jede Ausnahmezeichenfolge eine Null angegeben werden; folgen nach Ausnahmezeichenfolgen keine Suchzeichenfolgen mehr, kann für diese Ausnahmezeichenfolgen die Angabe einer Null entfallen.

#### Schritt 4

In diesem Schritt kann der Vergleichstext mit einem zuvor gemerkten Vergleichstext verglichen werden. Dabei kann geprüft werden, welcher der beiden Texte in der alphabetischen Ordnung kleiner bzw. größer ist oder ob sie übereinstimmen. Der Merk-Vergleichstext kann beim Programmstart mit dem Parameter VTV vorbelegt werden und während des Programmlaufs im Programmteil 5 jeweils neu definiert (gemerkt) werden.

**VGL** Angabe, in welcher Weise Vergleichstext und Merk-Vergleichstext miteinander verglichen werden sollen. [ 1 ]

Es können zwei Zahlenwerte angegeben werden:

1. Zahl: Falls 2. Zahl 0 (Null) ist, Art des Vergleichs.

- 0 = Beide Texte müssen gleich sein.
- 1 = Beide Texte müssen gleich sein oder der Vergleichstext muss bei alphabetischer Ordnung vor dem Merk-Vergleichstext stehen.
- 2 = Der Vergleichstext muss bei alphabetischer Ordnung vor dem Merk-Vergleichstext stehen.
- 3 = Beide Texte müssen gleich sein oder der Vergleichstext muss bei alphabetischer Ordnung nach dem Merk-Vergleichstext stehen.
- 4 = Der Vergleichstext muss bei alphabetischer Ordnung nach dem Merk-Vergleichstext stehen.
- 5 = Beide Texte müssen bis zur Länge des kürzeren Textes gleich sein.
- n = Beide Texte dürfen sich in n Prozent der Zeichen der kürzeren Zeichenfolge unterscheiden. Der Vergleichsalgorithmus hierfür ist auf Seite 982 beschrieben.

2. Zahl: Falls nicht 0 (Null), erlaubte Unterschiede.

- m = Beide Texte dürfen sich unterscheiden. Die Anzahl der erlaubten Unterschiede ist von der Länge der kürzeren Zeichenfolge abhängig. Sie beträgt bei bis zu n (= 1. Zahl dieses Parameters) Zeichen 0 Unterschiede, bis zu n+m Zeichen 1



Unterschied, bis zu  $n+m+m$  Zeichen 2 Unterschiede usw. Der Vergleichsalgorithmus hierfür ist auf Seite 982 beschrieben.

**VTB** Vergleichs-Tabelle (Sortieralphabet) zum Vergleich des Vergleichstextes mit dem Merk-Vergleichstext. [ VII ]

#### Schritt 5

In diesem Schritt kann überprüft werden, ob die erste im Vergleichstext vorkommende Zahl mit der dazugehörenden Prüfziffer versehen ist.

**PZP** Angabe der Art und der Form der Prüfziffer. [ I ]

Es können zwei Zahlenwerte angegeben werden:

1. Zahl: Art der Prüfziffer

0 = keine Prüfziffer im Vergleichstext überprüfen

1 = Prüfziffer = Differenz von der Summe der gewichteten Ziffern und der nachfolgenden durch 11 teilbaren Zahl

2 = Prüfziffer = Differenz von der Summe der gewichteten Ziffern und der vorangehenden durch 11 teilbaren Zahl

2. Zahl: Form der Prüfziffer

0 = Prüfziffer steht unmittelbar hinter der Zahl

1 = Prüfziffer ist mit einem »-« von der Zahl abgetrennt

Wie die Prüfziffer berechnet wird, ist unter »Prüfziffern« (Seite 981) beschrieben.

## PROGRAMMTEIL 2 (Verarbeitung):

### Setzen und Löschen von Wahlschaltern

**WSS** Wahlschalter, die gesetzt werden sollen. [ I ]

Es können bis zu 16 Zahlenwerte (Nummern der Wahlschalter 1 bis 16) angegeben werden.

**WSL** Wahlschalter, die gelöscht werden sollen. [ I ]

Es können bis zu 16 Zahlenwerte (Nummern der Wahlschalter 1 bis 16) angegeben werden.

**WSU** Wahlschalter, die umgesetzt werden sollen. [ I ]

Es können bis zu 16 Zahlenwerte (Nummern der Wahlschalter 1 bis 16) angegeben werden.

## Bestimmen des Arbeitstextes

- T** Text, der als Arbeitstext übernommen werden soll. [ II ]  
 Falls dieser Parameter angegeben ist, wird der angegebene Text als Arbeitstext übernommen. Dabei bleiben eventuell mit dem Parameter POS angegebene Positionsangaben unberücksichtigt; d. h. der bisherige Arbeitstext wird in jedem Fall ab Position 1 vollständig ersetzt.
- PK** Anfangsposition, ab der die einzelnen Textteile ausgewählt werden sollen, falls diese nicht 1 ist bzw. sich von der unter POS angegebenen Anfangsposition unterscheidet. Die Texteinheit wird bis zu dieser Position (ausschließlich) unverändert übernommen. Die erste unter ERG angegebene Zeichenfolge wird ggf. an dieser Stelle eingefügt. [ I ] <1. Zahlenwert des Parameters POS>
- AKn** Zeichenfolgen, die den Anfang des n-ten Teils ( $n=1, 2, \dots, 9$ ) für den neuen Arbeitstext kennzeichnen. [ IX ]
- EKn** Zeichenfolgen, die das Ende des n-ten Teils ( $n=1, 2, \dots, 9$ ) für den neuen Arbeitstext kennzeichnen. [ IX ]
- AEI** Index für die Parameter AKn und EKn. [ I ] <1, 1, 1, 1, 1, 1, 1, 1, 1>

Es können ein bis neun Zahlenwerte angegeben werden:

- 1 = Es wird der erste mit A/E gekennzeichnete Textteil (er beginnt mit einer Anfangskennung und endet vor der nachfolgenden Endekennung bzw. am Ende der Texteinheit) ausgewählt.  
 Ist nur A angegeben, so endet der ausgewählte Textteil am Ende der Texteinheit; ist nur E angegeben, so beginnt der ausgewählte Textteil am Anfang der Texteinheit.
- 0 = Es wird der Teil der Texteinheit ausgewählt, der bei 1 wegfallen würde.
- 3 = Wie 1, jedoch wird nicht nur der erste, sondern es werden alle mit A/E gekennzeichneten Textteile (der zweite beginnt mit der Anfangskennung, die dem Ende des ersten mit A/E gekennzeichneten Textteils folgt) ausgewählt.  
 Ist nur A angegeben, so beginnt der ausgewählte Textteil mit der letzten in der Texteinheit vorkommenden Anfangskennung und endet am Ende der Texteinheit; ist nur E angegeben, so beginnt der ausgewählte Textteil am Anfang der Texteinheit und endet vor der letzten in der Texteinheit vorkommenden Endekennung.
- 2 = Es wird der Teil der Texteinheit ausgewählt, der bei 3 wegfallen würde.

Bei den Werten 0 bis 3 wird die Anfangskennung jeweils zum nachfolgenden Text gerechnet, während die Endekennung nicht mehr zum davor stehenden Text gerechnet wird. Durch Aufaddieren von 10 und/oder 20 kann diese Regelung für die Anfangs- und/oder Endekennung umgekehrt werden. Wird zu den Werten 10 aufaddiert (also 10 bis 13 angegeben), so wird die Anfangskennung jeweils nicht zum

nachfolgenden Text gerechnet; wird 20 addiert, so wird die Endekennung jeweils noch zum davor stehenden Text gerechnet; wird 30 addiert, so wird die Anfangskennung nicht zum nachfolgenden Text und die Endekennung zum davor stehenden Text gerechnet.

Bei den Werten 2 und 3 wird die nächste Anfangskennung ab der ersten Position der zuletzt gefundenen Endekennung gesucht, da sie nicht mehr zum davor stehenden Textteil gehört. Anfangs- und Endekennung können sich also im Text überschneiden. Wurde auf diese Werte 20 oder 30 aufaddiert, so wird die nächste Anfangskennung erst ab der Position gesucht, die auf die letzte Position der zuletzt gefundenen Endekennung folgt, da diese noch zum davor stehenden Textteil gehört.

**(Kn** Zeichenfolgen, die bei der Auswahl des  $n$ -ten Teils ( $n=1, 2, \dots, 9$ ) für den neuen Arbeitstext (falls  $AK_n$  und/oder  $EK_n$  nicht angegeben) bzw. die bei der Eliminierung von Textteilen aus dem bereits mit  $AK_n$  und/oder  $EK_n$  ausgewählten Teil als öffnende Klammer dienen sollen. [ IX ]

Die Wirkung der öffnenden Klammer wird mit dem  $n$ -ten Zahlenwert des Parameters  $KL_I$  bestimmt.

**)Kn** Zeichenfolgen, die bei der Auswahl des  $n$ -ten Teils ( $n=1, 2, \dots, 9$ ) für den neuen Arbeitstext (falls  $AK_n$  und/oder  $EK_n$  nicht angegeben) bzw. die bei der Eliminierung von Textteilen aus dem bereits mit  $AK_n$  und/oder  $EK_n$  ausgewählten Teil als schließende Klammer dienen sollen. [ IX ]

Die Wirkung der schließenden Klammer wird mit dem  $n$ -ten Zahlenwert des Parameters  $KL_I$  bestimmt.

**KL<sub>I</sub>** Index für die Parameter  $(Kn$  und  $)Kn$ . [ I ]  $\langle 0, 0, 0, 0, 0, 0, 0, 0, 0 \rangle$

Es können ein bis neun Zahlenwerte angegeben werden:

- 0 = Es werden die Teile (einschließlich der Klammern) eliminiert, die eingeklammert sind. Fehlende Klammern werden am Anfang bzw. am Ende der Texteinheit bzw. des bereits ausgewählten Textteils logisch ergänzt.
- 1 = Es werden die Teile ausgewählt, die bei 0 wegfallen würden.
- 2 = Wie 0, jedoch werden fehlende Klammern nicht ergänzt, sondern die unpaarigen Klammern werden ignoriert.
- 3 = Es werden die Teile ausgewählt, die bei 2 wegfallen würden.

Bei den Werten 0 bis 3 werden die Klammern jeweils zum eingeklammerten Text gerechnet und werden mit ihm eliminiert bzw. bleiben mit ihm erhalten. Durch Aufaddieren von 10 und/oder 20 kann diese Regelung für die öffnenden und/oder schließenden Klammern umgekehrt werden. Wird auf diese Werte 10 aufaddiert (also 10 bis 13 angegeben), so werden die öffnenden Klammern nicht zum eingeklammerten Text gerechnet; wird 20 addiert, so werden die schließenden Klammern nicht zum eingeklammerten Text gerechnet; wird 30 ad-

diert, so werden beide Klammern nicht zum eingeklammerten Text gerechnet.

**ERG** Textteile, die am Anfang, zwischen den einzelnen (mit  $AK_n, EK_n$  und/oder  $(K_n, )K_n$  ausgewählten) Textteilen und am Ende der Texteinheit ergänzt werden sollen. [ II ]

Ist für einen Textteil kein Auswahlparameter angegeben, so wird die ganze Texteinheit (ggf. ab der mit dem Parameter  $POS$  bzw.  $PK$  angegebenen Anfangsposition) kopiert.

## Austauschen von Textteilen

**ATT** Zeichenfolgen, die den Anfang der Textteile kennzeichnen, die ausgetauscht werden sollen. [ IX ]

**ETT** Zeichenfolgen, die das Ende der Textteile kennzeichnen, die ausgetauscht werden sollen. [ IX ]

**(TT** Zeichenfolgen, die bei der Abgrenzung der auszutauschenden Textteile (ggf. nach Abgrenzung durch  $ATT$  und/oder  $ETT$ ) als öffnende Klammer dienen sollen. [ IX ]

Die Wirkung der öffnenden Klammer wird mit dem zweiten Zahlenwert des Parameters  $TTI$  bestimmt.

**)TT** Zeichenfolgen, die bei der Abgrenzung der auszutauschenden Textteile, (ggf. nach Abgrenzung durch  $ATT$  und/oder  $ETT$ ) als schließende Klammer dienen sollen. [ IX ]

Die Wirkung der schließenden Klammer wird mit dem zweiten Zahlenwert des Parameters  $TTI$  bestimmt.

**TTI** Index für die Parameter  $ATT, ETT, (TT$  und  $)TT$ . [ I ]

Es können zwei Zahlenwerte angegeben werden:

1. Zahl: Angabe analog zum Parameter  $\Delta EI$  (siehe Seite 954)  $\langle 1 \rangle$
2. Zahl: Angabe analog zum Parameter  $KLI$  (siehe Seite 955)  $\langle 0 \rangle$

**TTR** Zeichenfolgen, die als Trennzeichen zwischen den einzelnen auszutauschenden Textteilen stehen. [ IX ]

**TTT** Paare von Textteilen. [ IV ]

Der jeweils erste Textteil wird auf Übereinstimmung mit den durch die obigen Parameter ausgewählten einzelnen Textteilen im Arbeitstext geprüft. Dabei werden Groß- und Kleinbuchstaben nicht unterschieden. Bei Übereinstimmung wird der Textteil im Arbeitstext durch den jeweils zweiten Textteil ersetzt.

Die Parameter  $TTT, TUT$  und  $TTM$  schließen sich im gleichen Durchgang gegenseitig aus.

- TUT** Paare von Textteilen. [ IV ]  
 Wie Parameter TTT, jedoch werden bei der Prüfung auf Übereinstimmung Groß- und Kleinbuchstaben unterschieden.  
 Die Parameter TTT, TUT und TTM schließen sich im gleichen Durchgang gegenseitig aus.
- TTM** Paare von Textteilen. [ IV ]  
 Wie Parameter TTT, jedoch muss der jeweils zweite Textteil sämtliche Zeichen des jeweils ersten Textteils in der gleichen Reihenfolge enthalten.  
 Es ist gleichgültig, ob im jeweils ersten Textteil die Buchstaben groß oder klein geschrieben werden. Jeder Klein- bzw. Großbuchstabe muss aber im jeweils zweiten Textteil in gleicher Weise als Klein- bzw. Großbuchstabe angegeben werden.  
 Im zweiten Textteil dürfen an jeder Stelle zusätzliche Zeichen eingefügt werden, jedoch muss erkenntlich bleiben, welche Zeichen aus dem ersten Textteil übernommen wurden und welche Zeichen eingefügt wurden.  
 Beim Ersetzen eines Textteils werden die Buchstaben, die vom ersten in den zweiten Textteil übernommen wurden, in der Form (d. h. als Groß- bzw. Kleinbuchstabe) eingesetzt, in der sie im auszutauschenden Text (also nicht wie im Parameter angegeben) stehen.  
 Die Parameter TTT, TUT und TTM schließen sich im gleichen Durchgang gegenseitig aus.

## Umdrehen von Textteilen

- AUM** Zeichenfolgen, die den Anfang der Textteile kennzeichnen, die umgedreht werden sollen. [ IX ]  
 Der umzudrehende Textteil beginnt jeweils nach der Anfangskennung.
- EUM** Zeichenfolgen, die das Ende der Textteile kennzeichnen, die umgedreht werden sollen. [ IX ]  
 Der umzudrehende Textteil endet jeweils vor der Endekennung bzw. am Ende des Arbeitstextes.
- NUM** Zeichenfolgen, die nicht umgedreht werden sollen, falls sie in den umzudrehenden Textteilen vorkommen. [ IX ]
- XUM** Zeichenfolgenpaare (und Ausnahmezeichenfolgen), deren jeweils erste Zeichenfolge durch die jeweils zweite Zeichenfolge innerhalb der nicht umzudrehenden (mit dem Parameter NUM angegebenen) Zeichenfolgen ersetzt werden soll. [ X ]

## Berechnen der Druckpositionen

**DPB** Druckertyp, für den die Länge des Arbeitstextes in Anzahl Druckpositionen berechnet werden soll. [ XI ]

Hinweis: Mit dem Kommando #LISTE, DRUCKERTYPEN werden die definierten Druckertypen aufgelistet.

Die Anzahl der errechneten Druckpositionen wird in der Variablen S1 (vgl. Seite 984) abgespeichert.

## Lesen von Zahlenwerten aus dem Arbeitstext

**PL** Anfangsposition, ab der der Textteil beginnt bzw. ausgewählt werden soll, aus dem Zahlen gelesen werden sollen, falls diese Position nicht 1 ist bzw. sich von der unter POS angegebenen Anfangsposition unterscheidet. [ I ] <1. Zahlenwert des Parameters POS>

**AL** Zeichenfolgen, die den Anfang des Textteils kennzeichnen, aus dem Zahlen gelesen werden sollen. [ IX ]

**EL** Zeichenfolgen, die das Ende des Textteils kennzeichnen, aus dem Zahlen gelesen werden sollen. [ IX ]

**(L** Zeichenfolgen, die bei der Auswahl des Textteils, aus dem Zahlen gelesen werden sollen (falls AL und/oder EL nicht angegeben), bzw. die bei der Eliminierung von Textteilen aus dem bereits mit AL und/oder EL ausgewählten Teil als öffnende Klammer dienen sollen. [ IX ]

Die Wirkung der öffnenden Klammer wird mit dem zweiten Zahlenwert des Parameters LI bestimmt.

**)L** Zeichenfolgen, die bei der Auswahl des Textteils, aus dem Zahlen gelesen werden sollen (falls AL und/oder EL nicht angegeben), bzw. die bei der Eliminierung von Textteilen aus dem bereits mit AL und/oder EL ausgewählten Teil als schließende Klammer dienen sollen. [ IX ]

Die Wirkung der schließenden Klammer wird mit dem zweiten Zahlenwert des Parameters LI bestimmt.

**LI** Index für die Parameter AL, EL, (L und )L. [ I ]

Es können zwei Zahlenwerte angegeben werden:

1. Zahl: Angabe analog zum Parameter AET (siehe Seite 954) <1>
2. Zahl: Angabe analog zum Parameter KLI (siehe Seite 955) <0>

**XL** Zeichenfolgenpaare (und Ausnahmezeichenfolgen), deren jeweils erste Zeichenfolge durch die jeweils zweite Zeichenfolge in dem Textteil, aus dem Zahlen gelesen werden sollen, ersetzt werden soll. [ X ]

**LIV** I-Variablen, in die Zahlen eingelesen werden sollen. [ I ]

Ein Minuszeichen unmittelbar vor einer einzulesenden (in arabischen Ziffern geschriebenen) Zahl wird als Vorzeichen gewertet.

Werden weniger Zahlen gefunden als mit diesem Parameter I-Variablen angegeben sind, so werden die restlichen I-Variablen auf den Wert 0 gesetzt.

**LDN** Angaben parallel zu LIV, wieviele Dezimalstellen bei (in arabischen Ziffern geschriebenen) Zahlen nach dem Komma bzw. Dezimalpunkt gelesen werden sollen. Ist z. B. 2 angegeben, so ergeben »123«, »123,0«, »123,00« und »123,005« den Wert 12300. [ I ] <0>

**LIZ** Angaben parallel zu LIV, ob eine in arabischen Ziffern geschriebene, eine römische, eine hebräische oder eine Hexadezimalzahl gelesen werden soll. Bei römischen, hebräischen und Hexadezimalzahlen werden die dazugehörenden Angaben des Parameters LDN ignoriert. [ I ] <0>

Es kann für jede mit dem Parameter LIV angegebene I-Variable ein Zahlenwert angegeben werden:

- 0 = Zahl in arabischen Ziffern
- 1 = römische Zahl (aus I,V,X,L,C,D,M,i,v,x,l,c,d,m)
- 22 = Zahl mit zwei Hexadezimalziffern (= 1 Byte)
- 42 = Zahl mit vier Hexadezimalziffern (= 2 Bytes)
- 3 = hebräische Zahl

## Ausführen von Rechenanweisungen

**RR** Rechen-, Abfrage- und Sprunganweisungen [ XI ]  
Beschreibung siehe »Rechenanweisungen« Seite 983 ff.

## Einsetzen von Zahlenwerten in den Arbeitstext

**PE** Anfangsposition, ab der (die Zahlen, der Ersetzungstext, die laufende Nummer, die Seitennummer) ersetzt/eingesetzt werden soll, falls diese nicht 1 ist bzw. sich von der unter POS angegebenen Anfangsposition unterscheidet. [ I ] <1. Zahlenwert des Parameters POS>

**EIN** Zeichenfolgen, die den Anfang des Textteils kennzeichnen, in dem Zahlen ersetzt werden sollen. [ IX ]

**EIB** Zeichenfolgen, die das Ende des Textteils kennzeichnen, in dem Zahlen ersetzt werden sollen. [ IX ]

**EIV** I-Variablen, deren Werte die alten Zahlen ersetzen sollen. [ I ]

Ein Minus- oder Pluszeichen unmittelbar vor einer arabischen Zahl, die ersetzt wird, wird entfernt. Soll ein Wert (auf Grund entsprechender Angaben mit dem Parameter EIZ) als römische, als hebräische

oder als Hexadezimalzahl eingesetzt werden, so wird die römische, die hebräische bzw. die Hexadezimalzahl ersetzt.

Werden in dem mit EIN und/oder EIB abgegrenzten Textteil bzw. im gesamten Arbeitstext, falls kein Textteil abgegrenzt wurde, weniger Zahlen gefunden, als mit dem Parameter EIV Variablen angegeben sind, so werden die Zahlen am Ende des abgegrenzten Textteils bzw. Arbeitstextes eingefügt, falls mit dem Parameter EIZ nichts anderes bestimmt wird.

**EDV** Angaben parallel zu EIV, auf wieviele Stellen vor dem Komma die einzusetzende Zahl mit Leerzeichen aufgefüllt werden soll. [ I ] <0>

Soll mit führenden Nullen statt mit Leerzeichen aufgefüllt werden, so kann dies mit dem Parameter EIZ angegeben werden.

**EDN** Angaben parallel zu EIV, wieviele Stellen nach dem Komma stehen sollen. Ist z. B. 2 angegeben, so ergibt der Wert 12345 »123,45« [ I ] <0>

Die Werte der I-Variablen können in verschiedener Form in den Arbeitstext eingesetzt werden: als normale Zahlen (in arabischen Ziffern), als römische Zahlen, als hebräische Zahlen, als mit einer Prüfziffer versehene Zahlen oder als Hexadezimalzahlen. Die Form kann mit folgendem Parameter gewählt werden.

**EIZ** Angaben parallel zu EIV, in welcher Form der jeweilige Zahlenwert eingesetzt werden soll. [ I ] <0, 0, 0, . . . >

Es kann für jede mit dem Parameter EIV angegebene I-Variable ein Zahlenwert angegeben werden:

- 0 = normale Zahl (in arabischen Ziffern)
- 1 = römische Zahl in Kleinbuchstaben
- 2 = römische Zahl in Großbuchstaben
- x3 = Zahl mit Prüfziffer, die unmittelbar anschließt
- x4 = Zahl mit Prüfziffer, die mit »-« abgetrennt ist
- 5 = wie 0, jedoch wird bei positiven Zahlen mit führenden Nullen statt mit Leerzeichen aufgefüllt, falls dies mit dem Parameter EDV verlangt wird.
- 26 = Zahl aus zwei Hexadezimalziffern (= 1 Byte)
- 46 = Zahl aus vier Hexadezimalziffern (= 2 Bytes)
- 7 = hebräische Zahl, nur Ziffern bis Taw (= 400)
- 8 = hebräische Zahl, auch Ziffern größer Taw

Sollen nur vorhandene Zahlenwerte ersetzt und die überzähligen I-Variablen ignoriert werden, so muss für jede zu ignorierende I-Variable ein um 100 erhöhter Wert angegeben werden.

Soll eine Zahl mit Prüfziffer eingesetzt werden, so muss zusätzlich noch spezifiziert werden, welcher Typ von Prüfziffer (vgl. »Prüfziffern« Seite 981) verwendet werden soll. Dazu muss auf die Werte 3 bzw. 4 in jedem Fall noch 10 oder 20 aufaddiert werden, und zwar 10, falls Prüfziffer = Differenz von der Summe der gewichteten Ziffern und der nachfolgenden durch 11 teilbaren Zahl, und 20, falls Prüfziffer = Dif-



ferenz von der Summe der gewichteten Ziffern und der vorangehenden durch 11 teilbaren Zahl.

Wie die Prüfziffer berechnet wird, ist unter »Prüfziffern« (Seite 981) beschrieben.

**EIK** Zeichenfolgen, die die Stellen kennzeichnen, nach denen jeweils eine Zahl durch einen neuen Wert ersetzt werden soll. [ IX ]

Ist die im Arbeitstext gefundene Zeichenfolge als n-te Zeichenfolge mit dem Parameter **EIK** angegeben, so wird jeweils die nachfolgende Zahl durch den Wert der I-Variablen ersetzt, die als n-te mit dem Parameter **EIV** angegeben ist. Die einzelnen Zeichenfolgen können im Text auch mehrfach vorkommen. Es wird dann jeweils danach die entsprechende I-Variable eingesetzt.

Wird im Arbeitstext bis zur nächsten Zeichenfolge, nach der wiederum eine Zahl ersetzt werden soll, keine Zahl gefunden, so wird der Wert der n-ten I-Variablen unmittelbar vor dieser Zeichenfolge eingefügt. Entsprechend wird der Wert ans Ende des Arbeitstextes bzw. des mit **EIN** und/oder **EIB** abgegrenzten Textteils eingefügt, wenn keine solche Zeichenfolge mehr folgt.

Wenn der Parameter **EIK** angegeben ist, gilt folgendes: I-Variablen, die mit dem Parameter **EIV** angegeben sind, für die aber mit dem Parameter **EIK** keine entsprechende Zeichenfolge angegeben ist, werden nie eingesetzt. Wird die n-te mit dem Parameter **EIK** angegebene Zeichenfolge im Arbeitstext bzw. in dem mit **EIN** und/oder **EIB** abgegrenzten Textteil nicht gefunden, so wird auch die n-te mit dem Parameter **EIV** angegebene I-Variable nicht eingesetzt. Das gleiche gilt, wenn die n-te mit dem Parameter **EIK** angegebene Zeichenfolge keine Such-, sondern eine Ausnahmezeichenfolge ist.

## Austauschen von Zeichenfolgen

**PX** Anfangsposition, ab der ersetzt werden soll, falls diese nicht 1 ist bzw. sich von der unter **POS** angegebenen Anfangsposition unterscheidet. [ 1 ] <1. Zahlenwert des Parameters **POS**>

**AXX** Zeichenfolgen, die den Anfang des Textteils kennzeichnen, in dem Zeichenfolgen ersetzt werden sollen (nur mit dem Parameter **XX** erlaubt). [ IX ]

**EXX** Zeichenfolgen, die das Ende des Textteils kennzeichnen, in dem Zeichenfolgen ersetzt werden sollen (nur mit dem Parameter **XX** erlaubt). [ IX ]

**(XX** Zeichenfolgen, die bei der Abgrenzung des Textteils, in dem keine Zeichenfolgen ersetzt werden sollen, (ggf. nach Abgrenzung durch **AXX** und/oder **EXX**) als öffnende Klammer dienen sollen (nur mit dem Parameter **XX** erlaubt). [ IX ]

Die Wirkung der öffnenden Klammer wird mit dem zweiten Zahlenwert des Parameters **XXI** bestimmt.

**)XX** Zeichenfolgen, die bei der Abgrenzung des Textteils, in dem keine Zeichenfolgen ersetzt werden sollen, (ggf. nach Abgrenzung durch **AXX** und/oder **EXX**) als schließende Klammer dienen sollen (nur mit dem Parameter **XX** erlaubt). [ IX ]

Die Wirkung der schließenden Klammer wird mit dem zweiten Zahlenwert des Parameters **XXI** bestimmt.

**XXI** Index für die Parameter **AXX**, **EXX**, (**XX** und **)XX**. [ I ]

Es können zwei Zahlenwerte angegeben werden:

1. Zahl: Angabe analog zum Parameter **AEI** (siehe Seite 954) <1>
2. Zahl: Angabe analog zum Parameter **KLI** (siehe Seite 955) <0>

**XX** Zeichenfolgenpaare (und Ausnahmezeichenfolgen), deren jeweils erste Zeichenfolge durch die jeweils zweite Zeichenfolge ersetzt werden soll. [ X ]

## Austauschen von Zeichenfolgen mit Bedingungen

**PX** Anfangsposition zum Ersetzen von Zeichenfolgen  
(Beschreibung des Parameters **PX** auf Seite 961)

**ERN** Zeichenfolgen, nach denen ersetzt werden soll. [ IX ]

**ERB** Zeichenfolgen, bis zu denen (ausschließlich) ersetzt werden soll. [ IX ]

**ERS** Zeichenfolgen, die ersetzt werden sollen (nicht mit dem Parameter **XXB** zusammen erlaubt, Parameter **EZF** erforderlich). [ IX ]

**EZF** Reihe von Textteilen, die für die unter **ERS** angegebenen Zeichenfolgen eingesetzt werden sollen. Für die n-te gefundene Zeichenfolge in der Texteinheit, die unter **ERS** angegeben ist, wird der n-te (falls die Variable **S7** nicht 0 ist, die **S7+n-te**) Textteil von **EZF** bzw. der letzte von **EZF**, falls weniger angegeben sind, eingesetzt. [ II ]

**XXB** Zeichenfolgenpaare (und Ausnahmezeichenfolgen), deren jeweils erste Zeichenfolge durch die jeweils zweite Zeichenfolge ersetzt werden soll. [ X ]

**ERZ** Zusatzangaben zum Ersetzen. [ I ]

Es können fünf Zahlenwerte angegeben werden:

1. Zahl: Anzahl der Zeichenfolgen, die jeweils ersetzt werden sollen.  
<0>

0 = alle Zeichenfolge ersetzen

n = maximal n Zeichenfolgen ersetzen

2. Zahl: Angabe, wie oft eine unter ERN angegebene Zeichenfolge gesucht werden soll (jeweils hinter der zuvor gefundenen beginnend), um jeweils anschließend wieder die mit der 1. Zahl angegebene Anzahl Zeichenfolgen zu ersetzen. Ist ERN nicht angegeben, so wird jeweils vom Anfang der Texteinheit an ersetzt. <1>
3. Zahl: Angabe, nach der wievielten gefundenen Zeichenfolge aus ERN erstmals ersetzt werden soll. <1>
4. Zahl: Angabe, ob eine unter ERS angegebene Zeichenfolge durch eine unter EZF angegebene ersetzt oder ob sie getilgt werden soll, falls in der Texteinheit unmittelbar danach wieder eine Zeichenfolge folgt, die unter ERS angegeben ist, oder falls die Zeichenfolge am Ende der Texteinheit steht. <0>
- 0 = Zeichenfolge nicht tilgen, sondern ersetzen  
1 = Zeichenfolge tilgen
5. Zahl: Angabe, wie oft der Parameter XXB unmittelbar hintereinander ausgeführt werden soll, wobei beim Wiederholen jeweils vom Ergebnis der vorangehenden Ausführung ausgegangen wird. <1>
- 0 = Ausführung wiederholen bis keine Zeichenfolgen mehr ersetzt werden können  
n = wie 0, jedoch maximal n Mal ausführen.

## Einsetzen von zuvor definierten Textteilen

Mit den folgenden Parametern kann ein zuvor definierter »Ersetzungstext« in den Arbeitstext eingesetzt werden. Der Ersetzungstext kann beim Programmstart mit dem Parameter ETV vorbelegt werden und während des Programmlaufs im Programmteil 6 jeweils neu definiert werden.

- PE** Anfangsposition zum Einsetzen des Ersetzungstextes.  
(Beschreibung des Parameters PE auf Seite 959)
- EEN** Zeichenfolgen, nach denen der Ersetzungstext eingesetzt werden soll.  
[ IX ]
- EEB** Zeichenfolgen, bis zu denen (ausschließlich) der Ersetzungstext eingesetzt werden soll. [ IX ]
- ETE** Zeichenfolgen, die durch den Ersetzungstext ersetzt werden sollen bzw. vor oder hinter denen der Ersetzungstext eingesetzt werden soll.  
[ IX ]
- EEZ** Zusatzangaben zum Einsetzen des Ersetzungstextes. [ 1 ]

Im folgenden ist mit »gefundene Zeichenfolge« eine Zeichenfolge gemeint, die mit dem Parameter ETE als Suchzeichenfolge angegeben ist,

und die im Arbeitstext bzw. in dem mit EEN und/oder EEB abgegrenzten Textteil gefunden wurde.

Es können zwei Zahlenwerte angegeben werden:

1. Zahl: Wie oft und wo einsetzen <0>

- 0 = der Ersetzungstext wird einmal (für die erste gefundene Zeichenfolge) eingesetzt. Der Parameter ETE ist in diesem Fall obligat. Wird keine unter ETE angegebene Zeichenfolge gefunden, so bleibt der Arbeitstext unverändert.
- 1 = der Ersetzungstext wird einmal (für die erste gefundene Zeichenfolge) eingesetzt; falls jedoch keine unter ETE angegebene Zeichenfolge gefunden wird oder der Parameter ETE nicht angegeben ist, wird der Ersetzungstext am Anfang des Arbeitstextes bzw. am Anfang des mit EEN abgegrenzten Textteils eingefügt.
- 2 = wie 1, jedoch wird der Ersetzungstext ggf. ans Ende des Arbeitstextes angehängt bzw. am Ende des mit EEB abgegrenzten Textteils eingefügt.
- 3 = wie 0, jedoch wird der Ersetzungstext ggf. nicht nur einmal, sondern für jede gefundene Zeichenfolge eingesetzt.
- 4 = Für jede gefundene Zeichenfolge wird ein Teilstück des Ersetzungstextes eingesetzt, für die erste gefundene das erste, für die zweite gefundene das zweite usw. Mit dem Parameter ETR müssen die Trennzeichen angegeben werden, durch die der Ersetzungstext in einzelne Teilstücke unterteilt wird.

2. Zahl: Ersetzen oder einfügen <0>

- 0 = Der Ersetzungstext soll jeweils die Zeichenfolge ersetzen.
- 1 = Der Ersetzungstext soll jeweils hinter der Zeichenfolge eingesetzt werden.
- 2 = Der Ersetzungstext soll jeweils vor der Zeichenfolge eingesetzt werden.

#### **ETR**

Zeichenfolgen, die als Trennzeichen zum Unterteilen des Ersetzungstextes dienen. [ IX ]

Wenn mit dem Parameter EEZ eine 4 als erster Zahlenwert angegeben ist, gilt:

Für jede gefundene Zeichenfolge wird ein Teilstück des Ersetzungstextes eingesetzt, für die erste gefundene das erste, für die zweite gefundene das zweite usw. Besteht der Ersetzungstext aus weniger Teilen als Zeichenfolgen gefunden werden, so werden leere Zeichenfolgen eingesetzt; besteht er aus mehr Teilen, so bleiben die restlichen unberücksichtigt.

In allen anderen Fällen gilt:

Besteht der Ersetzungstext aus mehr als einem Teil, so wird die Texteinheit für jeden weiteren Teil ans Ende kopiert und der entsprechende Teil des Ersetzungstextes eingesetzt.

## Einsetzen einer laufenden Nummer

**PE** Anfangsposition zum Einsetzen der laufenden Nummer.  
(Beschreibung des Parameters **PE** auf Seite 959)

**LNR** Zeichenfolgen, hinter denen die laufende Nummer eingesetzt werden soll. [ IX ]

Der Anfangswert der laufenden Nummer kann mit dem Parameter **LNB** (siehe Seite 936) bestimmt werden.

Wenn als laufende Nummer nicht die Ausgabennummer (= 1 + Anzahl der bereits ausgegebenen Texteinheiten), sondern eine separat laufende Nummer eingesetzt werden soll, so muss dies mit dem 2. Zahlenwert des folgenden Parameters **LNZ** angegeben werden.

**LNZ** Zusatzangaben zu **LNR**. [ I ]

Es können vier Zahlenwerte angegeben werden:

1. Zahl: <0>

- 0 = Die laufende Nummer soll einmal eingesetzt werden (falls der Parameter **LNR** angegeben ist)
- 1 = Die laufende Nummer soll beliebig oft eingesetzt werden.
- 2 = Wie 1, jedoch soll die laufende Nummer, nachdem sie einmal in die Texteinheit eingesetzt wurde, vor jedem weiteren Einsetzen um 1 erhöht werden (nicht möglich, wenn die 2. Zahl dieses Parameters Null ist).

2. Zahl: <0>

- 0 = Anstatt der laufenden Nummer wird die Ausgabennummer (= 1 + Anzahl der bereits ausgegebenen Texteinheiten) eingesetzt.
- 1 = Laufende Nummer (vor dem Einsetzen) um 1 erhöhen, falls sie eingesetzt wird bzw. eingesetzt würde, falls sie durch die 4. Zahl dieses Parameters ohne Rest teilbar wäre.
- 2 = Laufende Nummer nicht verändern
- 3 = Laufende Nummer um 1 erhöhen (unabhängig davon, ob sie eingesetzt wird oder nicht; ggf. vor dem Einsetzen)
- 4 = Laufende Nummer auf 0 setzen (ggf. vor dem Einsetzen)
- 5 = Laufende Nummer auf 1 setzen (ggf. vor dem Einsetzen)

3. Zahl: Anzahl der Dezimalstellen <0>

Laufende Nummer links auf die angegebene Anzahl Stellen mit Nullen auffüllen.

## 4. Zahl: Schrittweite &lt;1&gt;

Laufende Nummer nur einsetzen, falls sie durch die angegebene Zahl ohne Rest teilbar ist.

**Einsetzen / Lesen der Seiten-Zeilen-Nummer**

- PE** Anfangsposition zum Einsetzen der Seiten-Zeilen-Nummer.  
(Beschreibung des Parameters PE auf Seite 959)
- SNR** Zeichenfolgen, hinter denen die Seiten- und ggf. Zeilennummer eingesetzt werden soll. [ IX ]
- SNZ** Zusatzangaben zu SNR. [ I ] <0, 0, 0>  
Es können drei Zahlenwerte angegeben werden:
1. Zahl: Wie oft einsetzen <0>
    - 0 = Die Seiten(-Zeilen)nummer soll einmal eingesetzt werden
    - 1 = Die Seiten(-Zeilen)nummer soll beliebig oft eingesetzt werden.
  2. Zahl: Was einsetzen <0>
    - 0 = Nur die Seitennummer einsetzen
    - 1 = Seiten-Zeilen-Nummer einsetzen
    - 2 = Seiten-Zeilen-Unterscheidungsnummer einsetzen
  3. Zahl: Nummer der ZIEL-Datei <0>

Falls dieser Zahlenwert ungleich 0 ist, wird die Seiten(-Zeilen)nummer nicht eingesetzt, sondern gelesen. Wird die Zeilennummer nicht gelesen (wenn die 2. Zahl dieses Parameters 0 ist), erhält sie den Wert 1; wird die Unterscheidungsnummer nicht gelesen (wenn die 2. Zahl dieses Parameters 0 oder 1 ist), erhält sie den Wert 0. Diese Seiten(-Zeilen)nummer erhält dann der nächste Satz, der in die n-te ZIEL-Datei (n = angegebene Zahl) ausgegeben wird, falls der zuletzt in diese Datei ausgegebene nicht schon eine höhere Nummer hatte.

**Definieren von Positionen (Spalten)**

- POS** Positionen, falls die Texteinheit nicht von der 1. Zeichenposition an verarbeitet werden soll, oder falls Zeichen von bestimmten Zeichenpositionen entfernt werden sollen. [ I ]  
Es können drei Zahlenwerte angegeben werden:
1. Zahl: Anfangsposition <1>

Anfangsposition, ab der die Texteinheit bearbeitet werden soll.

2. Zahl: Endposition <999999999>

Endposition, bis zu der die Zeichen erhalten bzw. eliminiert werden sollen (wirkungslos, wenn 3. Zahl Null ist)

3. Zahl: Modus <0>

0 = Die Zeichen auf Position 1 bis ausschließlich der mit der 1. Zahl angegebenen Position werden unverändert übernommen.

1 = Die Zeichen vor der mit der 1. Zahl angegebenen Position und nach der mit der 2. Zahl angegebenen Position werden am Ende der Verarbeitung eliminiert.

2 = Die Zeichen auf der mit der 1. Zahl angegebenen Position bis einschließlich der mit der 2. Zahl angegebenen Position werden am Ende der Verarbeitung eliminiert.

Achtung: Die 1. Zahl dieses Parameters wird auch als Voreinstellung für die Parameter PV, PK, PL, PX, PE übernommen. Falls dies nicht erwünscht ist, müssen diese Parameter ggf. mit der entsprechenden Positionsangabe (z. B. 1) zusätzlich angegeben werden.

### PROGRAMMTEIL 3 (Ausgabe, mit Durchgangsangabe):

#### Ausdrucken von Meldungen

**MLD** Textteile, die als Meldung zusammen mit dem Arbeitstext ins Ablaufprotokoll ausgegeben werden sollen. [ II ]

Jeder Textteil wird in eine eigene Zeile gedruckt.

Falls der Parameter MIV nicht angegeben ist, und die Meldung zwei oder mehr aufeinander folgende »x« enthält, so werden diese durch den Wert der Variablen S9 (vgl. Seite 985) ersetzt.

Falls der Parameter MLD angegeben ist, unterbleibt die Ausgabe der Texteinheit in die ZIEL-Datei, wenn nicht zusätzlich der Parameter ZD angegeben ist.

Beginnt der erste Textteil mit dem Zeichen »@«, so wird angenommen, dass es sich um eine Fehlermeldung handelt. Das Programm wird trotzdem fortgesetzt, aber am Ende mit Fehler beendet. Dies ist von Bedeutung, wenn Fehlerhalt (siehe Kommando #FEHLERHALT) eingeschaltet ist.

Beginnt der erste Textteil mit »@@@« , so wird die Meldung in der Farbe ausgegeben, die für Warnungen eingestellt ist.

Beginnt der erste Textteil mit »@@@@@@@@ «, so wird die Meldung in der Farbe ausgegeben, die für Fehlermeldungen eingestellt ist.

Alle anders beginnenden Meldungen werden in der Farbe ausgegeben, die für Parameterausgabe eingestellt ist.

Hinweis: Die Farben können mit dem Kommando #DEFINIERE eingestellt werden.

**MIV** I-Variablen, deren Werte in den unter MLD angegebenen Text vor der Ausgabe eingesetzt werden sollen. [ I ]

Die Werte werden in der angegebenen Reihenfolge an den Stellen eingesetzt, die mit einer Folge von mindestens zwei »x« gekennzeichnet sind. Sind mehr I-Variablen angegeben als mit »x« gekennzeichnete Stellen vorhanden sind, so bleiben die restlichen Werte unberücksichtigt.

**MDN** Angaben parallel zu den mit dem Parameter MIV angegebenen I-Variablen, wieviele Stellen jeweils nach dem Komma stehen sollen. [ I ]  
<0, 0, 0, . . . >

### Bestimmen der ZIEL-Datei

**ZD** Nummer der ZIEL-Datei, in die in diesem Durchgang der Arbeitstext ausgegeben werden soll. [ I ] <1>

Die Nummern der ZIEL-Dateien werden durch Abzählen bestimmt. Die erste erhält die Nummer 1, die zweite die Nummer 2, usw.

### PROGRAMMTEIL 3 (Ausgabe, ohne Durchgangsangabe):

#### Bestimmen von Zeilenanfang und -ende

Bei der Ausgabe kann der Arbeitstext in Zeilen eingeteilt werden. Dazu können Zeichenfolgen angegeben werden, vor bzw. nach denen jeweils eine neue Zeile begonnen werden soll.

**ZA** Zeichenfolgen, vor denen jeweils eine neue Zeile begonnen werden soll (Zeilenanfänge). [ IX ]

**ZE** Zeichenfolgen, nach denen jeweils eine neue Zeile begonnen werden soll (Zeilenenden). [ IX ]

Zeilen, die nach Abarbeiten der Parameter ZA und ZE so lang sind, dass sie auf Grund der Angaben mit dem Parameter SL (s. u.) in mehrere Sätze aufgeteilt werden müssen, können auch vor oder nach bestimmten Zeichenfolgen in mehrere Zeilen aufgeteilt werden, indem diese Zeichenfolgen mit einem der beiden folgenden Parameter angegeben werden. Eine Zeile wird bei einer solchen Zeichenfolge nur dann



aufgeteilt, wenn es nicht genügt, sie erst bei der nächsten aufzuteilen, um Zeilen der gewünschten Maximallänge zu erhalten.

**ZAB** Zeichenfolgen, die Zeilenanfänge kennzeichnen, vor denen bei Bedarf eine neue Zeile begonnen werden soll. [ IX ]

**ZEB** Zeichenfolgen, die Zeilenenden kennzeichnen, nach denen bei Bedarf eine neue Zeile begonnen werden soll. [ IX ]

## Bestimmen von Seitenanfang und -ende

Bei der Ausgabe kann bei Zeilen, die mit einer bestimmten Kennung beginnen, oder nach Zeilen, die mit einer bestimmten Kennung enden, die Nummerierung der Sätze auf die nächste Seite fortgeschaltet werden.

**SA** Zeichenfolgen, die als Kennung der Zeilen dienen, bei denen auf die nächste Seitennummer fortgeschaltet werden soll. [ VIII-a ]

Die Kennung wird nur erkannt, wenn sie unmittelbar am Anfang der Zeile steht. Ggf. kann dies mit Hilfe des Parameters ZA erreicht werden.

**SE** Zeichenfolgen, die als Kennung der Zeilen dienen, nach denen die Seitennummer fortgeschaltet werden soll. [ VIII-b ]

Die Kennung wird nur erkannt, wenn sie unmittelbar am Ende der Zeile steht. Ggf. kann dies mit Hilfe des Parameters ZE erreicht werden.

Falls für eine Zeile sowohl auf Grund des Parameters SE als auch auf Grund des Parameters SA die Seitennummer fortgeschaltet werden müsste, wird nur einmal fortgeschaltet.

## Einfügen von Leerzeilen

Bei der Ausgabe kann vor Zeilen, die mit einer bestimmten Kennung beginnen, oder nach Zeilen, die mit einer bestimmten Kennung enden, eine Leerzeile eingefügt werden.

**LZV** Zeichenfolgen, die als Kennung der Zeilen dienen, vor denen eine Leerzeile eingefügt werden soll. [ VIII-a ]

Die Kennung wird nur erkannt, wenn sie unmittelbar am Anfang der Zeile steht. Ggf. kann dies mit Hilfe des Parameters ZA erreicht werden.

Falls vor einer Zeile, die mit einer angegebenen Kennung beginnt, schon eine Leerzeile steht, wird keine weitere Leerzeile eingefügt.

**LZN** Zeichenfolgen, die als Kennung der Zeilen dienen, nach denen eine Leerzeile eingefügt werden soll. [ VIII-b ]

Die Kennung wird nur erkannt, wenn sie unmittelbar am Ende der Zeile steht. Ggf. kann dies mit Hilfe des Parameters ZE erreicht werden.

Falls zwischen zwei Zeilen sowohl auf Grund des Parameters LZV als auch auf Grund

des Parameters LZN eine Leerzeile eingefügt werden müsste, wird nur eine Leerzeile (statt zwei Leerzeilen) eingefügt.

## Austauschen von Zeichenfolgen

**XXX** Zeichenfolgenpaare (und Ausnahmezeichenfolgen), deren jeweils erste Zeichenfolge durch die jeweils zweite Zeichenfolge bei der Ausgabe ersetzt werden soll. [ X ]

## Unterdrücken von Leerzeichen am Zeilenanfang und -ende

**BLU** Angabe, ob Leerzeichen am Zeilenanfang bzw. am Zeilenende vor der Ausgabe entfernt werden sollen. [ I ] <0, 0>

Es können zwei Zahlenwerte angegeben werden:

1. Zahl: Leerzeichen am Zeilenanfang

- 0 = Am Zeilenanfang keine Leerzeichen entfernen
- 1 = Am Zeilenanfang Leerzeichen entfernen

2. Zahl: Leerzeichen am Zeilenende

- 0 = Am Zeilenende keine Leerzeichen entfernen
- 1 = Am Zeilenende Leerzeichen entfernen

## Unterdrücken von Leerzeilen

**LZU** Angabe, ob Leerzeilen unterdrückt werden sollen. [ I ] <0>

Es kann ein Zahlenwert angegeben werden:

- 0 = Leerzeilen werden ausgegeben
- 1 = Leerzeilen werden unterdrückt
- 2 = Unmittelbar aufeinander folgende Leerzeilen werden zu einer zusammengefasst.

Leerzeilen, die mit dem Parameter LZV oder LZN eingefügt wurden, werden in jedem Fall ausgegeben.

## Bestimmen der Satzlänge

**SL** Satzlänge der Ausgabesätze. [ I ]

Falls dieser Parameter nicht angegeben ist und die Zeileneinteilung von den Eingabedaten nicht übernommen werden kann, wird eine Zeile nicht unterteilt.

Es können zwei Zahlenwerte angegeben werden:

1. Zahl: Maximallänge für Sätze, in die diejenigen Zeilen unterteilt werden sollen, die länger sind, als mit der zweiten Zahl dieses Parameters angegeben ist. <100>
2. Zahl: Maximallänge für Zeilen, die nicht unterteilt werden sollen. <120 bzw. Wert der 1. Zahl, falls diese größer als 120 ist>

Vor der Ausgabe wird eine Zeile, die mehr Zeichen enthält, als mit dem 2. Zahlenwert angegeben ist, in mehrere Ausgabesätze unterteilt. Die Zeile wird so unterteilt, dass die Ausgabesätze maximal so lang sind, wie mit dem 1. Zahlenwert angegeben ist. Als Trennstelle wird ein Leerzeichen ausserhalb von Tags gewählt, vor dem kein »-« steht, das mit einem Silbentrennzeichen verwechselt werden kann. Falls innerhalb der mit dem 1. Zahlenwert angegebenen Anzahl von Zeichen keine solche Trennstelle vorhanden ist, wird die nächstmögliche Trennstelle gesucht.

Mit dem nachfolgend beschriebenen Parameter **SLZ** kann erlaubt werden, dass auch an Leerstellen innerhalb von Tags unterteilt werden darf.

**SLZ** Angabe, ob auf Grund des Parameters **SL** Zeilen auch an Leerzeichen innerhalb von Tags (z. B. <tag attribut="wert">) unterteilt werden dürfen. [I] <0>

Es kann ein Zahlenwert angegeben werden:

- 0 = Tags dürfen nicht unterteilt werden
- 1 = Tags dürfen unterteilt werden.

## Bestimmen der Nummerierung der Sätze

**NR** Angaben zur Nummerierung der Ausgabesätze. [I]

Es können drei Zahlenwerte angegeben werden:

1. Zahl: Seitennummer, mit der bei der Ausgabe begonnen werden soll bzw. 999999, falls mit der nächsten freien Seite begonnen werden soll. <MODUS=-STD-: 0; sonst: 999999>
2. Zahl: Maximale Anzahl der Sätze je Seite. <MODUS=-STD-: 1000000; sonst: 60>
3. Zahl: Schrittweite der Nummerierung, wenn innerhalb einer Texteinheit eine neue Satznummer vergeben werden muss. <MODUS=-STD-: 10; sonst: 1000>

**PROGRAMMTEIL 4** (Arbeitstext ==> Grundtext):**Umdefinieren des Grundtextes**

keine Parameter

**PROGRAMMTEIL 5** (Vergleichs-/Arbeitstext ==> Merk-Vergleichstext):**Definieren von Merk-Vergleichstexten**

**VTZ**            Angabe, für welche Durchgänge der in diesem Durchgang abgespeicherte Merk-Vergleichstext gelten soll. [ I ]

**PROGRAMMTEIL 6** (Arbeitstext ==> Ersetzungstext):**Definieren von Ersetzungstexten**

**ETZ**            Angabe, für welche Durchgänge der in diesem Durchgang abgespeicherte Ersetzungstext gelten soll. [ I ]

**PROGRAMMTEIL 7** (Arbeitstext ==> Merkttext):**Ersetzen/Ergänzen des Merkttextes durch den Arbeitstext**

Welcher Merkttext ersetzt/ergänzt wird, kann mit der Sondervariablen S17 (vgl. Seite 985) festgelegt werden.

**MTD**            Angabe zur Definition des Merkttextes. [ I ] <0>

Es kann ein Zahlenwert angegeben werden:

- 0 = Merkttext wird durch Arbeitstext ersetzt
- 1 = Arbeitstext wird an den bisherigen Merkttext angehängt.
- 2 = Arbeitstext wird vor den bisherigen Merkttext eingeschoben.

Die folgenden Auswahlparameter sind nur notwendig, wenn nicht der gesamte Arbeitstext unverändert in den Merkttext übernommen werden soll.

<b>AMT</b>	Zeichenfolgen, die den Anfang des Textteils im Arbeitstext kennzeichnen, der in den Merktex übernommen werden soll. [ IX ]
<b>EMT</b>	Zeichenfolgen, die das Ende des Textteils im Arbeitstext kennzeichnen, der in den Merktex übernommen werden soll. [ IX ]
<b>(MT</b>	Zeichenfolgen, die bei der Auswahl des Textes, der in den Merktex übernommen werden soll (falls AMT und/oder EMT nicht angegeben), bzw. die bei der Eliminierung von Textteilen aus dem bereits mit AMT und/oder EMT ausgewählten Teil als öffnende Klammer dienen sollen. [ IX ]  Die Wirkung der öffnenden Klammer wird mit dem zweiten Zahlenwert des Parameters MTI bestimmt.
<b>)MT</b>	Zeichenfolgen, die bei der Auswahl des Textes, der in den Merktex übernommen werden soll (falls AMT und/oder EMT nicht angegeben), bzw. die bei der Eliminierung von Textteilen aus dem bereits mit AMT und/oder EMT ausgewählten Teil als schließende Klammer dienen sollen. [ IX ]  Die Wirkung der schließenden Klammer wird mit dem zweiten Zahlenwert des Parameters MTI bestimmt.
<b>MTI</b>	Index für die Parameter AMT, EMT, (MT und )MT. [ I ]  Es können zwei Zahlenwerte angegeben werden: 1. Zahl: Angabe analog zum Parameter AEI (siehe Seite 954) <1> 2. Zahl: Angabe analog zum Parameter KLI (siehe Seite 955) <0>
<b>XMT</b>	Zeichenfolgenpaare (und Ausnahmezeichenfolgen), deren jeweils erste Zeichenfolge durch die jeweils zweite Zeichenfolge in dem Text ersetzt werden soll, der in den Merktex übernommen werden soll. [ X ]

## PROGRAMMTEIL 8 (Merktex <==> Arbeitstex):

### Austauschen von Merktex und Arbeitstex

Welcher Merktex ausgetauscht wird, kann mit der Sondervariablen S18 (vgl. Seite 986) festgelegt werden.

keine Parameter

**PROGRAMMTEIL 9** (Merktext ==> Arbeitstext):

Welcher Merktext geholt wird, kann mit der Sondervariablen S19 (vgl. Seite 986) festgelegt werden.

**Ersetzen/Ergänzen des Arbeitstextes durch den Merktext**

**MTH** Angabe zum Holen des Merktexes. [ I ] <0>

Es kann ein Zahlenwert angegeben werden:

- 0 = Arbeitstext wird durch Merktext ersetzt
- 1 = Merktext wird an den bisherigen Arbeitstext angehängt.
- 2 = Merktext wird vor den bisherigen Arbeitstext eingeschoben.

Falls der Parameter MTP angegeben ist und eine entsprechende Zeichenfolge im Arbeitstext gefunden wird, hat der Zahlenwert folgende Bedeutung:

- 0 = Gefundene Zeichenfolge wird durch Merktext ersetzt
- 1 = Merktext wird unmittelbar hinter der gefundenen Zeichenfolge eingefügt.
- 2 = Merktext wird unmittelbar vor der gefundenen Zeichenfolge eingefügt.

Wird auf diese Werte 3 aufaddiert (also 3, 4 oder 5 angegeben), so wird nach der Übertragung des Merktexes der Merktext gelöscht.

Soll der Merktext an einer bestimmten Stelle innerhalb des Arbeitstextes eingefügt werden, so kann diese Stelle mit folgendem Parameter bestimmt werden.

**MTP** Zeichenfolgen, die die Position der Stelle kennzeichnen, an der der Merktext in den Arbeitstext eingefügt werden soll. [ IX ]

Falls im Arbeitstext entsprechende Zeichenfolgen vorhanden sind, so wird an der ersten im Arbeitstext gefundenen Zeichenfolge der Merktext entsprechend den Angaben zum Parameter MTH eingefügt; andernfalls wird so verfahren wie wenn der Parameter MTP nicht angegeben wäre.

Soll nicht der ganze Merktext geholt und ggf. gelöscht werden, kann mit den folgenden Parametern ein Teil davon ausgewählt werden.

**MTA** Zeichenfolgen, die den Anfang des Textteils kennzeichnen, der aus dem Merktext geholt und dort ggf. gelöscht werden soll. [ IX ]

**MTE** Zeichenfolgen, die das Ende des Textteils kennzeichnen, der aus dem Merktext geholt und dort ggf. gelöscht werden soll. [ IX ]

Sind beide Parameter MTA und MTE angegeben, so beginnt der ausgewählte Textteil mit der ersten im Merktext gefundenen Anfangskennung und endet vor der nachfolgenden Endekennung bzw. am Ende der Texteinheit.

Ist nur der Parameter *MTA* angegeben, so beginnt der ausgewählte Textteil mit der letzten im Merkttext vorkommenden Anfangskennung und endet am Ende des Merkttextes.

Ist nur der Parameter *MTE* angegeben, so beginnt der ausgewählte Textteil am Anfang des Merkttextes und endet nach der ersten im Merkttext vorkommenden Endekennung.

## ABSCHLUSS

### Ausgeben eines Endetextes und einer Endemeldung

**RRR** Rechenanweisungen zum Berechnen der Endergebnisse, die mit den nachfolgend beschriebenen Parametern ausgegeben werden können. [ XI ]

Beschreibung siehe »Rechenanweisungen« Seite 983 ff.

**ZZZ** Text, der in die ZIEL-Datei ausgegeben werden soll. Bei jedem Trennzeichen wird ein neuer Satz begonnen. [ II ]

Ist mehr als eine ZIEL-Datei vorhanden, wird die erste Zeichenfolge in die erste ZIEL-Datei, die zweite Zeichenfolge in die zweite ZIEL-Datei, usw. ausgegeben. Leere Zeichenfolgen werden nicht in die entsprechenden ZIEL-Dateien ausgegeben. Sind mehr Zeichenfolgen angegeben als ZIEL-Dateien vorhanden sind, werden die weiteren Zeichenfolgen reihum wieder in die erste, zweite, usw. ausgegeben.

**ZIV** I-Variablen, deren Werte in den unter *ZZZ* angegebenen Text vor der Ausgabe eingesetzt werden sollen. [ I ]

Die Werte werden in der angegebenen Reihenfolge an den Stellen eingesetzt, die mit einer Folge von mindestens zwei »x« gekennzeichnet sind. Es müssen jeweils mindestens soviele »x« angegeben werden, dass der Wert der entsprechenden I-Variablen dafür eingesetzt werden kann; reichen die Stellen nicht aus, so bleiben die »x« im Text stehen. Sind mehr I-Variablen angegeben als mit »x« gekennzeichnete Stellen vorhanden sind, so bleiben die restlichen Werte unberücksichtigt.

**ZDN** Angaben parallel zu den mit dem Parameter *ZIV* angegebenen I-Variablen, wieviele Stellen jeweils nach dem Komma stehen sollen. [ I ]  
<0,0,0,...>

**DDD** Text, der ins Protokoll ausgegeben werden soll. Bei jedem Trennzeichen wird eine neue Zeile begonnen; das erste Zeichen nach dem Trennzeichen wird als Vorschubzeichen interpretiert. [ II ]

**DIV** I-Variablen, deren Werte in den unter *DDD* angegebenen Text vor dem Drucken eingesetzt werden sollen. [ I ]

Die Werte werden in der angegebenen Reihenfolge an den Stellen eingesetzt, die mit einer Folge von mindestens zwei »x« gekennzeichnet sind. Es müssen jeweils mindestens soviele »x« angegeben werden, dass der Wert der entsprechenden I-Variablen dafür eingesetzt werden kann; reichen die Stellen nicht aus, so bleiben die »x« im Text stehen. Sind mehr I-Variablen angegeben als mit »x« gekennzeichnete Stellen vorhanden sind, so bleiben die restlichen Werte unberücksichtigt.

**DDN** Angaben parallel zu den mit dem Parameter DIV angegebenen I-Variablen, wieviele Stellen jeweils nach dem Komma stehen sollen. [ I ]  
<0, 0, 0, ...>

### Retten der Endwerte von Variablen

**HVR** Name einer TUSTEP-Variablen, die definiert wird und als Inhalt die Zahlenwerte der H-Variablen erhält. [ XI ]

Die einzelnen Zahlenwerte der Variablen H0, H1, ..., H9 werden durch Apostroph getrennt. Abschließende Null-Werte werden weggelassen.

### Retten der Wahlschalter auf die Kommandoebene

**WSR** Wahlschalter, deren Stellung auf die Wahlschalter der Kommandoebene (das sind die Wahlschalter, die mit dem Kommando #WAHLSCHALTER gesetzt bzw. gelöscht werden können) übertragen werden sollen (Wahlschalter retten). [ I ]

Es können bis zu 7 Zahlenwerte (Nummern der Wahlschalter 1 bis 7) angegeben werden. Wahlschalter, die nicht angegeben sind, bleiben unverändert.



## Alphabetisches Verzeichnis der Parameter

Das Zeichen »n« in den Kennungen der Parameter steht für die Ziffern 1 bis 9 (z. B. steht AKn für AK1, AK2 bis AK9).

<b>(Kn</b>	Kopieren: Ausklammern von Textteilen . . . . .	955
<b>(L</b>	Lesen von Zahlen: Ausklammern von Textteilen . . . . .	958
<b>(MT</b>	Merktext definieren: Ausklammern von Textteilen . . . . .	973
<b>(TT</b>	Textteile tauschen: Ausklammern von Textteilen . . . . .	956
<b>(V</b>	Vergleich: Ausklammern von Textteilen . . . . .	945
<b>(XX</b>	Ersetzen: Ausklammern von Textteilen . . . . .	961
<b>)Kn</b>	Kopieren: Ausklammern von Textteilen . . . . .	955
<b>)L</b>	Lesen von Zahlen: Ausklammern von Textteilen . . . . .	958
<b>)MT</b>	Merktext definieren: Ausklammern von Textteilen . . . . .	973
<b>)TT</b>	Textteile tauschen: Ausklammern von Textteilen . . . . .	956
<b>)V</b>	Vergleich: Ausklammern von Textteilen . . . . .	946
<b>)XX</b>	Ersetzen: Ausklammern von Textteilen . . . . .	962
<b>AA</b>	Anfang einer Texteinheit (Abschnittsanfang) . . . . .	941
<b>AAZ</b>	Zusatzangaben zum Parameter AA . . . . .	941
<b>AE</b>	Ende einer Texteinheit (Abschnittsende) . . . . .	941
<b>AEI</b>	Kopieren: Index für AKn und EKn . . . . .	954
<b>AEZ</b>	Zusatzangaben zum Parameter AE . . . . .	942
<b>AKn</b>	Kopieren: Anfangskennung . . . . .	954
<b>AL</b>	Lesen von Zahlen: Anfangskennung . . . . .	958
<b>ALZ</b>	Bilden einer Texteinheit (Abschnitt) bei Leerzeilen . . . . .	940
<b>AMT</b>	Merktext definieren: Anfangskennung . . . . .	973
<b>ANR</b>	Bilden einer Texteinheit (Abschnitt) nach Nummern . . . . .	940
<b>ATT</b>	Textteile tauschen: Anfangskennung . . . . .	956
<b>AUM</b>	Umdrehen: Anfangskennung . . . . .	957
<b>AV</b>	Vergleich: Anfangskennung . . . . .	945
<b>AXX</b>	Ersetzen: Anfangskennung . . . . .	961
<b>AZ+</b>	Positiv-Auswahl über Zeichenfolgen im Vergleichstext . . . . .	948
<b>AZ-</b>	Negativ-Auswahl über Zeichenfolgen im Vergleichstext . . . . .	949
<b>BER</b>	Auswahl eines Bereichs aus der QUELL-Datei . . . . .	939
<b>BLU</b>	Leerzeichen unterdrücken . . . . .	970
<b>DDD</b>	Drucken von Endemeldungen: Text . . . . .	975
<b>DDN</b>	Drucken von Endemeldungen: Dezimalst. nach dem Komma . . . . .	976
<b>DIV</b>	Drucken von Endemeldungen: I-Variablen . . . . .	975
<b>DPB</b>	Druckpositionen berechnen . . . . .	958
<b>EDN</b>	Einsetzen von Zahlen: Dezimalstellen nach dem Komma . . . . .	960
<b>EDV</b>	Einsetzen von Zahlen: Dezimalstellen vor dem Komma . . . . .	960
<b>EEB</b>	Ersetzungstext nur bis zu best. Zeichenflg. einsetzen . . . . .	963
<b>EEN</b>	Ersetzungstext nur nach best. Zeichenfolgen einsetzen . . . . .	963
<b>EEZ</b>	Zusatzangaben zum Einsetzen des Ersetzungstextes . . . . .	963
<b>EIB</b>	Einsetzen von Zahlen: Bis zu bestimmten Zeichenfolgen . . . . .	959
<b>EIK</b>	Einsetzen von Zahlen: Kennzeichen . . . . .	961
<b>EIN</b>	Einsetzen von Zahlen: Nach bestimmten Zeichenfolgen . . . . .	959
<b>EIV</b>	Einsetzen von Zahlen: I-Variablen . . . . .	959

<b>EIZ</b>	Einsetzen von Zahlen: Zusätzliche Angaben . . . . .	960
<b>EKn</b>	Kopieren: Endekennung . . . . .	954
<b>EL</b>	Lesen von Zahlen: Endekennung . . . . .	958
<b>ELS</b>	Schwellenwert für Endlosschleife . . . . .	938
<b>EMT</b>	Merktext definieren: Endekennung . . . . .	973
<b>ERB</b>	Ersetzen: Bis zu bestimmten Zeichenfolgen . . . . .	962
<b>ERG</b>	Kopieren: Ergänzen von Zeichenfolgen . . . . .	956
<b>ERN</b>	Ersetzen: Nach bestimmten Zeichenfolgen . . . . .	962
<b>ERS</b>	Ersetzen: Zu ersetzende Zeichenfolgen . . . . .	962
<b>ERZ</b>	Ersetzen: Zusatzangaben . . . . .	962
<b>ETE</b>	Ersetzungstext für best. Zeichenfolgen einsetzen . . . . .	963
<b>ETR</b>	Ersetzungstext durch Trennzeichen unterteilen . . . . .	964
<b>ETT</b>	Textteile tauschen: Endekennung . . . . .	956
<b>ETV</b>	Ersetzungstext: Vorbelegung . . . . .	937
<b>ETZ</b>	Ersetzungstext: Zusatzangaben . . . . .	972
<b>EUM</b>	Umdrehen: Endekennung . . . . .	957
<b>EV</b>	Vergleich: Endekennung . . . . .	945
<b>EXX</b>	Ersetzen: Endekennung . . . . .	961
<b>EZ+</b>	Positiv-Auswahl über Zeichenfolgen im Vergleichstext . . . . .	948
<b>EZ-</b>	Negativ-Auswahl über Zeichenfolgen im Vergleichstext . . . . .	949
<b>EZF</b>	Ersetzen: Einzusetzende Zeichenfolgen . . . . .	962
<b>GTU</b>	Grundtext unterteilen . . . . .	944
<b>GTZ</b>	Zusatzangaben zum Parameter GTU . . . . .	944
<b>HVH</b>	H-Variablen holen . . . . .	937
<b>HVR</b>	H-Variablen retten . . . . .	976
<b>KEN</b>	Kennung von Texteinheiten . . . . .	945
<b>KLI</b>	Kopieren: Index für (Kn und ) Kn . . . . .	955
<b>KSP</b>	Sprung in Abhängigkeit von einer Kennung . . . . .	945
<b>LDN</b>	Lesen von Zahlen: Dezimalstellen nach dem Komma . . . . .	959
<b>LI</b>	Lesen von Zahlen: Index für AL, EL, (L und ) L . . . . .	958
<b>LIV</b>	Lesen von Zahlen: I-Variablen . . . . .	958
<b>LIZ</b>	Lesen von Zahlen: Arabische oder römische Zahlen . . . . .	959
<b>LNB</b>	Basis (Anfangswert) für die laufende Nummer . . . . .	936
<b>LNR</b>	Einsetzen einer laufende Nummer . . . . .	965
<b>LNZ</b>	Zusatzangaben zum Einsetzen einer laufende Nummer . . . . .	965
<b>LZN</b>	Leerzeile nach bestimmten Kennungen . . . . .	969
<b>LZU</b>	Leerzeilen unterdrücken . . . . .	970
<b>LZV</b>	Leerzeile vor bestimmten Kennungen . . . . .	969
<b>MAX</b>	Maximum für Testzwecke . . . . .	940
<b>MDN</b>	Meldung ausgeben: Dezimalstellen nach dem Komma . . . . .	968
<b>MIV</b>	Meldung ausgeben: I-Variablen . . . . .	968
<b>MLD</b>	Meldung ausgeben . . . . .	967
<b>MTA</b>	Merktext holen: Anfangskennung . . . . .	974
<b>MTD</b>	Merktext definieren . . . . .	972
<b>MTE</b>	Merktext holen: Endekennung . . . . .	974
<b>MTH</b>	Merktext holen . . . . .	974
<b>MTI</b>	Merktext: Index für AMT, EMT, (MT und ) MT . . . . .	973
<b>MTP</b>	Merktext holen: Position zum Einfügen . . . . .	974

<b>MTV</b>	Merktext vorbelegen . . . . .	937
<b>NR</b>	Nummerierung der Sätze . . . . .	971
<b>NR+</b>	Auswahl auf Grund von Satznummern aus der QUELL-Datei . . . . .	939
<b>NR-</b>	Auswahl auf Grund von Satznummern aus der QUELL-Datei . . . . .	939
<b>NUM</b>	Umdrehen: Nicht umzudrehende Zeichenfolgen . . . . .	957
<b>PAR</b>	Parameter-Konvention . . . . .	934
<b>PE</b>	Anfangsposition zum Einsetzen . . . . .	959
<b>PK</b>	Anfangsposition zum Kopieren . . . . .	954
<b>PL</b>	Anfangsposition zum Lesen von Zahlen . . . . .	958
<b>POS</b>	Positionsangaben generell . . . . .	966
<b>PV</b>	Anfangsposition zum Vergleichen . . . . .	944
<b>PX</b>	Anfangsposition zum Ersetzen . . . . .	961
<b>PZP</b>	Vergleich: Prüfwerte prüfen . . . . .	953
<b>R</b>	Rechenanweisungen zur Vorbesetzung der Variablen . . . . .	937
<b>RR</b>	Rechen-, Abfrage- und Sprunganweisungen . . . . .	959
<b>RRR</b>	Rechenanweisungen zum Berechnen der Endergebnisse . . . . .	975
<b>SA</b>	Seitenanfangs-Kennung . . . . .	969
<b>SE</b>	Seitenende-Kennung . . . . .	969
<b>SL</b>	Satzlänge der Ausgabesätze . . . . .	970
<b>SLZ</b>	Satzlänge: Zusatzangaben . . . . .	971
<b>SNR</b>	Einsetzen der Seitennummer . . . . .	966
<b>SNZ</b>	Zusatzangaben zum Einsetzen der Seitennummer . . . . .	966
<b>SPn</b>	Sprungtabelle für Sprung nach Programmteil n . . . . .	938
<b>SPJ</b>	Sprungtabelle für Sprung bei erfüllter Bedingung . . . . .	938
<b>SPN</b>	Sprungtabelle für Sprung bei nicht erfüllter Bedingung . . . . .	938
<b>SPR</b>	Sprungtabelle für die einzelnen Durchgänge . . . . .	938
<b>SPW</b>	Sprungtabelle für Sprung bei Wahlschalter-Abfrage . . . . .	938
<b>STE</b>	Silbentrennzeichen-Ersatz . . . . .	943
<b>STR</b>	Silbentrennung aufheben . . . . .	942
<b>SWS</b>	Sonderwahlschalter . . . . .	935
<b>T</b>	Text für den Arbeitstext . . . . .	954
<b>T+</b>	Positiv-Auswahl über den ganzen Vergleichstext . . . . .	947
<b>T+N</b>	Positiv-Auswahl über den ganzen Vergleichstext . . . . .	947
<b>T+U</b>	Positiv-Auswahl über den ganzen Vergleichstext . . . . .	947
<b>T-</b>	Negativ-Auswahl über den ganzen Vergleichstext . . . . .	947
<b>T-N</b>	Negativ-Auswahl über den ganzen Vergleichstext . . . . .	947
<b>T-U</b>	Negativ-Auswahl über den ganzen Vergleichstext . . . . .	947
<b>TA+</b>	Positiv-Auswahl über den Anfang des Vergleichstextes . . . . .	948
<b>TA-</b>	Negativ-Auswahl über den Anfang des Vergleichstextes . . . . .	949
<b>TE+</b>	Positiv-Auswahl über das Ende des Vergleichstextes . . . . .	948
<b>TE-</b>	Negativ-Auswahl über das Ende des Vergleichstextes . . . . .	949
<b>TTI</b>	Textteile tauschen: Index für ATT, ETT, (TT und )TT . . . . .	956
<b>TTM</b>	Textteile tauschen: Paare von Textteilen . . . . .	957
<b>TTR</b>	Textteile tauschen: Trennzeichen . . . . .	956
<b>TTT</b>	Textteile tauschen: Paare von Textteilen . . . . .	956
<b>TUT</b>	Textteile tauschen: Paare von Textteilen . . . . .	957
<b>VGL</b>	Vergleich: Vergleichstext mit Merk-Vergleichstext . . . . .	952
<b>VI</b>	Vergleich: Index für AV, EV, (V und )V . . . . .	946

<b>VSP</b>	Sprung in Abhängigkeit einer Variablen . . . . .	945
<b>VTB</b>	Vergleich: Zeichen-Tabelle . . . . .	953
<b>VTV</b>	Vergleichstext: Vorbelegung . . . . .	937
<b>VTZ</b>	Vergleichstext: Zusatzangaben . . . . .	972
<b>WS+</b>	Positiv-Abfrage von Wahlschaltern . . . . .	944
<b>WS-</b>	Negativ-Abfrage von Wahlschaltern . . . . .	944
<b>WSG</b>	Wahlschalter-Grundstellung . . . . .	936
<b>WSH</b>	Wahlschalter holen . . . . .	936
<b>WSL</b>	Wahlschalter löschen . . . . .	953
<b>WSR</b>	Wahlschalter retten . . . . .	976
<b>WSS</b>	Wahlschalter setzen . . . . .	953
<b>WSU</b>	Wahlschalter umsetzen . . . . .	953
<b>X</b>	Ersetzen von Zeichenfolgen bei der Eingabe . . . . .	943
<b>XL</b>	Ersetzen von Zeichenfolgen vor dem Lesen von Zahlen . . . . .	958
<b>XMT</b>	Ersetzen von Zeichenfolgen für den Merktext . . . . .	973
<b>XUM</b>	Ersetzen von Zeichenfolgen beim Umdrehen . . . . .	957
<b>XV</b>	Ersetzen von Zeichenfolgen vor dem Vergleichen . . . . .	946
<b>XX</b>	Ersetzen von Zeichenfolgen . . . . .	962
<b>XXB</b>	Ersetzen mit Bedingungen . . . . .	962
<b>XXI</b>	Ersetzen: Index für <b>XXX</b> , <b>EXX</b> , ( <b>XX</b> und ) <b>XX</b> . . . . .	962
<b>XXX</b>	Ersetzen von Zeichenfolgen bei der Ausgabe . . . . .	970
<b>Z</b>	Ausgeben eines Anfangstextes . . . . .	939
<b>ZA</b>	Zeilenanfangs-Kennung . . . . .	968
<b>ZAB</b>	Zeilenanfangs-Kennung bei Bedarf . . . . .	969
<b>ZB+</b>	Positiv-Auswahl über Zeichenfolgen im Vergleichstext . . . . .	948
<b>ZB-</b>	Negativ-Auswahl über Zeichenfolgen im Vergleichstext . . . . .	948
<b>ZD</b>	<b>ZIEL</b> -Datei bestimmen . . . . .	968
<b>ZDN</b>	Ausgaben eines Endetextes: Dezimalst. nach dem Komma . . . . .	975
<b>ZE</b>	Zeilenende-Kennung . . . . .	968
<b>ZEB</b>	Zeilenende-Kennung bei Bedarf . . . . .	969
<b>ZF+</b>	Positiv-Auswahl über Zeichenfolgen im Vergleichstext . . . . .	948
<b>ZF-</b>	Negativ-Auswahl über Zeichenfolgen im Vergleichstext . . . . .	949
<b>ZF</b>	Reihenfolge/Häufigkeit von Zeichenfolgen prüfen . . . . .	951
<b>ZFH</b>	Angabe, wie oft Zf. höchstens vorkommen dürfen . . . . .	952
<b>ZFM</b>	Angabe, wie oft Zf. mindestens vorkommen müssen . . . . .	951
<b>ZFP</b>	Vergleich: Auf bestimmte Zeichenfolgen prüfen . . . . .	947
<b>ZFS</b>	Zeichenfolgen im Arbeitstext suchen . . . . .	945
<b>ZFZ</b>	Zeichenfolgen im Vergleichstext zählen . . . . .	946
<b>ZIV</b>	Ausgaben eines Endetextes: I-Variablen . . . . .	975
<b>ZSP</b>	Sprung in Abhängigkeit von den gezählten Zeichenfolgen . . . . .	946
<b>ZZZ</b>	Ausgeben eines Endetextes . . . . .	975

## Prüfziffern

Die Prüfziffer wird auf der Basis des Moduls 11 mit der Gewichtung 10 bis 2 berechnet; d. h. die letzte Ziffer (Einer) der Zahl wird mit 2 multipliziert («gewichtet»), die zweitletzte (Zehner) wird mit 3 multipliziert usw. Die Produkte, die sich dabei ergeben, werden addiert. Die Prüfziffer vom Typ 1 ergibt sich nun aus der Differenz von der der Summe folgenden (nächstgrößeren) durch 11 teilbaren Zahl und der Summe; die vom Typ 2 ergibt sich aus der Differenz von der Summe und der der Summe vorangehenden (nächstkleineren) durch 11 teilbaren Zahl. Ist die Summe der Produkte selbst durch 11 teilbar, ist die Prüfziffer 0. Hat die Prüfziffer den Wert 10, wird dafür X (römische Ziffer) als Prüfziffer verwendet.

Beispiel:

Zahl:						1	9	8	5
Gewicht:	10	9	8	7	6	5	4	3	2
Produkt:	0	+ 0	+ 0	+ 0	+ 0	+ 5	+ 36	+ 24	+ 10 = 75

Die Prüfziffer vom Typ 1 der Zahl 1985 ist also 2 ( $77-75$ ), die Prüfziffer vom Typ 2 dieser Zahl ist 9 ( $75-66$ ).

## Vergleichsalgorithmus

Wird beim Vergleich je eines Zeichens aus Vergleichstext und Merk-Vergleichstext ein Unterschied gefunden, so wird geprüft, ob damit die vorgegebene Fehlergrenze überschritten wurde. Wenn nicht, wird wie folgt weitervergleichen:

Die zu vergleichenden Zeichenfolgen seien die Zeichenpositionen ...ABC... und ...123... . A und 1 seien die Positionen der ersten Zeichen, die sich unterscheiden. Die Reihenfolge der Vergleiche ist dann:

A..1, B..2, A..2, B..1, C..3, B..3, C..2, A..3, C..1

Sobald bei diesen Vergleichen zwei gleiche Zeichen gefunden werden, wird solange paarweise weitervergleichen, bis wieder zwei unterschiedliche Zeichen gefunden werden. Diese werden dann als Ausgangsbasis (A und 1) genommen und das gleiche Verfahren wiederholt, falls die Fehlergrenze noch nicht überschritten ist. Die Zahl der bei diesen Einzelvergleichen gefundenen Unterschiede bestimmt sich jeweils aus der größten Zahl (max. 2) von Zeichen, die bei diesen Einzelvergleichen in einer der beiden Zeichenfolgen übergangen werden müssen, bis eine Übereinstimmung gefunden wird.

Beispiel:

1. Zeichenfolge . . . abcdg . . .
2. Zeichenfolge . . . bcde . . .

Die Zeichen a und b sind verschieden, entsprechen also den Zeichenpositionen A..1 im oben beschriebenen Algorithmus. Dann wird b mit c (B..2), a mit c (A..2) und b mit b (B..1) verglichen und dabei wieder eine Übereinstimmung gefunden. In der 1. Zeichenfolge musste 1 Zeichen (a), in der 2. Zeichenfolge kein Zeichen übergangen werden. Es wird also 1 Unterschied angerechnet. Da dieser Algorithmus u. U. verschiedene Ergebnisse liefert, je nachdem welche Zeichenfolge als die erste behandelt wird (also den Positionen ABC entspricht), wird das ganze Verfahren mit den vertauschten Zeichenfolgen wiederholt, falls mehr als die vorgegebenen Anzahl Unterschiede festgestellt wurden. Ist nach dem 9. Einzelvergleich (C..1) keine Übereinstimmung von 2 Zeichen gefunden worden, wird der Vergleich beendet und die beiden Zeichenfolgen als nicht übereinstimmend gewertet.

## Rechenanweisungen

Die einzelnen Anweisungen müssen durch Strichpunkt getrennt werden. Leerzeichen sind bedeutungslos und können an jeder beliebigen Stelle zur besseren Lesbarkeit der Anweisungen eingefügt werden. Ebenso kann eine Anweisung an jeder beliebigen Stelle unterbrochen und in der nächsten Zeile (mit der gleichen Parameterkennung) fortgesetzt werden.

Nachfolgend werden zuerst einige Begriffe definiert, die anschließend bei der Definition der einzelnen Anweisungen benutzt werden.

### Variablen

Eine Variable ist ein mit einem Namen versehener Speicherplatz, der einen ganzzahligen Zahlenwert enthält. Jede dieser Variablen hat einen eindeutigen Namen, über den auf den Inhalt zugegriffen werden kann. Dies geschieht in der Weise, dass der Name der Variablen an die Stelle geschrieben wird, an der der Zahlenwert definiert (siehe »Wertzuweisung«) bzw. benötigt (siehe »Arithmetischer Ausdruck«) wird. Die Namen der Variablen sind fest vorgegeben und können nicht frei gewählt werden. Beim Programmstart wird in alle Variablen der Wert Null abgespeichert. Der größte Zahlenwert, der in einer Variablen abgespeichert werden kann, ist vom jeweiligen Rechner abhängig; in den meisten Fällen ist dies innerhalb von TUSTEP der Wert 2 000 000 000.

#### – H-Variablen

Es sind 10 H-Variablen (Hilfsvariablen) mit den Namen H0, H1 bis H9 definiert.

#### – I-Variablen

Die I-Variablen (100 Speicherplätze) können auf zwei Arten angesprochen werden: entweder als einfache Variable mit den Namen I0, I1 bis I99, oder als indizierte Variable in der Form I(Index). Dabei darf als Index ein beliebiger arithmetischer Ausdruck (s. u.) stehen. Der Wert des arithmetischen Ausdrucks muss jedoch zwischen 0 und 99 (je einschließlich) liegen.

Diese Variablen unterscheiden sich dadurch von den H- und B-Variablen, dass in ihnen Zahlenwerte nicht nur durch eine Wertzuweisung (s. u.) abgespeichert werden können, sondern auch durch Einlesen von Zahlen, die in den zu verarbeitenden Daten stehen (vgl. »Lesen von Zahlenwerten« Seite 958); außerdem können die in diesen Variablen gespeicherten Zahlenwerte nicht nur in arithmetischen Ausdrücken (s. u.) benutzt/abgerufen werden, sondern auch in die zu verarbeitenden Daten eingesetzt werden (vgl. »Einsetzen von Zahlenwerten« Seite 959).

#### – B-Variablen

Diese Variablen können nur als indizierte Variablen in der Form B (Index) angesprochen werden. Dabei darf als Index ein beliebiger arithmetischer Ausdruck (s. u.) stehen. Der Wert des arithmetischen Ausdrucks muss jedoch zwischen 0 und 99 (je einschließlich) liegen.

## – S-Variablen

Bei den S-Variablen (Sondervariablen) hat jede Variable eine besondere Bedeutung:

- S0 Der Inhalt dieser Variablen gibt die aktuelle Länge des Arbeitstextes an. Wird der Wert von S0 um n verkleinert, so werden die letzten n Zeichen des Arbeitstextes eliminiert; wird der Wert von S0 um n vergrößert, so werden n Leerzeichen am Ende des Arbeitstextes ergänzt. Die Zeichen werden sofort nach Beendigung der Rechenanweisungen, in denen der Inhalt dieser Variablen verändert wird, eliminiert bzw. ergänzt.
- S1 In diese Variable wird jeweils bei der Abarbeitung des Parameters DPB (siehe Seite 958) ein Zahlenwert abgespeichert. Ihr Inhalt gibt die Anzahl der Druckpositionen an, die der Arbeitstext beim Ausdrucken benötigen würde. Eine Änderung des Inhalts dieser Variablen ist ohne Bedeutung.
- S2 Stand der laufenden Nummer (nicht der Ausgabennummer; siehe »Einsetzen einer laufenden Nummer« Seite 965). Diese Variable kann beliebig verändert werden.
- S3 Der Inhalt dieser Variablen gibt die Länge des letzten Vergleichstextes an. Der Vergleichstext wird jeweils bei der Ausführung von Programmteil 1 definiert, wenn dieser Programmteil nicht schon über einen der Parameter SPW (bzw. die entsprechende Sprungadresse des Parameters SPR), VSP oder KSP verlassen wird. Eine Änderung des Inhalts dieser Variablen ist ohne Bedeutung.
- S4 In diese Variable wird jeweils bei der Abarbeitung des Parameters ZF (siehe Seite 951) ein Zahlenwert abgespeichert. Ihr Inhalt gibt an, als wievielte die Zeichenfolge, die im Vergleichstext als erste gefunden wurde, zum Parameter angegeben ist. Falls jedoch die Reihenfolge der im Vergleichstext gefundenen Zeichenfolgen nicht mit der Reihenfolge, in der sie zum Parameter angegeben sind, übereinstimmte oder die mit den Parametern ZFM und ZFH geforderten Bedingungen nicht erfüllt wurden, gibt der Inhalt der Variablen S4 an, als wievielte die Zeichenfolge, die die Reihenfolge oder die Bedingung verletzte, zum Parameter ZF angegeben ist. Eine Änderung des Inhalts dieser Variablen ist ohne Bedeutung.
- S5 In diese Variable wird jeweils bei der Abarbeitung des Parameters ZFZ (siehe Seite 946) ein Zahlenwert abgespeichert. Ihr Inhalt gibt die Anzahl der im Vergleichstext gefundenen Zeichenfolgen an. Eine Änderung des Inhalts dieser Variablen ist ohne Bedeutung.
- S6 In diese Variable wird jeweils bei der Abarbeitung des Parameters ZFZ (siehe Seite 946) ein Zahlenwert abgespeichert. Ihr Inhalt gibt an, als wievielte die Zeichenfolge, die im Vergleichstext als erste gefunden wurde, zum Parameter angegeben war. Eine Änderung des Inhalts dieser Variablen ist ohne Bedeutung.
- S7 Der Inhalt dieser Variablen wird bei jeder Abarbeitung der Parameter ERS und EZF (siehe Seite 962) ausgewertet.
- S8 In diese Variable wird jeweils bei der Abarbeitung des Parameters ZFS (siehe



Seite 945) ein Zahlenwert abgespeichert. Ihr Inhalt gibt an, als wievielte die Zeichenfolge, die im Arbeitstext als erste gefunden wurde, zum Parameter angegeben war. Der Inhalt dieser Variablen bei jeder Abarbeitung des Parameters `VSP` (siehe Seite 945) ausgewertet.

- S9 Der Zahlenwert, der in dieser Variablen gespeichert ist, kann in eine Meldung eingesetzt werden, die mit dem Parameter `MLD` (siehe Seite 967) ausgegeben wird.
- S10 Durch den Inhalt dieser Variablen kann definiert werden, welche Satznummer (Seiten-Zeilen-Unterscheidungsnummer) in den Variablen `S11` und `S12` vor Abarbeitung des Parameters `RR` abgespeichert werden soll. Enthält `S10` den Wert Null, so enthalten `S11` und `S12` die Satznummer der aktuellen Texteinheit. Enthält `S10` den Wert `n`, so enthalten `S11` und `S12` die Satznummer des zuletzt auf die `n`-te `ZIEL`-Datei ausgegebenen Satzes. Wurde jedoch für diese Datei schon die Satznummer für den nächsten Satz eingestellt (siehe `S13`), so enthält `S11` und `S12` diese eingestellte Satznummer. Ist der Zahlenwert in `S10` kleiner als Null oder größer als die Anzahl der im Programmaufruf angegebenen `ZIEL`-Dateien, wird das Programm mit einer entsprechenden Fehlermeldung abgebrochen.
- S11 Diese Variable enthält den Teil der Satznummer, der die Seitennummer angibt (siehe auch `S10` und `S13`).
- Was eine Satznummer ist, und wie sie sich zusammensetzt, ist in der Beschreibung »TUSTEP-Grundlagen« im Kapitel »Dateistruktur« beschrieben.
- S12 Diese Variable enthält den Teil der Satznummer, der die Zeilen- und Unterscheidungsnummer angibt (siehe auch `S10` und `S13`).
- Was eine Satznummer ist, und wie sie sich zusammensetzt, ist in der Beschreibung »TUSTEP-Grundlagen« im Kapitel »Dateistruktur« beschrieben.
- S13 Durch den Inhalt dieser Variablen kann definiert werden, welche Satznummer (Seiten-Zeilen-Unterscheidungsnummer) nach Abarbeitung des Parameters `RR` durch den Inhalt der Variablen `S11` und `S12` ersetzt werden soll. Enthält `S13` den Wert Null, so wird die Satznummer der aktuellen Texteinheit ersetzt. Enthält `S13` den Wert `n`, so wird durch `S11` und `S12` die Satznummer für den nächsten Satz, der auf die `n`-te `ZIEL`-Datei ausgegeben wird, eingestellt. Ist der Zahlenwert in `S13` kleiner als Null oder größer als die Anzahl der im Programmaufruf angegebenen `ZIEL`-Dateien, wird das Programm mit einer entsprechenden Fehlermeldung abgebrochen.
- S14 Der Inhalt dieser Variablen gibt an, wieviele Zeichen am Anfang des Arbeitstextes eliminiert werden sollen. Die Zeichen werden sofort nach Beendigung der Rechenanweisungen, in denen dieser Variablen ein Wert zugewiesen wird, eliminiert; danach wird die Variable `S14` wieder automatisch auf Null gesetzt. Ist der Inhalt von `S14` größer als der Inhalt von `S0` (Länge des Arbeitstextes), wird der gesamte Arbeitstext gelöscht.
- S17 Der Inhalt dieser Variablen gibt an, welcher Merktext im Programmteil 7 definiert wird. Die Merktex te sind von 0 bis 9 durchnummeriert.

- S18 Der Inhalt dieser Variablen gibt an, welcher Merktext im Programmteil 8 mit dem Arbeitstext ausgetauscht wird. Die Merktex-te sind von 0 bis 9 durchnummeriert.
- S19 Der Inhalt dieser Variablen gibt an, welcher Merktext im Programmteil 9 geholt wird. Die Merktex-te sind von 0 bis 9 durchnummeriert.

#### – Wahlschalter

Als Wahlschalter sind 16 Variablen mit den Namen `WS1`, `WS2` bis `WS16` definiert, die nur die beiden Zahlenwerte 0 und 1 enthalten können. Ein Wahlschalter ist gelöscht, wenn er den Zahlenwert 0 enthält, und ist gesetzt, wenn er den Zahlenwert 1 enthält. In einer Wertzuweisung (s. u.), in der links vom Gleichheitszeichen ein Wahlschalter angegeben ist, darf rechts vom Gleichheitszeichen kein beliebiger arithmetischer Ausdruck (s. u.) stehen, sondern nur eine der Zahlen 0 und 1. Ein Wahlschalter kann nicht nur durch eine Wertzuweisung verändert werden, sondern auch durch die Parameter `WSL`, `WSS` und `WSU` (siehe Seite 953); außerdem kann ein Wahlschalter nicht nur in arithmetischen Ausdrücken benutzt/abgerufen werden, sondern auch durch die Parameter `WS+` und `WS-` (siehe Seite 944).

## Funktionen

Zur Unterstützung von Berechnungen stehen Funktionen zur Verfügung. Eine Funktion wird mit ihrem Namen und einem oder mehreren Argumenten aufgerufen. Die Argumente werden, durch Komma getrennt, in Klammern hinter dem Funktionsnamen angegeben. Sie können beliebige arithmetische Ausdrücke (s. u.) sein. Durch den Aufruf einer Funktion mit diesen Argumenten wird nach der Funktionsvorschrift ein Zahlenwert, der Funktionswert, errechnet.

(In den Beispielen stehen für die einzelnen Argumente der Einfachheit wegen Zahlen. In der Praxis stehen dafür meistens Variablen.)

#### – Berechnung des Absolutbetrags: `IABS (arg)`

Als Funktionswert erhält man den Absolutbetrag von `arg`.

Beispiel: Der Funktionswert von `IABS (-4)` ist 4.

#### – Berechnung des Minimums: `MIN (arg1, arg2)`

Als Funktionswert erhält man den kleineren der beiden Zahlenwerte `arg1` und `arg2`.

Beispiel: Der Funktionswert von `MIN (-5, +3)` ist -5.

#### – Berechnung des Maximums: `MAX (arg1, arg2)`

Als Funktionswert erhält man den größeren der beiden Zahlenwerte `arg1` und `arg2`.

Beispiel: Der Funktionswert von `MAX (-5, +3)` ist +3.

– Berechnung des Divisionsrestes: `MOD (arg1, arg2)`

Als Funktionswert erhält man den Rest, der sich bei der Division von `arg1` durch `arg2` ergibt.

Beispiel: Der Funktionswert von `MOD (234, 10)` ist 4.

– Intervall-Funktion: `IV (arg, arg1, arg2, arg3, ..., argn)`

Mit dieser Funktion kann festgestellt werden, in welches Intervall der Zahlenwert `arg` fällt. Man erhält als Funktionswert Null, falls `arg` kleiner als `arg1` ist, den Wert 1, falls `arg` größer oder gleich `arg1` aber kleiner als `arg2` ist, den Wert 2, falls `arg` größer oder gleich `arg2` aber kleiner als `arg3` ist, ..., den Wert `n`, falls `arg` größer oder gleich `argn` ist. Die Anzahl der Argumente ist bei dieser Funktion beliebig, jedoch muss `arg1` kleiner als `arg2` sein, `arg2` kleiner als `arg3` sein usw.

Beispiel: Der Funktionswert von `IV (16, 1, 10, 100, 1000)` ist 2.

– Berechnung einer Prüfziffer: `IP (arg)`

Als Funktionswert erhält man eine Zahl zwischen 0 und 10 (je einschließlich), die nach folgender Regel berechnet wird: Die letzte Ziffer des Zahlenwertes von `arg` wird mit 2 multipliziert, die zweitletzte mit 3, die drittletzte mit 4 usw. Die Produkte, die sich dabei ergeben, werden addiert. Lässt sich diese Summe ohne Rest durch 11 teilen, ist der Funktionswert Null. Andernfalls entspricht er der Differenz zwischen der nächsten durch 11 teilbaren Zahl und dieser Summe. Ist der Zahlenwert von `arg` kleiner als 1 oder größer als 999999999, so erhält man als Funktionswert -1, da in diesem Fall die Prüfziffer nicht definiert ist.

– Datumsfunktion: `ID (tag, monat, jahr, nummer, modus)`

Mit dieser Funktion sind verschiedene Be- und Umrechnungen von Kalenderdaten möglich. Sie wird über das Argument `modus` gesteuert.

Für die Berechnung wird jeweils der Gregorianische Kalender zugrunde gelegt, für ein Datum vor dem 15. Oktober 1582 jedoch der Julianische Kalender. Soll in jedem Fall der Julianische Kalender gelten, so ist als Argument `modus` der entsprechende negative Wert anzugeben.

Um das Rechnen mit Kalenderdaten zu erleichtern, werden die Tage durchnummeriert. Damit hat jeder Tag eine eindeutige Nummer, die Tagesnummer.

`modus=0`: Aktuelles Datum

Eingabe: keine

Ergebnis: Aktuelles Tagesdatum in den Argumenten `tag`, `monat`, `jahr` und aktuelle Tagesnummer im Argument `nummer`

`modus=1`: Berechnung der Tagesnummer aus dem Tagesdatum

Eingabe: Tagesdatum in den Argumenten `tag`, `monat`, `jahr`

Ergebnis: Tagesnummer im Argument `nummer`

`modus=2`: Berechnung des Tagesdatums aus der Tagesnummer

Eingabe: Tagesnummer im Argument `nummer`

Ergebnis: Tagesdatum in den Argumenten `tag`, `monat`, `jahr`

- `modus=3`: Berechnung des Osterdatums  
 Eingabe: Jahreszahl im Argument `jahr`  
 Ergebnis: Osterdatum in den Argumenten `tag`, `monat`, `jahr` und entsprechende Tagesnummer im Argument `nummer`
- `modus=4`: Berechnung des Zeitraumes zwischen zwei Kalenderdaten  
 Eingabe: Tagesdatum des ersten Datums in den Argumenten `tag`, `monat`, `jahr` und Tagesnummer des zweiten (späteren) Datums im Argument `nummer`  
 Ergebnis: Der zwischen den beiden Daten liegende Zeitraum in Jahren, Monaten und Tagen (in den Argumenten `jahr`, `monat` und `tag`) sowie in Tagen (im Argument `nummer`)  
 Achtung: Wird für `modus -4` angegeben, so wird für beide Kalenderdaten der Julianische Kalender zugrunde gelegt.
- `modus=5`: Berechnung des Zeitraumes zwischen zwei Kalenderdaten  
 Eingabe: Tagesdatum des ersten Datums in den Argumenten `tag`, `monat`, `jahr` und Tagesnummer des zweiten (späteren) Datums im Argument `nummer`  
 Ergebnis: Der zwischen den beiden Daten liegende Zeitraum in Jahren, Monaten und Tagen (in den Argumenten `jahr`, `monat` und `tag`) sowie in Tagen (im Argument `nummer`)  
 Achtung: Wird für `modus 5` angegeben, so wird für das erste Datum der oben angegebenen Regel entsprechende Kalender und für das zweite Datum der Julianische Kalender zugrunde gelegt; wird für `modus -5` angegeben, so wird für das erste Datum der Julianische Kalender und für das zweite Datum der oben angegebenen Regel entsprechende Kalender zugrunde gelegt.

Als Funktionswert erhält man, unabhängig vom Wert des Arguments `modus`, den Wochentag (1=Montag, 2=Dienstag, 3=Mittwoch, 4=Donnerstag, 5=Freitag, 6=Samstag, 7=Sonntag) des Tages, dessen Datum in den Argumenten `tag`, `monat` und `jahr` steht. Bei unzulässiger Eingabe (wenn z. B. `modus` einen nicht vorgesehenen Wert hat oder ein ungültiges Datum, z. B. 29. Februar eines Nicht-Schaltjahres, angegeben wird) ist der Funktionswert Null.

Beispiel: Berechnen des Pfingstdatums von 2008

```
H0 = ID (H1, H2, 2008, H4, 3);
H0 = ID (H1, H2, H3, H4+49, 2);
```

Es wird zuerst das Osterdatum für das Jahr 2008 errechnet. Dabei ist es belanglos, welche Zahlenwerte `H1`, `H2` und `H4` vor dem Aufruf der Datumsfunktion enthalten. Da zu `modus 3` angegeben ist, wird nur das Argument `jahr` ausgewertet. Wochentag, Tag und Monat von Ostern werden in `H0`, `H1` und `H2` abgespeichert, werden aber nicht weiter verwendet. Die Tagesnummer von Ostern wird in `H4` abgespeichert. Da zwischen Ostern und Pfingsten genau 7 Wochen (= 49 Tage) liegen, erhält man die Tagesnummer von Pfingsten, indem man zur Tagesnummer von Ostern 49 addiert. Diese Tagesnummer wird dann in das entsprechende Tagesdatum umgerechnet. Dabei ist es belanglos, welche Zahlenwerte `H1`, `H2` und `H3` vor

dem Aufruf der Datumsfunktion enthalten. Da zu `modus 2` angegeben ist, wird nur das Argument `nummer` ausgewertet. In `H0` wird der Wochentag (7 für Sonntag), in `H1`, `H2` und `H3` werden Tag, Monat und Jahr des Pfingstdatums abgespeichert.

## Arithmetischer Ausdruck

Ein arithmetischer Ausdruck ist eine Rechenvorschrift, durch die ein numerischer Wert bestimmt wird. Er besteht aus Operanden, arithmetischen Operatoren und Klammerpaaren.

Ein Operand kann eine (ganze) Zahl, eine (einfache oder indizierte) Variable oder ein Funktionsaufruf sein. Im einfachsten Fall besteht ein arithmetischer Ausdruck nur aus einer dieser drei Angaben.

Es stehen folgende arithmetischen Operatoren zur Verfügung:

- + für die Addition
- \* für die Multiplikation
- für die Subtraktion
- / für die Division

Bei der Auswertung des arithmetischen Ausdrucks werden Multiplikation und Division vor Addition und Subtraktion ausgeführt. Folgen mehrere Multiplikationen und/oder Divisionen aufeinander, so werden diese von links nach rechts ausgewertet. Das gleiche gilt für aufeinander folgende Additionen und/oder Subtraktionen. Soll von dieser Regelung abgewichen werden, kann die Reihenfolge durch Setzen von Klammern festgelegt werden.

Bei der Division ist zu beachten, dass grundsätzlich nur mit ganzen Zahlen gerechnet wird. Der Divisionsrest geht verloren; eine Rundung findet nicht statt. Dies hat zur Folge, dass z. B.  $3/2$  den Wert 1 (nicht 1,5) und  $3/2*4$  den Wert 4 (nicht 6) ergibt.

## Vergleichsbedingung

Bei der Vergleichsbedingung werden zwei numerische Werte miteinander verglichen. Sie besteht aus zwei arithmetischen Ausdrücken, die durch einen Vergleichsoperator miteinander verbunden sind:

arithm. Ausdruck Vergleichsoperator arithm. Ausdruck

Es stehen folgende Vergleichsoperatoren zur Verfügung:

- .EQ. für »equal«
- .NE. für »not equal«
- .GT. für »greater than«
- .LT. für »less than«
- .GE. für »greater or equal«
- .LE. für »less or equal«

Eine Vergleichsbedingung ist entweder erfüllt, dann hat sie den logischen Wert WAHR, oder sie ist nicht erfüllt, dann hat sie den logischen Wert FALSCH.

Beispiele:

$I1.EQ.0$	ist erfüllt, falls I1 den Zahlenwert 0 enthält, sonst nicht erfüllt.
$MOD(I1,10).LT.5$	ist erfüllt, falls bei der Division von I1 durch 10 sich ein Rest ergibt, der kleiner als 5 ist, sonst nicht erfüllt.
$I1+I2.EQ.4*I3$	ist erfüllt, falls sich bei der Addition von I1 und I2 das Vierfache von I3 ergibt, sonst nicht erfüllt.

## Logischer Ausdruck

Ein logischer Ausdruck besteht aus Vergleichsbedingungen, die mit logischen Operatoren miteinander verbunden sein können. Im einfachsten Fall kann ein logischer Ausdruck auch nur aus einer Vergleichsbedingung bestehen.

Es stehen folgende logische Operatoren zur Verfügung:

- .AND. für logisches UND
- .OR. für logisches ODER

Die Auswertung eines logischen Ausdrucks erfolgt analog zur Auswertung eines arithmetischen Ausdrucks. Als Ergebnis erhält man entweder den logischen Wert WAHR oder den logischen Wert FALSCH. Bei der Auswertung wird das logische UND vor dem logischen ODER ausgeführt. Soll von dieser Regelung abgewichen werden, kann die Reihenfolge durch Setzen von Klammern festgelegt werden.

Beispiele:

$I1.EQ.I2$	hat den logischen Wert WAHR, wenn I1 den gleichen Zahlenwert enthält wie I2, sonst den logischen Wert FALSCH.
$I1.GE.1 .AND. I1.LE.8$	hat den logischen Wert WAHR, wenn I1 einen Zahlenwert zwischen 1 und 8 (je einschließlich) enthält, sonst den logischen Wert FALSCH.
$I1.EQ.0 .AND. (I2.EQ.1 .OR. I2.EQ.2)$	hat den logischen Wert WAHR, wenn I1 den Zahlenwert Null enthält und I2 einen der Zahlenwerte 1 und 2 enthält, sonst den logischen Wert FALSCH.

## Wertzuweisung

Eine Wertzuweisung hat die Form:

Variable = arithmetischer Ausdruck

Sie bewirkt die Auswertung des arithmetischen Ausdrucks. Das Ergebnis wird in die links vom Gleichheitszeichen angegebene (einfache oder indizierte) Variable abgespeichert. Das Gleichheitszeichen hat bei dieser Anweisung die Funktion eines Zu-

weisungsoperators und unterscheidet sich somit von seiner Bedeutung in der Mathematik.

Beispiele:  $H1 = 0; I1 = I1 + 1; I(I1) = \text{MAX}(H2, H3);$   
 $I1 = I1 * (H1 + H2); IO = \text{MOD}(S12, 1000);$

## Sprunganweisung

Eine Sprunganweisung hat die Form:

GOTO arithmetischer Ausdruck

Sie bewirkt die Auswertung des arithmetischen Ausdrucks. Das Ergebnis wird als Sprungziel interpretiert, der Sprung an die entsprechende Stelle im Programm ausgeführt. Dabei bezeichnet die Einerstelle des Ergebnisses den Programmteil und die Hunderter- und Zehnerstelle die Nummer des Durchgangs.

Beispiele:  $\text{GOTO } 0; \text{GOTO } H1 * 10 + 2; \text{GOTO } B(I1)$

Außerdem gibt es zwei spezielle Sprunganweisungen:

GOON

bewirkt ein Überspringen sämtlicher nachfolgender Rechenanweisungen (des gleichen Durchgangs).

EXIT

darf nur zwischen den Anweisungen LOOP und ENDDLOOP stehen (vgl. Schleifenanweisung) und bewirkt ein Überspringen der nachfolgenden Anweisungen bis hinter das nächste ENDDLOOP.

## Bedingungsanweisung

Die Bedingungsanweisung ermöglicht es, dass eine oder mehrere aufeinander folgende Anweisungen nur unter bestimmten Bedingungen ausgeführt werden. Die Bedingungen werden durch einen logischen Ausdruck angegeben. Es gibt drei Formen der Bedingungsanweisung:

1.) Wenn eine einzige Anweisung (dies darf in diesem Fall nur eine Wertzuweisung oder eine Sprunganweisung sein) nur unter den (im logischen Ausdruck) angegebenen Bedingungen ausgeführt werden soll, kann folgende Form gewählt werden:

IF (logischer Ausdruck) Anweisung

Die nach den Klammern angegebene Anweisung wird nur dann ausgeführt, wenn der in Klammern stehende logische Ausdruck den Wert WAHR hat. Andernfalls wird diese Anweisung übergangen.

Beispiele:  $\text{IF } (I1.EQ.0) \text{ } I2 = I2 + 1;$   
 $\text{IF } (\text{MOD}(S12, 1000).EQ.0) \text{ } \text{GOTO } 3;$

2.) Wird die folgende Form der bedingten Anweisung gewählt, so können eine oder mehrere (beliebige) Anweisungen in Abhängigkeit von den (im logischen Ausdruck) angegebenen Bedingungen ausgeführt werden:

```
IF (logischer Ausdruck) THEN;
  Anweisungen;
ENDIF
```

Die zwischen THEN und ENDIF stehenden Anweisungen werden nur dann ausgeführt, wenn der in Klammern stehende logische Ausdruck den Wert WAHR hat. Andernfalls werden diese Anweisungen übergangen.

```
Beispiele: IF (I1.EQ.60) THEN; S11=S11+1; S12=1000; ENDIF;
           IF (WS1.NE.0) THEN; I2=I2+1; GOTO 0; ENDIF;
```

3.) Die dritte Form der bedingten Anweisung ist eine Erweiterung der zweiten. Mit ihr können zwei Anweisungsfolgen angegeben werden, von denen jeweils eine ausgeführt und die andere übergangen wird:

```
IF (logischer Ausdruck) THEN;
  Anweisungen;
ELSE;
  Anweisungen;
ENDIF
```

Hat der in Klammern stehende logische Ausdruck den Wert WAHR, so werden die Anweisungen zwischen THEN und ELSE ausgeführt, die zwischen ELSE und ENDIF stehenden übergangen. Hat der logische Ausdruck den Wert FALSCH, so werden die Anweisungen zwischen THEN und ELSE übergangen, die zwischen ELSE und ENDIF stehenden ausgeführt.

## Schleifenanweisung

Die Schleifenanweisung ermöglicht es, eine Folge von Anweisungen beliebig oft zu durchlaufen. Diese Folge von Anweisungen wird durch die Anweisung

```
LOOP
```

eingeleitet und durch die Anweisung

```
ENDLOOP
```

abgeschlossen.

Es gibt drei Möglichkeiten, diese Anweisungsfolge zu verlassen:

- Mit der Anweisung EXIT kann hinter das nachfolgende ENDLOOP gesprungen werden.
- Mit der Anweisung GOTO kann zu einem bestimmten Programmteil gesprungen werden.
- Mit der Anweisung GOON kann die Ausführung der Rechenanweisungen beendet werden, d. h. alle nachfolgenden (auch die hinter ENDLOOP stehenden) werden übergangen.



Beispiel: Es sollen die Variablen  $I(1)$  bis  $I(32)$  auf Null gesetzt werden. Dabei wird die Variable  $H0$  als Zähler und als Index benutzt.

```
H0 = 1;  
LOOP;  
  I(H0) = 0;  
  IF (H0.EQ.32) EXIT;  
  H0 = H0 + 1;  
ENDLOOP;
```

## Logischer Programmaufbau

Auf der folgenden Seite ist der logische Programmaufbau für drei Durchgänge dargestellt. Entsprechend sind auch die Voreinstellungen für die Parameter *SPR* bzw. *SP0*, *SPW*, *SPN* usw. für drei Durchgänge angegeben. Bei mehr Durchgängen ergibt sich die Voreinstellung für den jeweils nächsten Durchgang durch Aufaddieren von 10. Eine Ausnahme bildet der Parameter *SPN*, bei dem für alle Durchgänge 0 voreingestellt ist, und die beiden Parameter *SPW* und *SP2*. Bei *SPW* und *SP2* gilt die oben angegebene Regel für alle Durchgänge mit Ausnahme des letzten: Im letzten Durchgang ist für den Parameter *SPW* 8 und für den Parameter *SP2* 3 voreingestellt.

Abkürzungen, die auf der folgenden Seite verwendet werden:

AT	Arbeitstext
ET	Ersetzungstext
GT	Grundtext
MT	Merktext
MVT	Merk-Vergleichstext
VT	Vergleichstext

		[SPR]	[SPR]	[SPR]
0		10	20	30
Datei ⇒ GT → 10	[SP0]	GT ⇒ AT → 11	GT ⇒ AT → 21	GT ⇒ AT → 31
		11	21	31
	[SPW]	WS-ABFRAGE erf. → 21 VERGLEICH AT ⇒ VT	WS-ABFRAGE erf. → 31 VERGLEICH AT ⇒ VT	WS-ABFRAGE erf. → 8 VERGLEICH AT ⇒ VT
	[SPN]	nein → 0	nein → 0	nein → 0
	[SPJ]	ja → 12 → 12	ja → 22 → 22	ja → 32 → 32
		12	22	32
	[SP2]	VERARBEITG AT ⇒ AT → 21	VERARBEITG AT ⇒ AT → 31	VERARBEITG AT ⇒ AT → 3
3		13	23	33
AT ⇒ Datei → 0	[SP3]	AT ⇒ Datei → 10	AT ⇒ Datei → 20	AT ⇒ Datei → 30
4		14	24	34
FEHLER → 0	[SP4]	AT ⇒ GT → 11	AT ⇒ GT → 21	AT ⇒ GT → 31
5		15	25	35
VT/AT ⇒ MVT → 0	[SP5]	VT/AT ⇒ MVT → 10	VT/AT ⇒ MVT → 20	VT/AT ⇒ MVT → 30
6		16	26	36
AT ⇒ ET → 0	[SP6]	AT ⇒ ET → 10	AT ⇒ ET → 20	AT ⇒ ET → 30
7		17	27	37
AT ⇒ MT → 0	[SP7]	AT ⇒ MT → 10	AT ⇒ MT → 20	AT ⇒ MT → 30
8		18	28	38
FEHLER STOP	[SP8]	MT ⇔ AT → 11	MT ⇔ AT → 21	MT ⇔ AT → 31
9		19	29	39
STOP	[SP9]	MT ⇒ AT → 11	MT ⇒ AT → 21	MT ⇒ AT → 31



**#NUMMEREI**

---

Kommando . . . . .	999
Leistung . . . . .	1000
Beispiele . . . . .	1000
Modi . . . . .	1002
Parameter . . . . .	1003
Einstellen der Parameter-Konvention . . . . .	1003
Auswahl der Daten . . . . .	1003
Einsetzen einer laufenden Nummer . . . . .	1004
Aktualisieren der Verweisnummern . . . . .	1005
Verweise auf Seiten- und Zeilennummern . . . . .	1005
Angaben zum Protokoll . . . . .	1006
Alphabetisches Verzeichnis der Parameter . . . . .	1006

## Kommando

### #NUMMERIERE

QUELLE	= datei	Name der Datei mit den Daten, in denen Nummern und/oder Verweise aktualisiert werden sollen.
	= -STD-	Die Daten, in denen Nummern und/oder Verweise aktualisiert werden sollen, stehen in der Standard-Text-Datei.
ZIEL	= datei	Name der Datei für die Daten mit den aktualisierten Nummern und/oder Verweisen.
	= -STD-	Die Daten mit den aktualisierten Nummern und/oder Verweisen in die Standard-Text-Datei ausgeben.
MODUS	= -STD-	* Normalfall.
	= MERKE	Konkordanz von alter und neuer Nummerierung in die KONKORDANZ-Datei ausgeben; evtl. in dieser Datei enthaltene Daten werden unabhängig von der Angabe zur Spezifikation LOESCHEN gelöscht.
	= HOLE	Konkordanz von alter und neuer Nummerierung von der KONKORDANZ-Datei einlesen.
	= ERGAENZE	Konkordanz von alter und neuer Nummerierung in der KONKORDANZ-Datei ergänzen.
LOESCHEN	= -	* Daten in der ZIEL- und in der PROTOKOLL-Datei nicht löschen.
	= +	Daten in der ZIEL- und in der PROTOKOLL-Datei zuerst löschen.
PARAMETER	= datei	Name der Datei mit den Parametern.
	= *	Die Parameter folgen auf das Kommando und sind mit *EOF abgeschlossen.
KONKORDANZ	= -	* Normalfall.
	= datei	Name der Datei für die bzw. mit der Konkordanz von alter und neuer Nummerierung.
	= -STD-	Die Konkordanz von alter und neuer Nummerierung in die Standard-Daten-Datei ausgeben bzw. von der Standard-Daten-Datei einlesen.
PROTOKOLL	= -	Keine Protokoll der Konkordanz von alter und neuer Nummerierung erstellen.
	= +	Protokoll der Konkordanz von alter und neuer Nummerierung ins Ablaufprotokoll ausgeben.

---

= -STD-	* Protokoll der Konkordanz von alter und neuer Nummerierung in die Standard-Protokoll-Datei ausgeben.
= datei	Name der Datei für das Protokoll der Konkordanz von alter und neuer Nummerierung.

## Leistung

Mit diesem Programm können (laufende) Nummern und die dazugehörigen Verweise aktualisiert werden. Dazu können in einem Text an entsprechend gekennzeichneten Stellen laufende Nummern eingesetzt werden. Eine dort ggf. bereits vorhandene (alte) Nummer wird dabei ersetzt. Gleichzeitig können entsprechend gekennzeichnete Verweise, die sich auf die alten Nummern beziehen, aktualisiert werden.

Außerdem kann das Programm an entsprechend gekennzeichneten Stellen Verweise, die sich auf eine Seiten-Zeilen-Nummer beziehen, aktualisieren, nachdem sich die Seiten-Zeilen-Einteilung verändert hat.

## Beispiele

Daten in Datei alt:

```
...
2.3 ... (siehe Seite @i *4 \9) ...
...
3.4 ... @i #2 ...
...
5.6 ... @i #4 ...
...
7.8 ... (siehe Seite @i *2 \1) ...
...
```

Das Kommando

```
#NU,alt,neu,,+,*
LNR      /@i #/
VNR      /@i \*/
SK       /\
*EOF
```

aktualisiert die Seitenzahlen im Text und



liefert in der Datei neu folgendes Ergebnis:

```

...
2.3 ... (siehe Seite @i *4 \5) ...
...
3.4 ... @i #2 ...
...
5.6 ... @i #4 ...
...
7.8 ... (siehe Seite @i *2 \3) ...
...

```

Das Kommando

```

#NU,alt,neu,,+,*
LNR      /@i #/
VNR      /@i \*/
*EOF

```

numeriert die laufenden Nummern neu durch,  
aktualisiert die Verweisnummern entsprechend und  
liefert in der Datei neu folgendes Ergebnis:

```

...
2.3 ... (siehe Seite @i *2 \9) ...
...
3.4 ... @i #1 ...
...
5.6 ... @i #2 ...
...
7.8 ... (siehe Seite @i *1 \1) ...
...

```

## Modi

Die laufenden Nummern und die Verweise stehen in der Regel in der selben Datei. In diesem Fall braucht zur Spezifikation `MODUS` nichts angegeben zu werden (gleichbedeutend mit `MODUS=-STD-`).

Stehen die laufenden Nummern und die Verweise in verschiedenen Dateien, so muss zuerst die Datei mit den laufenden Nummern bearbeitet werden. Danach steht fest, in welcher Weise die Verweise aktualisiert werden müssen, d. h. welche alte Verweisnummer durch welche neue Verweisnummer ersetzt werden muss, bzw. auf welcher Seite und Zeile sich die Bezugspunkte (= laufende Nummern) befinden. Diese Information (= Konkordanz) muss für den zweiten Aufruf des Programms, bei dem die Datei mit den Verweisen bearbeitet wird, zwischengespeichert werden. Das Programm wird dazu beim ersten Aufruf durch die Angabe `MODUS=MERKE` veranlasst. Das Zwischenspeichern erfolgt auf die Datei, die zur Spezifikation `KONKORDANZ` angegeben ist.

Durch die Angabe `MODUS=HOLE` wird das Programm beim zweiten Aufruf, bei dem die Verweise aktualisiert werden, veranlasst, die zwischengespeicherte Konkordanz auszuwerten. Zur Spezifikation `KONKORDANZ` muss die Datei angegeben werden, die beim ersten Aufruf (mit `MODUS=MERKE`) zu dieser Spezifikation angegeben war.

Befinden sich die Daten mit den laufenden Nummern auf mehreren Dateien, so müssen zuerst alle diese Dateien einzeln bearbeitet werden (mit dem Parameter `LNR`, ohne den Parameter `VNR`, aber ggf. mit den Parametern `SK` und `ZK`, damit auch die Seiten-Zeilen-Nummern gemerkt werden). Dabei muss für die Bearbeitung der ersten Datei `MODUS=MERKE` und bei allen folgenden `MODUS=ERGAENZE` angegeben werden. Bei `MODUS=ERGAENZE` wird zu Anfang die Konkordanz eingelesen (wie bei `MODUS=HOLE`) und zum Schluss in der ergänzten Form wieder ausgegeben (wie bei `MODUS=MERKE`). Anschließend werden die Dateien mit den Verweisen mit `MODUS=HOLE` bearbeitet (mit dem Parameter `VNR`, ohne den Parameter `LNR`, aber ggf. mit den Parametern `SK` und `ZK`, damit die Seiten-Zeilen-Nummern eingesetzt werden). Bei diesem Vorgehen dürfen die Dateien laufende Nummern und Verweise enthalten.

## Parameter

Jeder Parameter ist einer Parameterart zugeordnet. Sie ist jeweils als römische Zahl in [ ] angegeben. Damit ist festgelegt, in welcher Form die Angaben vom Programm erwartet werden. Die einzelnen Parameterarten sind in der Beschreibung »TUSTEP-Grundlagen« im Kapitel »{}-Parameter« bzw. »<>-Parameter« definiert.

Für manche Parameter gibt es voreingestellte Werte, die angenommen werden, falls die entsprechenden Werte nicht angegeben sind. Diese Voreinstellungen sind in < > angegeben.

Außer den im folgenden beschriebenen Parametern sind auch Parameter zur Definition von Zeichen- und Stringgruppen möglich. [ V ]

### Einstellen der Parameter-Konvention

Ob die Parameter nach der »{}-Parameter-Konvention« oder nach der »<>-Parameter-Konvention« interpretiert werden, kann mit dem Kommando #PARAMETER, {} bzw. #PARAMETER, <> eingestellt werden. Darüber hinaus kann die Konvention auch mit folgendem Parameter eingestellt werden.

- PAR** Parameter-Konvention einstellen.
- { } Nachfolgende Parameter nach der »{}-Parameter-Konvention« interpretieren.
  - <> Nachfolgende Parameter nach der »<>-Parameter-Konvention« interpretieren.

Die Einstellung mit dem Parameter **PAR** hat Vorrang vor der mit dem Kommando #PARAMETER gewählten Einstellung; sie gilt jeweils nur für die nachfolgenden Parameter bis zum nächsten Parameter **PAR** bzw. bis zum Ende der Parameter dieses Programms.

### Auswahl der Daten

Soll die gesamte Datei bearbeitet werden, braucht keiner der folgenden Parameter angegeben zu werden.

- BER** Angabe eines Bereichs (»Seite.Zeile–Seite.Zeile«, falls die Datei im Textmodus nummeriert ist; Zeile–Zeile, falls die Datei im Programmmodus nummeriert ist) oder einer Anfangsstelle (»Seite.Zeile« bzw. »Zeile«), falls nicht die ganze Datei bearbeitet werden soll. [ XI ]

Soll ein Segment einer Segment-Datei verarbeitet werden, kann anstelle des Bereichs der Name des Segments angegeben werden.

Dieser Parameter ist nur zugelassen, wenn die Satznummern in der Datei alle aufsteigend sind.

**MAX** Angabe für Testzwecke, wieviele Eingabesätze maximal bearbeitet werden sollen. [ I ] <999999999>

## Einsetzen einer laufenden Nummer

**LNR** Zeichenfolgen, hinter denen die laufende Nummer eingesetzt werden soll. Steht unmittelbar hinter einer solchen Zeichenfolge schon eine Zahl, so wird diese ersetzt. Falls mit dem Parameter **ART** nichts anderes angegeben ist, wird bei jedem Auftreten einer Zeichenfolge, die mit dem Parameter **LNR** angegeben ist, die laufende Nummer um 1 erhöht. [ IX ]

Die größte erlaubte laufende Nummer ist 9 999 999.

**LNB** Zahlenwert, der auf die neue laufende Nummer aufaddiert werden soll. [ I ] <0>

Falls die neue laufende Nummer nicht mit 1 (bzw. mit dem mit dem Parameter **LNS** angegebenen Wert) beginnen soll, kann mit diesem Parameter ein Zahlenwert angegeben werden, nach dem sie beginnen soll. Die erste neue laufende Nummer ist also um 1 (bzw. um den mit dem Parameter **LNS** angegebenen Wert) höher als der mit dem Parameter **LNB** angegebene Wert.

Bei **MODUS=HOLE** und bei **MODUS=ERGAENZE** ist dieser Parameter nicht zugelassen.

**LNS** Schrittweite der neuen laufende Nummer. [ I ] <1>

Die erste neue laufende Nummer beginnt mit dem angegebenen Wert. Falls jedoch der Parameter **LNB** angegeben ist, wird die erste neue laufende Nummer um den dort angegeben Wert erhöht. Danach wird die laufende Nummer jeweils um den mit dem Parameter **LNS** angegebenen Wert erhöht.

**ART** Angabe, in welcher Weise die laufende Nummer hochgezählt werden soll. [ I ] <0>

0 = Bei jedem Auftreten einer Zeichenfolge, die mit dem Parameter **LNR** angegeben ist, soll die laufende Nummer um 1 erhöht werden.

1 = Die laufende Nummer soll nur dann um 1 erhöht werden, wenn in den Eingabedaten unmittelbar nach der Zeichenfolge, nach der sie eingesetzt werden soll, noch keine Zahl steht, oder wenn die dort in den Eingabedaten stehende Zahl zum ersten Mal vorkommt. Ist die dort stehende Zahl vorher schon in den Eingabedaten vorgekommen, so wird sie durch die gleiche Zahl ersetzt wie bei ihrem ersten Vorkommen.

**MSZ** Anzahl der Stellen, auf die die laufenden Nummern und die Verweisnummern mit führenden Nullen aufgefüllt werden sollen, falls die Nummer weniger Stellen hat. [ I ] <0>

## Aktualisieren der Verweisnummern

**VNR** Zeichenfolgen, hinter denen die Verweisnummer ersetzt werden soll. Diese Verweisnummer darf nicht fehlen. Sie muss außerdem genau einmal als laufende Nummer hinter einer Zeichenfolge stehen, die mit dem Parameter `LNR` angegeben ist (Ausnahme: Parameter `ART = 1`). Die Verweisnummer wird durch die gleiche Zahl ersetzt wie ihre zugehörige laufende Nummer. [ IX ]

Kommt die zugehörige laufende Nummer mehr als einmal vor, so wird der Verweis entsprechend dem ersten Vorkommen geändert. Fehlt die zugehörige laufende Nummer, so wird die Verweisnummer durch 0 ersetzt.

## Verweise auf Seiten- und Zeilennummern

Durch Angabe des Parameters `SK` wird dem Programm angezeigt, dass Verweise auf Seitennummern eingesetzt werden sollen. Wird zusätzlich der Parameter `ZK` angegeben, so werden auch Zeilennummern eingesetzt.

Die laufenden Nummern dienen in diesem Fall lediglich zur Kennzeichnung einer Stelle, auf die sich ein Verweis bezieht. Sie werden nicht verändert (Parameter `ART` nicht zugelassen). Die Verweisnummern bleiben ebenfalls unverändert. Sie dienen nur zur Angabe, auf welche Stelle (nämlich die Stelle, an der die »laufende Nummer« mit der gleichen Zahl steht) verwiesen werden soll.

Die Seitennummer der Seite, auf die sich der Verweis bezieht, wird im Text unmittelbar nach der Zeichenfolge eingesetzt, die mit dem Parameter `SK` angegeben ist. Diese Zeichenfolge muss im Text nach der Verweisnummer in der gleichen oder in der nachfolgenden Zeile vorkommen. Steht unmittelbar nach dieser Zeichenfolge schon eine Zahl, so wird diese ersetzt. Ist auch der Parameter `ZK` angegeben, so wird die Zeilenzahl der Zeile, auf die sich der Verweis bezieht, im Text nach der Zeichenfolge eingesetzt, die mit diesem Parameter angegeben ist. Diese Zeichenfolge muss im Text nach der Zeichenfolge für die Seitennummer vorkommen. Eine evtl. schon vorhandenen Zahl wird ersetzt. Seiten- und Zeilennummer werden für jeden Verweis nur einmal eingesetzt, auch wenn die mit den Parametern `SK` und `ZK` angegebenen Zeichenfolgen mehrfach vorkommen.

**SK** Zeichenfolgen, die die Stelle im Text kennzeichnen, an der die Seitennummer eingesetzt werden soll. [ IX ]

**ZK** Zeichenfolgen, die die Stelle im Text kennzeichnen, an der die Zeilennummer eingesetzt werden soll. [ IX ]

### Angaben zum Protokoll

- PR**            Angabe, welche laufenden Nummern und welche Verweisnummern protokolliert werden sollen. [ I ] <1>
- 0 = Nur diejenigen laufenden Nummern protokollieren,
    - die mehrfach vorkommen;
    - auf die nicht verwiesen wird;
 und nur diejenigen Verweisnummern protokollieren,
    - die als lfd. Nummer nicht vorkommen;
    - die als lfd. Nummer mehrfach vorkommen.
    - bei denen die Kennung für die Seiten- oder Zeilennummer fehlt.
  - 1 = Alle Nummern protokollieren.
  - 2 = wie 0, jedoch laufende Nummern, auf die nicht verwiesen wird, nicht protokollieren.

### Alphabetisches Verzeichnis der Parameter

<b>ART</b>	Angabe zum Hochzählen der laufenden Nummer . . . . .	1004
<b>BER</b>	Auswählen eines Bereichs aus der QUELL-Datei . . . . .	1003
<b>LNB</b>	Basis (Anfangswert) für laufende Nummer . . . . .	1004
<b>LNR</b>	Kennzeichen für laufende Nummer . . . . .	1004
<b>LNS</b>	Schrittweite für laufende Nummer . . . . .	1004
<b>MAX</b>	Maximum für Testzwecke . . . . .	1004
<b>MSZ</b>	Mindeststellenzahl für lfd. Nummern und Verweisnummern . . .	1004
<b>PAR</b>	Parameter-Konvention . . . . .	1003
<b>PR</b>	Angabe zum Umfang des Protokolls . . . . .	1006
<b>SK</b>	Kennzeichen für Seitennummer . . . . .	1005
<b>VNR</b>	Kennzeichen für Verweisnummer . . . . .	1005
<b>ZK</b>	Kennzeichen für Zeilennummer . . . . .	1005

\* \* \* \* \*

**#RAUFBEREITE**

---

Kommando . . . . .	1010
Leistung . . . . .	1011
Beispiel . . . . .	1012
Parameter . . . . .	1014
Einstellen der Parameter-Konvention . . . . .	1014
Vorbemerkungen zu typenabhängigen Ergänzungen . . . . .	1014
Vorbemerkungen zum KWIC-Index . . . . .	1015
Ausgeben eines Anfangstextes . . . . .	1015
Angaben zum Protokoll . . . . .	1015
Eingabedaten-Format . . . . .	1019
Eingabedaten-Begrenzung . . . . .	1019
Typ der Registereinträge . . . . .	1020
Definieren der Textteile der Registereinträge . . . . .	1021
Einfügen/Unterdrücken von Zeilenwechslern . . . . .	1021
Auswählen der Registereinträge bzw. Schlüsselwörter . . . . .	1022
Behandlung gleicher Textteile . . . . .	1024
Ersetzen von bzw. in Textteilen . . . . .	1024
Ergänzungen um die einzelnen Textteile . . . . .	1024
Positionieren der Textteile . . . . .	1025
Einfügen einer laufenden Nummer vor Textteilen . . . . .	1025
Ergänzungen um die laufende Nummer . . . . .	1025
Positionieren der laufenden Nummer . . . . .	1025
Einfügen der absoluten Häufigkeit hinter Textteilen . . . . .	1026
Ergänzungen um die absolute Häufigkeit . . . . .	1026
Positionieren der absoluten Häufigkeit . . . . .	1026
Einfügen der relativen Häufigkeit hinter Textteilen . . . . .	1026
Ergänzungen um die relative Häufigkeit . . . . .	1027
Positionieren der relativen Häufigkeit . . . . .	1027
Ergänzungen um die Referenzen . . . . .	1027
Definieren der Referenzteile . . . . .	1028
Behandlung der Referenzen . . . . .	1028
Behandlung gleicher Referenzteile . . . . .	1028
Ersetzen von bzw. in Referenzteilen . . . . .	1028
Ergänzungen um die einzelnen Referenzteile . . . . .	1029
Ergänzungen bei zusammengefassten Referenzen . . . . .	1029
Positionieren der Referenzteile . . . . .	1029
Einfügen einer referenzbezogenen Häufigkeit . . . . .	1030
Ergänzungen um die referenzbezogene Häufigkeit . . . . .	1030
Positionieren der referenzbezogenen Häufigkeit . . . . .	1030
Prüfen auf gleiches Schlüsselwort . . . . .	1030
Ergänzungen um das Schlüsselwort . . . . .	1031
Positionieren des Schlüsselwortes . . . . .	1031
Ergänzungen um den Kontext . . . . .	1032
Ergänzungen um den Referenz/Kontext . . . . .	1031



---

Einfügen eines lebenden Kolumnentitels . . . . .	1032
Ergänzungen um den lebenden Kolumnentitel . . . . .	1032
Ersetzen im lebenden Kolumnentitel . . . . .	1032
Zwischenüberschriften bei Anfangsbuchstaben-Wechsel . . . . .	1033
Begrenzen der Satzlänge . . . . .	1033
Ausgeben eines Endetextes . . . . .	1034
Alphabetisches Verzeichnis der Parameter . . . . .	1034

## Kommando

#RAUFBEREITE

QUELLE	= datei	Name der Datei mit den Registereinträgen, die aufbereitet werden sollen, bzw. mit den Einträgen für den KWIC-Index.
	= -STD-	Die Registereinträge, die aufbereitet werden sollen, bzw. die Einträge für den KWIC-Index stehen in der Standard-Text-Datei.
ZIEL	= -	* Nur Protokoll-Ausgabe.
	= datei	Name der Datei für das aufbereitete Register.
	= -STD-	Das aufbereitete Register in die Standard-Text-Datei ausgeben.
MODUS	= +	* Register aufbereiten; Eingabedaten enthalten eine Referenz.
	= -	Register aufbereiten; Eingabedaten enthalten keine Referenz.
	= KWIC	KWIC-Index aufbereiten.
LOESCHEN	= -	* Daten in der ZIEL- und in der PROTOKOLL-Datei nicht löschen.
	= +	Daten in der ZIEL- und in der PROTOKOLL-Datei zuerst löschen.
PARAMETER	= datei	Name der Datei mit den Parametern.
	= *	Die Parameter folgen auf das Kommando und sind mit *EOF abgeschlossen.
DATEN	= -	* Die Registereinträge stehen vollständig in der QUELL-Datei.
	= datei	Name der Datei mit dem Datenteil der Registereinträge bzw. der KWIC-Indexeinträge, der zum Sortieren nicht benötigt wurde.
	= -STD-	Der Datenteil der Registereinträge bzw. KWIC-Indexeinträge, der zum Sortieren nicht benötigt wurde, steht in der Standard-Daten-Datei.
PROTOKOLL	= -STD-	* Protokoll mit dem aufbereiteten Register in die Standard-Protokoll-Datei ausgeben.
	= datei	Name der Datei für das Protokoll mit dem aufbereiteten Register.
	= -	Kein Protokoll mit dem aufbereiteten Register erstellen.

## Leistung

Mit diesem Programm können Registereinträge bzw. Texteinheiten zu einem Register (z. B. Wortformenregister oder KWIC-Index) oder einem Verzeichnis (z. B. Bibliographie) zusammengefasst und aufbereitet werden.

Die Registereinträge bzw. Texteinheiten können zuvor mit dem Kommando #RVORBEREITE oder #SVORBEREITE erzeugt und zum Sortieren vorbereitet und dann mit dem Kommando #SORTIERE sortiert werden.

Das Programm bietet u. a. folgende Möglichkeiten: Ergänzung von Steuerzeichen und Kennzeichen, beliebiges Format der Druckausgabe, lebende Kolumnentitel, Zwischenüberschriften, Auswahl von bestimmten Registereinträgen bzw. Texteinheiten, hierarchische Gliederung in Haupt- und Untereinträge, Errechnen von absoluten und relativen Häufigkeiten.

## Hinweis

Zum Drucken des erstellten Registers bzw. Verzeichnisses mit dem Kommando #DRUCKE muss die PROTOKOLL-Datei, nicht die ZIEL-Datei verwendet werden. Die ZIEL-Datei wird dann benötigt, wenn das Register bzw. Verzeichnis weiterverarbeitet werden soll (z. B. zur Satzherstellung). Das Programm kann bei einem Aufruf jedoch nur entweder eine ZIEL-Datei oder eine PROTOKOLL-Datei erstellen.

## Beispiel

Erstellen und Ausdrucken von Wortformenregistern: Ein alphabetisch sortiertes mit Stellenangaben und ein nach Häufigkeit der Wortformen sortiertes mit Häufigkeitsangaben. Der Text, der FORMATIERE-Anweisungen enthält, steht in der Datei `text`. Ausgedruckt werden soll auf einem PostScript-Drucker, der den vom System vorgegebenen Druckernamen `ps1` hat:

```
#= Zerlegen des Textes in einzelne Wörter
```

```
#RV,text,-STD-,+,+,*
>SZ      {@}{-}\
>PM      +-
          Formatieranweisungen
TR       /${00}?{}{0} /&{!}{#}/@{!}{0}{Z:PM}/
          Sonderzeichen außer Sonderbuchstaben und Akzente
TR       /{C/SZ}//#. ?/%{1--2}{%}//
          Auszeichnungen
XX       /#{!}{Z:PM}//\//
          Zahlen
TA-      /{0}/
          Alphabetische Anordnung
          Wörter ohne Akzente vor solchen mit Akzenten
XS1      /ß/ss/%{1--2}{%}//
XS2      /ä/az/ö/oz/ü/uz/ß/sz/
A2       {}|%
SSL      20 20
*EOF
```

```
#= Sortieren der Wortformen nach Alphabet
```

```
#SO,-STD-,-STD-,17+40,+
```

```
#= Erstellen des Registers mit Stellenangaben
```

```
#RA,-STD-,-,+,+,*
SSL      20 20
RFF      1
DRT      PS-10
*EOF
```

```
#= Einrichten einer Hilfsdatei
```

```
#DA,hilf,,,-
```

#= Erstellen des Registers mit Häufigkeitsangaben

```
#RA,-STD-,hilf,+,+,*,,-
SSL      20 20
AH       1
VAH     / (/
NAH     /)/
REF      0
*EOF
```

#= Vorbereiten zum Sortieren nach den Häufigkeiten

```
#SV,hilf,-STD-,-,+,*
AK1     /(/
EK1     /)/
AEI     11
DEZ     5
SSL     5
*EOF
```

#= Sortieren der Wortformen nach Häufigkeit

```
#SO,-STD-,-STD-,1+5-F,+
```

#= Aufbereiten des Registers nach Häufigkeit

```
#RA,-STD-,-,-,-,*
SSL      5
DR       2 10 32 4
DRT     PS-10
*EOF
```

#= Ausdrucken der beiden Register

```
#DR,,PS-10,ps1
```

#= Löschen der Hilfsdatei

```
#LO,,hilf
```

Ein weiteres Beispiel mit dem Kommando #RAUFBEREITE ist in der Beschreibung des Kommandos #RVORBEREITE angegeben.

## Parameter

Jeder Parameter ist einer Parameterart zugeordnet. Sie ist jeweils als römische Zahl in [ ] angegeben. Damit ist festgelegt, in welcher Form die Angaben vom Programm erwartet werden. Die einzelnen Parameterarten sind in der Beschreibung »TUSTEP-Grundlagen« im Kapitel »{}-Parameter« bzw. »<>-Parameter« definiert.

Für manche Parameter gibt es voreingestellte Werte, die angenommen werden, falls die entsprechenden Werte nicht angegeben sind. Diese Voreinstellungen sind in < > angegeben.

Außer den im folgenden beschriebenen Parametern sind auch Parameter zur Definition von Zeichen- und Stringgruppen möglich. [ v ]

Ein »n« hinter der Parameterkennung steht für eine Ziffer in Spalte 7 des Parameters. Sie bezeichnet jeweils den Textteil bzw. den Referenzteil, auf den sich der Parameter bezieht.

### Einstellen der Parameter-Konvention

Ob die Parameter nach der »{}-Parameter-Konvention« oder nach der »<>-Parameter-Konvention« interpretiert werden, kann mit dem Kommando #PARAMETER, {} bzw. #PARAMETER, <> eingestellt werden. Darüber hinaus kann die Konvention auch mit folgendem Parameter eingestellt werden.

- PAR**            Parameter-Konvention einstellen.
- {}    Nachfolgende Parameter nach der »{}-Parameter-Konvention« interpretieren.
  - <>    Nachfolgende Parameter nach der »<>-Parameter-Konvention« interpretieren.

Die Einstellung mit dem Parameter PAR hat Vorrang vor der mit dem Kommando #PARAMETER gewählten Einstellung; sie gilt jeweils nur für die nachfolgenden Parameter bis zum nächsten Parameter PAR bzw. bis zum Ende der Parameter dieses Programms.

### Vorbemerkungen zu typenabhängigen Ergänzungen

Beim Aufbereiten eines Registers können an verschiedenen Stellen eines Registereintrags zusätzliche Zeichenfolgen eingefügt werden. Diese Zeichenfolgen können über entsprechende Parameter angegeben werden. Bei der Beschreibung dieser Parameter ist jeweils angegeben, dass die Ergänzungen »typenabhängig« erfolgen. Dies bedeutet, dass bei diesen Parametern für jeden möglichen Typ eines Registereintrags eine eigene Zeichenfolge angegeben werden kann. Die erste Zeichenfolge gilt jeweils für Registereinträge mit der Typnummer 1, die zweite für die mit der Typnummer 2 usw. Gegebenenfalls müssen leere Zeichenfolgen für nicht vorkommende Typnummern angegeben werden, damit die Zuordnung stimmt. Sind auf solchen Parametern nicht für alle vorkommenden Typen Zeichenfolgen (auch keine leeren) angegeben,

so wird für die fehlenden Zeichenfolgen jeweils die letzte mit dem Parameter angegebene Zeichenfolge eingefügt. Soll also z. B. unabhängig vom Typ des Registereintrags die gleiche Zeichenfolge eingefügt werden, so genügt es, diese Zeichenfolge mit dem entsprechenden Parameter einmal anzugeben.

## Vorbemerkungen zum KWIC-Index

Für dieses Programm entspricht ein KWIC-Index logisch gesehen einem Wortformenregister (jeder Registereintrag besteht aus einem einzigen Textteil), wobei

- der Textteil aus dem jeweiligen Schlüsselwort besteht,
- Referenzen nicht zusammengefasst werden,
- vor jeder Referenz eine neue Zeile begonnen wird,
- hinter jeder Referenz jeweils zusätzlich der Kontext, in dem das Schlüsselwort vorkommt, ausgegeben wird.

Deshalb können für einen KWIC-Index die selben Parameter wie für ein Wortformenregister verwendet werden. Jedoch sind einige Parameter bei einem KWIC-Index nicht sinnvoll, so z. B. Parameter, die das Zusammenfassen von Referenzen betreffen. Für den Kontext gibt es zusätzliche Parameter.

Wird der KWIC-Index in die PROTOKOLL-Datei ausgegeben, so wird der Kontext automatisch auf eine Zeile begrenzt.

Wird der KWIC-Index in die ZIEL-Datei ausgegeben, so wird zu den Schlüsselwörtern jeweils der gesamte Kontext ausgegeben. Es gibt in diesem Programm (noch) keine Möglichkeit, den Kontext in diesem Fall zu begrenzen. Um bei der Weiterverarbeitung das Schlüsselwort innerhalb des Kontextes auffinden zu können, sollte mit den Parametern VSW und NSW eine entsprechende Kennung eingefügt werden.

## Ausgeben eines Anfangstextes

**z** Text, der in die ZIEL-Datei ausgegeben werden soll. Bei jedem Trennzeichen wird ein neuer Satz begonnen. [ II ]

## Angaben zum Protokoll

**DR** Angaben zur Druckausgabesteuerung. [ I ]

Es können vier Zahlenwerte angegeben werden:

1. Zahl: Spalten <1>

Anzahl der Spalten, die auf jeder Seite nebeneinander gedruckt werden sollen

2. Zahl: Rand <10>

Anzahl der Leerstellen links der ersten Spalte

3. Zahl: Breite <64>

Anzahl der Zeichen je Spalte

4. Zahl: Zwischenraum <0>

Anzahl der Leerstellen zwischen den Spalten

**DRZ**

Zusätzliche Angaben zur Druckausgabesteuerung. [ I ]

Es können sieben Zahlenwerte angegeben werden:

1. Zahl: Kopftext <3>

Anzahl der Zeilen für den Kopftext, einschließlich der Leerzeilen

2. Zahl: Höhe <60>

Anzahl der Zeilen je Reihe, ohne die Zeilen für den Kopf- und Fußtext

3. Zahl: Fußtext <0>

Anzahl der Zeilen für den Fußtext, einschließlich der Leerzeilen

4. Zahl: Reihen <1>

Anzahl der Reihen pro Seite. Jede Reihe besteht aus sovielen Zeilen, wie mit der 2. Zahl angegeben wird.

5. Zahl: Wiederholung des Fußtextes <0>

Anzahl der Zeilen, die vom Fußtext der Seite (von der ersten nach unten gezählt) auch nach jeder Reihe (außer nach der untersten Reihe, nach der alle Zeilen des Fußtextes gedruckt werden) wiederholt werden sollen

6. Zahl: Leerzeilen <0>

Anzahl der Leerzeilen zwischen den einzelnen Reihen

7. Zahl: Wiederholung des Kopftextes <0>

Anzahl der Zeilen, die vom Kopftext der Seite (von der letzten nach oben gezählt) auch vor jeder Reihe (außer vor der obersten Reihe, vor der alle Zeilen des Kopftextes gedruckt werden) wiederholt werden sollen

**KT**

Textteile, die als Kopftext oben auf jeder Seite gedruckt werden sollen.

[ II ] < : &Q3 @/ &D2 &U2 &#6 : >

In den Textteilen werden folgende Steueranweisungen durch die entsprechenden aktuellen Werte ersetzt:

&Q1 Projektname der QUELL-Datei (ohne Dateiname)

&Q2 Dateiname der QUELL-Datei (ohne Projektname)

&Q3 Projekt- und Dateiname der QUELL-Datei

&D0 Wochentag (z. B. Sonntag)

&D1 Datum xx.xx.xx (z. B. 02.04.96)



- &D2 Datum xx. xxx. xxxx (z. B. 2. Apr. 2008)  
 &D3 Datum xx. xxxxxxxxxxx xxx (z. B. 2. April 2008)  
 &U1 Uhrzeit xx.xx (z. B. 12.00)  
 &U2 Uhrzeit xx:xx (z. B. 12:00)  
 &#n Seitennummer mit maximal n (n=1 bis 6) Stellen

Die Seitennummer kann nur einmal eingesetzt werden. Wird für die Seitennummer »- &#n -« (n=1 bis 6) angegeben, so wird die Seitennummer in die Mitte zwischen die Minuszeichen eingesetzt; die Minuszeichen werden bis auf ein Leerzeichen als Zwischenraum nach rechts bzw. links zur Seitennummer hin verschoben.

Werden pro Seite mehrere Reihen (4. Zahlwert des Parameters DRZ) ausgegeben und wird &#n in einer Zeile angegeben, die für jede Reihe wiederholt wird (4. bzw. 7. Zahlwert des Parameters DRZ), so wird statt der Seitennummer die laufende Nummer der jeweiligen Reihe eingesetzt.

Jeder der Textteile kann durch die Formatieranweisungen »@z« und »@/« in drei Teile gegliedert sein:

linksbündig @z auf Mitte zentriert @/ rechtsbündig  
 Diese einzelnen Teile werden linksbündig, auf Mitte zentriert und rechtsbündig eingesetzt. Jeder einzelne Teil kann (bei Teil zwei und drei einschließlich der davor stehenden Formatieranweisung) fehlen.

Jeder Textteil wird in eine neue Zeile des Kopftextes gedruckt. Bei mehrspaltigem Druck können auch Textteile für die einzelnen Spalten angegeben werden. Dazu gibt es folgende Regelung:

Beginnt ein Textteil mit »\* :«, so wird der Rest des Textteils über jede Spalte in den Kopftext eingetragen. Steht anstelle des Sterns eine Zahl, so wird der Rest des Textteils über die durch die Zahl bezeichnete Spalte eingetragen. Hat die Zahl den Wert 0, so gilt der Rest des Textteils für die ganze Zeile. Beginnt ein Textteil nicht in der beschriebenen Weise, so wird »0 :« angenommen (Normalfall).

Mit einem Textteil, der für eine ganze Zeile des Kopftextes gilt, wird immer eine neue Zeile begonnen. Ein Textteil, der über einer bestimmten Spalte stehen soll, wird in die gleiche Zeile wie der vorangehende Textteil eingetragen, falls diese Zeile nicht schon einen Text für die ganze Zeile oder für diese oder eine weiter rechts stehende Spalte enthält; andernfalls wird mit diesem Textteil eine neue Zeile begonnen.

- KTZ** Angabe, ob auf der ersten Seite der Kopftext gedruckt werden soll.  
 [ I ] <1>  
 0 = Kopftext unterdrücken  
 1 = Kopftext drucken
- FT** Textteile, die als Fußtext unten auf jeder Seite gedruckt werden sollen.  
 [ II ] <>

Es gelten die gleichen Regelungen wie für den Kopftext (Parameter  $\kappa T$ ). Die Seitennummer kann jedoch nicht in den Fußtext eingesetzt werden, wenn sie schon in den Kopftext eingesetzt worden ist.

**DRT**

Druckertyp, für den die Daten aufbereitet werden. [ XI ]

Dieser Parameter ist obligat.

Hinweis: Mit dem Kommando #LISTE, DRUCKERTYPEN werden die definierten Druckertypen aufgelistet.

**NR**

Erste Seitennummer für die Ausgabe. [ I ] <1>

**ROM**

Angabe, ob die Seitenzahl in arabischen Ziffern oder als römische Zahlen in den Kopf- bzw. den Fußtext eingesetzt werden soll. [ I ] <0>

0 = arabische Ziffern

1 = römische Zahlen mit Kleinbuchstaben

2 = römische Zahlen mit Großbuchstaben

**DRE**

n Druckausgabesteuerung für einen neuen Eintrag, der mit Textteil n beginnt. [ I ]

Es können sechs Zahlenwerte angegeben werden:

1. Zahl: Vorschub vor Eintrag <1>

Anzahl der Zeilenvorschübe (= Leerzeilen + 1) vor einem neuen Eintrag. Zwischen zwei Einträgen wird der größere Wert von der 1. und 3. Zahl dieses Parameters zur Vorschubsteuerung verwendet.

2. Zahl: Restzeilen <1>

Anzahl der Zeilen, die vor einem neuen Eintrag in einer Spalte noch mindestens frei sein müssen, damit der Eintrag noch in dieser Spalte begonnen wird. Andernfalls wird der Eintrag in einer neuen Spalte begonnen.

3. Zahl: Vorschub nach Eintrag <1>

Anzahl der Zeilenvorschübe (= Leerzeilen + 1) nach einem Eintrag. Zwischen zwei Einträgen wird der größere Wert von der 1. und 3. Zahl dieses Parameters zur Vorschubsteuerung verwendet.

4. Zahl: Einrücken bei neuem Eintrag <0>

Anzahl der Leerstellen am Anfang eines neuen Eintrags

5. Zahl: Einrücken bei Fortsetzungszeilen <4>

Anzahl der Leerstellen am Anfang von Fortsetzungszeilen

6. Zahl: Vorschub vor Fortsetzungszeilen <1>

Anzahl der Zeilenvorschübe (= Leerzeilen + 1) vor Fortsetzungszeilen

- DRA** n Druckausgabesteuerung bei Anfangsbuchstaben-Wechsel bei einem mit Textteil n beginnenden Eintrag. [ I ]
- Die Angaben entsprechen denen zu Parameter DRE.

## Eingabedaten-Format

Die vier folgenden Parameter müssen mit den gleichen Werten angegeben werden, wie beim Vorbereiten der Daten mit dem Kommando #RVORBEREITE bzw. #SVORBEREITE.

Für die Parameter SNL und SSL gibt es jedoch eine Ausnahme: Wenn beim Sortieren der vorbereiteten Daten die Sortiernummer und/oder der Sortierschlüssel eliminiert wurde (durch eine entsprechende Angabe zur Spezifikation TILGEN beim Kommando #SORTIERE), muss mit den Parametern SNL und/oder SSL der Wert Null angegeben werden.

Der Parameter SSL ist obligat, die anderen können weggelassen werden, wenn sie auch beim Vorbereiten der Daten nicht angegeben wurden.

- IRL** Interne Referenzlänge. [ I ] <14>
- Angaben wie beim gleichnamigen Parameter des Kommandos #RVORBEREITE bzw. #SVORBEREITE.
- SNL** Länge der Sortiernummer. [ I ] <0>
- Angaben wie beim gleichnamigen Parameter des Kommandos #RVORBEREITE bzw. #SVORBEREITE.
- SSL** Länge der Sortierschlüssel. [ I ] <0, 0, 0, 0, 0, 0, 0, 0, 0, 0>
- Dieser Parameter ist obligat.
- Angaben wie beim gleichnamigen Parameter des Kommandos #RVORBEREITE bzw. #SVORBEREITE.
- HFL** Länge der Häufigkeitsangabe. [ I ] <0>
- Angaben wie beim gleichnamigen Parameter des Kommandos #RVORBEREITE bzw. #SVORBEREITE.

## Eingabedaten-Begrenzung

- MAX** Angaben für Testzwecke, wieviele Einträge maximal eingelesen bzw. ausgegeben werden sollen bzw. wieviele Seiten maximal aufbereitet werden sollen. [ I ]

Es können drei Zahlenwerte angegeben werden:

1. Zahl: Einzulesende Einträge <999999999>
2. Zahl: Auszugebende Einträge <999999999>
3. Zahl: Auszugebende Seiten <999999999>

## Typ der Registereinträge

Jeder Registereintrag ist einem Typ zugeordnet; anhand dieser Typen können die einzelnen Registereinträge differenziert werden.

Bei `MODUS=-` enthalten die Registereinträge jedoch keine Angabe über den Typ; in diesem Fall werden die Registereinträge so behandelt, als wären sie alle dem Typ 1 zugeordnet.

<b>TYP</b>	Umdefinieren des Typs 0 auf einen anderen Typ. [ I ] <1> 0 = Eintrag (ggf. einschließlich der dazugehörenden Referenz) nicht ausgeben n = Eintrag behandeln wie Einträge vom Typ n
<b>TXT</b>	Typenabhängige Angabe, ob der Eintrag (ggf. einschließlich der dazugehörenden Referenz) ausgegeben werden soll (1) oder nicht (0). [ I ] <1, 1, . . . > Für die Auswahl ist jeweils der ursprüngliche Typ des Registereintrags maßgebend; die Parameter <code>TFT</code> und <code>TFR</code> werden ggf. erst danach berücksichtigt.
<b>REF</b>	Typenabhängige Angabe, ob die Referenz ausgegeben werden soll (1) oder nicht (0). [ I ] <1, 1, . . . > Für die Auswahl ist jeweils der ursprüngliche Typ des Registereintrags maßgebend; der Parameter <code>TFR</code> wird ggf. erst danach berücksichtigt.

Gleiche Registereinträge bzw. gleiche Textteile von Registereinträgen werden nur zusammengefasst, wenn sie auch dem selben Typ zugeordnet sind. Dabei werden ggf. die mit dem Parameter `TFT` angegebenen Umdefinitionen des ursprünglichen Typs berücksichtigt. Entsprechendes gilt für das Zusammenfassen von gleichen Referenzen bzw. gleichen Referenzteilen, wobei ggf. die mit dem Parameter `TFR` angegebenen Umdefinitionen berücksichtigt werden.

Soll das Programm z. B. Registereinträge zusammenfassen, die ursprünglich verschiedenen Typen zugeordnet sind, jedoch die dazugehörenden Referenzen typenabhängig unterscheiden (z. B. durch Kursivdruck oder eine Ergänzung, vgl. dazu die »Vorbemerkungen zu typenabhängigen Ergänzungen« Seite 1014), so müssen

- die Textteile dieser Registereinträge durch Umdefinition der Typen mit dem Parameter `TFT` jeweils dem selben Typ zugeordnet werden,
- die Referenzteile mit dem Parameter `TFR` jeweils verschiedenen Typen zugeordnet werden, falls der Parameter `TFR` angegeben wird.

**TFT**    n    Umdefinieren der Typen für den Textteil n. [ I ] <1, 2, 3, . . . >

**TFR**    n    Umdefinieren der Typen für den Referenzteil n. [ I ] <1, 2, 3, . . . >

## Definieren der Textteile der Registereinträge

Registereinträge, die aus mehreren Teilen (z. B. Haupt- und Unterbegriffen) bestehen, können anhand von Trennzeichen unterteilt werden. Mit dem Parameter **TT** muss in diesem Fall angegeben werden, wieviele Textteile maximal unterschieden werden sollen und mit dem Parameter **TR** müssen die möglichen Trennzeichen angegeben werden. Sollen die Registereinträge nicht unterteilt werden, braucht keiner der folgenden Parameter angegeben zu werden.

<b>TT</b>	Anzahl der Textteile (bis zu 9), aus denen ein Registereintrag maximal besteht. [ I ] <1>
<b>TR</b>	Zeichenfolgen, die als Trennzeichen zwischen den Textteilen interpretiert werden sollen. [ IX ]
<b>TRN</b>	Angabe parallel zu <b>TR</b> , als wieviertes Trennzeichen die entsprechende Zeichenfolge mindestens gelten soll (Trennzeichennummer). [ I ] <1, 1, . . . >
<b>TRV</b>	Angabe parallel zu <b>TR</b> , vom wievielten Trennzeichen an (d. h. ab dem wievielten Textteil) die entsprechende Zeichenfolge erst als Trennzeichen gelten soll. [ I ] <1, 1, . . . >
<b>TRB</b>	Angabe parallel zu <b>TR</b> , bis zu welchen Trennzeichen einschließlich (d. h. zum wievielten Textteil) die entsprechende Zeichenfolge als Trennzeichen gelten soll. [ I ] <8, 8, . . . >
<b>TRU</b>	Angabe parallel zu <b>TR</b> , ob das Trennzeichen bei der Ausgabe des Registereintrags unterdrückt werden soll (1) oder nicht (0). [ I ] <0, 0, . . . >

Unabhängig von der Angabe zu diesem Parameter wird das Trennzeichen zwischen zwei Textteilen unterdrückt, wenn an dieser Stelle bei der Ausgabe des Registereintrags auf Grund der nachfolgend beschriebenen Parameter **NZB** oder **NZ** eine neue Zeile begonnen wird.

## Einfügen/Unterdrücken von Zeilenwechsell

<b>NZB</b>	Neue Zeile hinter jedem Textteil (unter Wegfall des Trennzeichens) bis einschließlich hinter dem Textteil, dessen Nummer hier angegeben ist. [ I ] <0>
<b>NZ</b>	n Neue Zeile (unter Wegfall des Trennzeichens) hinter Textteil n: 0 = nein, 1 = ja. [ I ] <0>
<b>NZU</b>	Neue Zeile unterdrücken ab dem angegebenen Textteil m: wenn in aufeinander folgenden Einträgen die ersten m-1 Textteile identisch sind, soll vor solchen Einträgen kein Zeilenwechsel vorgenommen werden. [ I ] <10>

## Auswählen der Registereinträge bzw. Schlüsselwörter

Falls nicht alle Registereinträge (bei `MODUS=KWIC` alle Schlüsselwörter) ausgegeben werden sollen, so können sie (zusätzlich zur Auswahl mit dem Parameter `TYP`) mit den folgenden Parametern ausgewählt werden.

Stimmt der  $n$ -te Textteil eines Registereintrags (bei `MODUS=KWIC` das Schlüsselwort) mit einem zum Parameter `T+N`, `T+U` oder `T+` angegebenen überein, wird der Registereintrag ausgegeben. In diesem Fall unterbleiben weitere Überprüfungen auf Grund der übrigen Parameter dieses Abschnitts. Sind keine anderen Parameter dieses Abschnitts angegeben, so werden nur diese Registereinträge ausgegeben.

**T+N**     $n$     Textteile, von denen einer mit dem Textteil  $n$  eines Registereintrags bzw. mit dem Schlüsselwort übereinstimmen muss, damit der Registereintrag ausgegeben wird. [ III ]

Groß- und Kleinbuchstaben werden nicht unterschieden.

**T+U**     $n$     Textteile, von denen einer mit dem Textteil  $n$  eines Registereintrags bzw. mit dem Schlüsselwort übereinstimmen muss, damit der Registereintrag ausgegeben wird. [ III ]

Groß- und Kleinbuchstaben werden unterschieden.

**T+**         $n$     Textteile, von denen einer mit dem Textteil  $n$  eines Registereintrags bzw. mit dem Schlüsselwort übereinstimmen muss, damit der Registereintrag ausgegeben wird. [ III ]

Beim Vergleich wird die Sonder-Sortierfolge zugrunde gelegt; d. h. die Sonderzeichen, die mit `x` bzw. `^x` codiert sind (z. B. `!` und `^!`, `&` und `^&`) sowie die Groß- und Kleinbuchstaben werden nicht unterschieden.

Stimmt der  $n$ -te Textteil eines Registereintrags (bei `MODUS=KWIC` das Schlüsselwort) mit einem zum Parameter `T-N`, `T-U` oder `T-` angegebenen überein, wird der Registereintrag nicht ausgegeben. In diesem Fall unterbleiben weitere Überprüfungen auf Grund der übrigen Parameter dieses Abschnitts.

**T-N**     $n$     Textteile, von denen keiner mit dem Textteil  $n$  eines Registereintrags bzw. mit dem Schlüsselwort übereinstimmen darf, damit der Register- eintrag ausgegeben wird. [ III ]

Groß- und Kleinbuchstaben werden nicht unterschieden.

**T-U**     $n$     Textteile, von denen keiner mit dem Textteil  $n$  eines Registereintrags bzw. mit dem Schlüsselwort übereinstimmen darf, damit der Register- eintrag ausgegeben wird. [ III ]

Groß- und Kleinbuchstaben werden unterschieden.

**T-**         $n$     Textteile, von denen keiner mit dem Textteil  $n$  eines Registereintrags bzw. mit dem Schlüsselwort übereinstimmen darf, damit der Register- eintrag ausgegeben wird. [ III ]

Beim Vergleich wird die Sonder-Sortierfolge zugrunde gelegt; d. h. die Sonderzeichen, die mit `x` bzw. `^x` codiert sind (z. B. `!` und `^!`, `&` und `^&`) sowie die Groß- und Kleinbuchstaben werden nicht unterschieden.

Mit den Parametern **ZF+**, **TA+** und **TE+** können Bedingungen angegeben werden, unter denen ein Registereintrag ausgegeben werden soll. Falls von diesen Parametern einer oder mehrere angegeben sind, muss mindestens eine dieser Bedingungen erfüllt sein, damit der Registereintrag ausgegeben wird.

- ZF+**     n     Zeichenfolgen, von denen mindestens eine im n-ten Textteil des Registereintrags bzw. im Schlüsselwort vorkommen muss, damit der Registereintrag ins Register ausgegeben wird. [ IX ]
- TA+**     n     Zeichenfolgen, von denen mindestens eine mit dem Anfang des n-ten Textteils des Registereintrags bzw. dem Anfang des Schlüsselwortes übereinstimmen muss, damit der Registereintrag ins Register ausgegeben wird. [ VIII-a ]
- TE+**     n     Zeichenfolgen, von denen mindestens eine mit dem Ende des n-ten Textteils des Registereintrags bzw. dem Ende des Schlüsselwortes übereinstimmen muss, damit der Registereintrag ins Register ausgegeben wird. [ VIII-b ]

Mit den Parametern **ZF-**, **TA-** und **TE-** können Bedingungen angegeben werden, unter denen ein Registereintrag nicht ausgegeben werden soll. Falls von diesen Parametern einer oder mehrere angegeben sind, genügt es, wenn eine dieser Bedingungen erfüllt ist, damit der Registereintrag nicht ausgegeben wird.

- ZF-**     n     Zeichenfolgen, von denen keine im n-ten Textteil des Registereintrags bzw. im Schlüsselwort vorkommen darf, damit der Registereintrag ins Register ausgegeben wird. [ IX ]
- TA-**     n     Zeichenfolgen, von denen keine mit dem Anfang des n-ten Textteils bzw. dem Anfang des Schlüsselwortes übereinstimmen darf, damit der Registereintrag ins Register ausgegeben wird. [ VIII-a ]
- TE-**     n     Zeichenfolgen, von denen keine mit dem Ende des n-ten Textteils des Registereintrags bzw. dem Ende des Schlüsselwortes übereinstimmen darf, damit der Registereintrag ins Register ausgegeben wird. [ VIII-b ]

Sind sowohl einer oder mehrere der Parameter **ZF+**, **TA+** und **TE+** als auch einer oder mehrere der Parameter **ZF-**, **TA-** und **TE-** angegeben, so muss ein Registereintrag mindestens eine der Bedingungen der Parameter **ZF+**, **TA+** und **TE+** erfüllen und darf keine der Bedingungen der Parameter **ZF-**, **TA-** und **TE-** erfüllen, um ausgegeben zu werden.

Mit den beiden folgenden Parametern können Registerinträge über die Häufigkeit ausgewählt werden. Werden dadurch Registerinträge unterdrückt, die aus mehreren Textteilen bestehen, so werden diese trotzdem bei der Berechnung der Häufigkeit von übergeordneten Textteilen mitgezählt.

- AHM**    n     Registereintrag nur ausgeben, falls der Textteil n mindestens die angegebene absolute Häufigkeit hat. [ I ]
- AHH**    n     Registereintrag nur ausgeben, falls der Textteil n höchstens die angegebene absolute Häufigkeit hat. [ I ]

## Behandlung gleicher Textteile

<b>GKU</b>		Angabe, ob Groß- und Kleinbuchstaben beim Vergleich auf Übereinstimmung für das Zusammenfassen von Registereinträgen bzw. von Referenzen unterschieden werden sollen oder nicht.  Es können zwei Zahlenwerte angegeben werden: 1. Zahl: Zusammenfassen von Registereinträgen <0> 0 = Groß- und Kleinbuchstaben nicht unterscheiden 1 = Groß- und Kleinbuchstaben unterscheiden 2. Zahl: Zusammenfassen von Referenzen <0> 0 = Groß- und Kleinbuchstaben nicht unterscheiden 1 = Groß- und Kleinbuchstaben unterscheiden
<b>TTE</b>	n	Zeichenfolgen, die (typenabhängig) als Ersatz für den (wegen Identität mit dem gleichen Textteil des vorhergehenden Eintrags) wegfallenden Textteil n eingesetzt werden sollen. [ II ] < -- > < -- -- > ...
<b>TTW</b>	n	Wiederholung ab dem angegebenen Textteil, falls mit dem vorangehenden Registereintrag identische übergeordnete Textteile bis Teil n wegfallen würden. [ I ] <n+1>

## Ersetzen von bzw. in Textteilen

<b>TTT</b>	n	Austauschen des Textteils n. [ IV ]
<b>XTT</b>	n	Austauschen von Zeichenfolgen in Textteil n. [ X ]
<b>LTT</b>	n	Zeichenfolgen, die (typenabhängig) als Ersatz für einen leeren Textteil n eingesetzt werden sollen. [ II ]  Wenn dieser Parameter angegeben und der Textteil n leer ist, wird nur die dem Typ entsprechende Zeichenfolge eingesetzt; die eventuell mit den Parametern VTT und NTT angegebenen Zeichenfolgen entfallen.
<b>ITT</b>	n	Zeichenfolgen, die (typenabhängig) als Ersatz für einen Textteil n eingesetzt werden sollen, der wegfallen würde (weil dieser und alle übergeordnete Textteile mit dem vorangehenden Registereintrag identisch sind), er aber auf Grund einer entsprechenden Angabe zum Parameter TTW wiederholt werden sollte. [ II ]

## Ergänzungen um die einzelnen Textteile

<b>VTT</b>	n	Zeichenfolgen, die (typenabhängig) vor dem Textteil n ergänzt werden sollen. [ II ]
<b>NTT</b>	n	Zeichenfolgen, die (typenabhängig) nach dem Textteil n ergänzt werden sollen. [ II ]



## Positionieren der Textteile

Die beiden folgenden Parameter wirken sich nur im Protokoll aus.

- TTP** n Position, die nach Textteil n erreicht sein soll (ggf. wird mit Leerstellen aufgefüllt). [ I ] <0>
- TTZ** n Zusatz zu TTP: In dem durch TTP nach rechts begrenzten Feld soll der Textteil n linksbündig (0) bzw. rechtsbündig (1) stehen. [ I ] <0>

## Einfügen einer laufenden Nummer vor Textteilen

Eine laufende Nummer wird nur eingefügt, wenn mit dem Parameter LN ein Zahlenwert ungleich Null angegeben wird.

- LN** n Länge des Feldes, das vor Textteil n für die laufende Nummer freigehalten werden soll. [ I ] <0>
- Benötigt die laufende Nummer weniger Stellen als angegeben, so wird sie rechtsbündig in das Feld eingespeichert; benötigt sie mehr, so wird das Feld entsprechend vergrößert.
- LNB** n Basiswert, der auf die laufende Nummer für Textteil n aufaddiert werden soll. [ I ] <0>
- LNN** n Die laufende Nummer für Textteil n soll auf 0 zurückgesetzt werden, wenn der Textteil mit der angegebenen Nummer (oder ein übergeordneter) wechselt. [ I ] <0>
- LNW** n Bei Wiederholung von Textteil n (vgl. Parameter TTW) soll die laufende Nummer ebenfalls wiederholt werden (1) bzw. nicht wiederholt werden (0). [ I ] <1>

## Ergänzungen um die laufende Nummer

- VLN** n Zeichenfolgen, die (typenabhängig) vor der laufenden Nummer vor Textteil n ergänzt werden sollen. [ II ] < >
- NLN** n Zeichenfolgen, die (typenabhängig) nach der laufenden Nummer vor Textteil n ergänzt werden sollen. [ II ] < >

## Positionieren der laufenden Nummer

Die beiden folgenden Parameter wirken sich nur im Protokoll aus.

- LNP** n Position, die nach der laufenden Nummer erreicht sein soll (ggf. wird mit Leerstellen aufgefüllt). [ I ] <0>
- LNZ** n Zusatz zu LNP: In dem durch LNP nach rechts begrenzten Feld soll die laufende Nummer linksbündig (0) bzw. rechtsbündig (1) stehen. [ I ] <0>

## Einfügen der absoluten Häufigkeit hinter Textteilen

Die absolute Häufigkeit wird nur eingefügt, wenn mit dem Parameter **AH** ein Zahlenwert ungleich Null angegeben wird.

**AH**     n     Länge des Feldes, das hinter Textteil n für die absolute Häufigkeit freigehalten werden soll. [ I ] <0>

Benötigt die Häufigkeitsangabe weniger Stellen als angegeben, so wird sie rechtsbündig in das Feld eingespeichert; benötigt sie mehr, so wird das Feld entsprechend vergrößert.

**AHW**    n     Bei Wiederholung von Textteil n soll auch die absolute Häufigkeit wiederholt werden (1) bzw. nicht wiederholt werden (0). [ I ] <0>

## Ergänzungen um die absolute Häufigkeit

**VAH**    n     Zeichenfolgen, die (typenabhängig) vor der absoluten Häufigkeit hinter Textteil n ergänzt werden sollen. [ II ] < >

**NAH**    n     Zeichenfolgen, die (typenabhängig) nach der absoluten Häufigkeit hinter Textteil n ergänzt werden sollen. [ II ] <>

## Positionieren der absoluten Häufigkeit

Die beiden folgenden Parameter wirken sich nur im Protokoll aus.

**AHP**    n     Position, die nach der absoluten Häufigkeit hinter Textteil n erreicht sein soll (ggf. wird mit Leerstellen aufgefüllt). [ I ] <0>

**AHZ**    n     Zusatz zu **AHP**: In dem durch **AHP** nach rechts begrenzten Feld soll die absolute Häufigkeit hinter Textteil n linksbündig (0) bzw. rechtsbündig (1) stehen. [ I ] <0>

## Einfügen der relativen Häufigkeit hinter Textteilen

Die relative Häufigkeit wird nur eingefügt, wenn mit dem Parameter **RH** ein Zahlenwert ungleich Null angegeben wird.

**RH**     n     Länge des Feldes, das hinter Textteil n für die relative Häufigkeit freigehalten werden soll. [ I ] <0>

Benötigt die Häufigkeitsangabe weniger Stellen als angegeben, so wird sie rechtsbündig in das Feld eingespeichert; benötigt sie mehr, so wird das Feld entsprechend vergrößert.

**RHD**    n     Dezimalstellen (hinter dem Komma) für die relative Häufigkeit hinter Textteil n. [ I ] <2>

**RHB**    n     Die relative Häufigkeit hinter Textteil n bezieht sich nicht auf die Gesamthäufigkeit (s. u.), sondern auf die Häufigkeit des übergeordneten Textteils, dessen Nummer hier angegeben ist. [ I ] <0>

**RHW** n Bei Wiederholung von Textteil n soll auch die relative Häufigkeit wiederholt werden (1) bzw. nicht wiederholt werden (0). [ I ] <0>

Bei der Berechnung der relativen Häufigkeit wird als Gesamthäufigkeit die Anzahl der Sätze in der QUELL-Datei (d. i. die Anzahl der einzelnen, noch nicht zusammengefassten Registereinträge) genommen. Dies ist jedoch nicht immer die richtige Gesamthäufigkeit; z. B. dann nicht, wenn die einzelnen Registereinträge selbst schon Häufigkeiten (vgl. Parameter HFL auf Seite 1019) enthalten. In diesem Fall muss die richtige Gesamthäufigkeit mit folgendem Parameter angegeben werden.

**GHF** Gesamthäufigkeit, die bei der Berechnung der relativen Häufigkeit zugrunde gelegt werden soll. [ I ] <Anzahl der Sätze der QUELL-Datei>

### Ergänzungen um die relative Häufigkeit

**VRH** n Zeichenfolgen, die (typenabhängig) vor der relativen Häufigkeit hinter Textteil n ergänzt werden sollen. [ II ] < >

**NRH** n Zeichenfolgen, die (typenabhängig) nach der relativen Häufigkeit hinter Textteil n ergänzt werden sollen. [ II ] <^%>

### Positionieren der relativen Häufigkeit

Die beiden folgenden Parameter wirken sich nur im Protokoll aus.

**RHP** n Position, die nach der relativen Häufigkeit hinter Textteil n erreicht sein soll (ggf. wird mit Leerstellen aufgefüllt). [ I ] <0>

**RHZ** n Zusatz zu RHP: In dem durch RHP nach rechts begrenzten Feld soll die relative Häufigkeit hinter Textteil n linksbündig (0) bzw. rechtsbündig (1) stehen. [ I ] <0>

### Ergänzungen um die Referenzen

**VRF** Zeichenfolgen, die (abhängig vom Typ des letzten Textteils) vor der ersten Referenz ergänzt werden sollen. [ II ] < > bei MODUS=+, <> bei MODUS=KWIC

**NRF** Zeichenfolgen, die (abhängig vom Typ des letzten Textteils) nach der letzten Referenz ergänzt werden sollen. [ II ] <> bei MODUS=+, < > bei MODUS=KWIC

**ZRF** n Zeichenfolgen, die (abhängig vom Typ des letzten Textteils) zwischen den einzelnen Referenzen ergänzt werden sollen, wenn die zweite der durch ZRF getrennten Referenzen mit Referenzteil n beginnt. [ II ] < >

## Definieren der Referenzteile

- RFL** Zahlenwerte, die die Längen der einzelnen Referenzteile angeben (vgl. Kommando #RVORBEREITE). [ I ] <6>
- Durch die Anzahl der angegebenen Zahlenwerte (bis zu 9) wird die Anzahl der Referenzteile festgelegt.

## Auswählen der Referenzen

- MRF** Auch die mit dem Kommando #RVORBEREITE durch den Parameter ORF als nicht auszugehend markierten Referenzen sollen ausgegeben werden (1) oder nicht (0). [ I ] <0>
- RFF** Aufeinander folgende, lückenlos aufsteigende Referenzen sollen einzeln aufgeführt werden (0) bzw. nur unter sich zusammengefasst werden (1) bzw. auch mit Paaren von Referenzen zusammengefasst werden, die sich durch die Parameter VON und BIS vom Kommando #RVORBEREITE ergeben (2). [ I ] <0>
- RFU** Ausgabe der Referenzen unterdrücken, wenn mehr als die angegebene Anzahl Referenzen auszugeben wären. [ I ] <999999>
- Hinweis: Mit dem Parameter REF (siehe Seite 1020) können die Referenzen von Registereinträgen, die bestimmten Typen zugeordnet sind, unterdrückt werden.
- RTU** n Referenzteil n unterdrücken (1) oder nicht (0). [ I ] <0>

## Behandlung gleicher Referenzteile

- RTE** n Zeichenfolgen, die (typenabhängig) als Ersatz für einen (wegen Identität mit dem gleichen Teil der vorhergehenden Referenz) wegfallenden Referenzteil n eingesetzt werden sollen. [ II ] <>
- RTW** n Wiederholung ab dem Referenzteil, dessen Nummer hier angegeben ist, falls mit der vorangehenden Referenz identische übergeordnete Referenzteile bis Teil n wegfallen würden. [ I ] <n+1>

## Ersetzen von bzw. in Referenzteilen

- XRT** n Austauschen von Zeichenfolgen im Referenzteil n. [ X ]
- TRT** n Austauschen des Referenzteils n. [ IV ]
- LRT** n Zeichenfolgen, die (typenabhängig) als Ersatz für einen leeren Referenzteil n eingesetzt werden sollen. [ II ] <>
- Wenn dieser Parameter angegeben und der Referenzteil n (eventuell auf Grund des Parameters XRT) leer ist, wird nur die dem Typ entsprechende Zeichenfolge eingesetzt; die eventuell mit den Parametern VRT und NRT angegebenen Zeichenfolgen entfallen.

## Ergänzungen um die einzelnen Referenzteile

- VRT**     n     Zeichenfolgen, die (typenabhängig) vor dem Referenzteil n ergänzt werden sollen. [ II ] <>
- NRT**     n     Zeichenfolgen, die (typenabhängig) nach dem Referenzteil n ergänzt werden sollen. [ II ] <>

## Ergänzungen bei zusammengefassten Referenzen

Referenzen werden zu Paaren zusammengefasst, wenn es mit dem Parameter **RFF** verlangt wurde und wenn für die betreffenden Referenzfolgen nicht mit den Parametern **F1**, **F2**, **F3** eine andere Regelung getroffen wurde. Darüber hinaus werden die mit dem Kommando **#RVORBEREITE** durch die Parameter **VON** und **BIS** gekennzeichneten Referenzen zu Paaren zusammengefasst; evtl. dazwischen liegende Referenzen werden übergangen.

- F**            Zeichenfolgen, die (typenabhängig) hinter einer mit dem Kommando **#RVORBEREITE** durch den Parameter **MF** markierten Referenz ergänzt werden sollen. [ II ] <f>
- FF**           Zeichenfolgen, die (typenabhängig) hinter einer mit dem Kommando **#RVORBEREITE** durch den Parameter **MF** markierten Referenz ergänzt werden sollen. [ II ] <ff>
- F1**           Zeichenfolgen, die (typenabhängig) hinter einer Referenz ergänzt werden sollen, auf die genau eine nächsthöhere folgen würde (vgl. Parameter **RFF**). [ II ]
- F2**           Zeichenfolgen, die (typenabhängig) hinter einer Referenz ergänzt werden sollen, auf die genau zwei nächsthöhere folgen würden (vgl. Parameter **RFF**). [ II ]
- F3**           Zeichenfolgen, die (typenabhängig) hinter einer Referenz ergänzt werden sollen, auf die drei oder mehr nächsthöhere folgen würden (vgl. Parameter **RFF**). [ II ]
- ZRP**        n     Zeichenfolgen, die (typenabhängig) bei der Zusammenfassung von Referenzen zwischen einem Referenzpaar ergänzt werden sollen, wenn die zweite Referenz des Paares mit Referenzteil n beginnt. [ II ] <->

## Positionieren der Referenzteile

Die beiden folgenden Parameter wirken sich nur im Protokoll aus.

- RTP**        n     Position, die nach Referenzteil n erreicht sein soll (ggf. wird mit Leerstellen aufgefüllt). [ I ] <0>

**RTZ**     n     Zusatz zu RTP: In dem durch RTP nach rechts begrenzten Feld soll der Referenzteil n linksbündig (0) bzw. rechtsbündig (1) stehen. [ I ] <0>

### Einfügen einer referenzbezogenen Häufigkeit

Die referenzbezogene Häufigkeit wird nur eingefügt, wenn mit dem Parameter HF ein Zahlenwert ungleich Null angegeben wird.

**HF**            Länge des Feldes, das hinter einer Referenz für die Häufigkeitsangabe freigehalten werden soll, falls das Registerstichwort mehr als einmal mit der gleichen Referenz vorkommt. [ I ] <0>

Benötigt die Häufigkeitsangabe weniger Stellen als angegeben, so wird sie rechtsbündig in das Feld eingespeichert; benötigt sie mehr, so wird das Feld entsprechend vergrößert.

**HFH**          Zeichenfolgen, die (typenabhängig) als Ersatz für die fehlende referenzbezogene Häufigkeit (weil diese 1 ist) eingesetzt werden sollen. [ II ] <>

### Ergänzungen um die referenzbezogene Häufigkeit

**VHF**          Zeichenfolgen, die (typenabhängig) vor der referenzbezogenen Häufigkeit ergänzt werden sollen. [ II ] < (>

**NHF**          Zeichenfolgen, die (typenabhängig) nach der referenzbezogenen Häufigkeit ergänzt werden sollen. [ II ] <>

### Positionieren der referenzbezogenen Häufigkeit

Die beiden folgenden Parameter wirken sich nur im Protokoll aus.

**HFP**          Position, die nach der referenzbezogenen Häufigkeit im Protokoll erreicht sein soll (ggf. wird mit Leerstellen aufgefüllt). [ I ] <0>

**HFZ**          Zusatz zu HFP: In dem durch HFP nach rechts begrenzten Feld soll die referenzbezogene Häufigkeit linksbündig (0) bzw. rechtsbündig (1) stehen. [ I ] <0>

### Prüfen auf gleiches Schlüsselwort

(nur bei MODUS=KWIC)

Jedesmal wenn sich das Schlüsselwort ändert, wird das neue Schlüsselwort zusätzlich in einer eigenen Zeile als Zwischenüberschrift ausgegeben. Mit dem folgenden Parameter können vor dem Prüfen, ob sich das Schlüsselwort geändert hat, Zeichenfolgen im Schlüsselwort ausgetauscht werden.

**XSW** Zeichenfolgenpaare (und Ausnahmezeichenfolgen), deren jeweils erste Zeichenfolge durch die jeweils zweite Zeichenfolge im Schlüsselwort ersetzt werden soll. [ X ]

Dieser Parameter ermöglicht (bei entsprechender Sortierung), dass verschiedene Schreibweisen eines Wortes (z. B. »wort« und »#/+wort#/-«) unter einem Schlüsselwort zusammengefasst werden. Wenn dieser Parameter angegeben ist, werden in den Zwischenüberschriften die Schlüsselwörter in der veränderten Form ausgegeben, in den Zeilen mit dem Kontext werden sie unverändert ausgegeben.

Ist der Parameter XTT ebenfalls angegeben, so werden die Schlüsselwörter für die Zwischenüberschriften mit diesem modifiziert; der Parameter XSW wird in diesem Fall nur zur Prüfung auf Übereinstimmung von aufeinander folgenden Schlüsselwörtern verwendet.

### Ergänzungen um das Schlüsselwort

(nur bei MODUS=KWIC)

Die beiden folgenden Parameter beziehen sich auf das Schlüsselwort, das im Kontext steht.

**VSW** Zeichenfolgen, die (typenabhängig) im Kontext vor dem Schlüsselwort ergänzt werden sollen. [ II ] <>

**NSW** Zeichenfolgen, die (typenabhängig) im Kontext nach dem Schlüsselwort ergänzt werden sollen. [ II ] <>

### Positionieren des Schlüsselwortes

(nur bei MODUS=KWIC)

Die beiden folgenden Parameter wirken sich nur im Protokoll aus. Sie beziehen sich auf das Schlüsselwort, das im Kontext steht.

**SWP** Position, die vor bzw. nach dem Schlüsselwort erreicht sein soll. [ I ] <0>

**SWZ** Zusatz zu SWP: Die mit dem Parameter SWP angegebene Position soll vor (0) bzw. nach (1) dem Schlüsselwort erreicht sein. [ I ] <0>

### Ergänzungen um Referenz/Kontext

(nur bei MODUS=KWIC)

Die beiden folgenden Parameter wirken sich nur in der ZIEL-Datei aus. Mit ihnen kann zwischen dem Satz, der das Schlüsselwort enthält, und dem ersten Satz, der dazu Referenz und Kontext enthält, bzw. nach dem letztem Satz, der zum gleichen Schlüsselwort Referenz und Kontext enthält, jeweils ein eigener Satz eingefügt werden.

- VRK** Zeichenfolgen, die (typenabhängig) vor dem ersten Satz mit Referenz und Kontext ergänzt werden sollen. [ II ] <>
- NRK** Zeichenfolgen, die (typenabhängig) nach dem letzten Satz mit Referenz und Kontext ergänzt werden sollen. [ II ] <>

### Ergänzungen um den Kontext

(nur bei MODUS=KWIC)

- VK** Zeichenfolgen, die (typenabhängig) vor dem Kontext ergänzt werden sollen. [ II ] <>
- NK** Zeichenfolgen, die (typenabhängig) nach dem Kontext ergänzt werden sollen. [ II ] <>

### Einfügen eines lebenden Kolumnentitels

Die Parameter für den lebenden Kolumnentitel werden z. Z. nur bei Ausgabe in die ZIEL-Datei (nicht im Protokoll) berücksichtigt. Die damit zusätzlich in die ZIEL-Datei ausgegebenen Textteile können bei der Weiterverarbeitung mit dem Programm SATZ als lebende Kolumnentitel verwendet werden, wenn sie (mit den Parametern VLK und NLK) entsprechend gekennzeichnet werden.

- LK** n Textteil n des Registereintrags soll in den lebenden Kolumnentitel übernommen werden (1) bzw. nicht übernommen werden (0). [ I ] <0>

### Ergänzungen um den lebenden Kolumnentitel

- VLK** n Zeichenfolgen, die (typenabhängig) vor dem Textteil n im Kolumnentitel ergänzt werden sollen. [ II ]
- NLK** n Zeichenfolgen, die (typenabhängig) nach dem Textteil n im Kolumnentitel ergänzt werden sollen. [ II ]

### Ersetzen im lebenden Kolumnentitel

- XLK** n Ersetzen von Zeichenfolgen in dem Teil des Kolumnentitels, der aus Textteil n übernommen wurde. [ X ]
- LLK** n Zeichenfolgen, die (typenabhängig) als Ersatz für einen leeren Textteil n im Kolumnentitel eingesetzt werden sollen. [ II ] <>



## Zwischenüberschriften bei Anfangsbuchstaben-Wechsel

- AB** n Wechselt der Anfangsbuchstabe des Textteils  $n$ , so soll er eigens (als Großbuchstabe) ausgegeben werden (1) bzw. nicht ausgegeben werden (0). [ I ] <0>
- SAB** n Zeichenfolgen, die am Anfang des Textteils  $n$  beim Suchen des Anfangsbuchstabens übergangen werden sollen. [ IX ]
- Enthält ein Textteil am Anfang z. B. Auszeichnungen wie »#k+« (= Umschaltung auf Kapitälchen), so müssen diese mit dem Parameter SAB angegeben werden. Andernfalls würde z. B. das »k« von »#k+« als Anfangsbuchstabe interpretiert.
- ABD** Zeichenfolgen, die als Anfangsbuchstaben gelten (definiert werden) sollen. [ IX ]
- Wenn der Parameter ABD angegeben wird, muss auch der folgende Parameter ABE angegeben werden.
- Gehören mehrere Zeichenfolgen zum gleichen Anfangsbuchstaben, so muss mit dem Parameter ABE zu diesen Zeichenfolgen jeweils der gleiche Textteil angegeben werden.
- Wird der Parameter ABD nicht angegeben, so sind die Buchstaben a bis z, Umlaute, scharfes s und alle anderen mit »^« codierten Buchstaben Anfangsbuchstaben. Dabei gelten die Umlaute usw. als gleichwertig mit den entsprechenden Grundbuchstaben.
- ABE** Textteile (parallel zu ABD), die statt des Anfangsbuchstaben (als Ersatz für diese) ausgegeben werden sollen. [ II ]
- Wird mit dem Parameter ABE zu mehreren Zeichenfolgen des Parameters ABD der gleiche Textteil angegeben, so gelten die entsprechenden Zeichenfolgen als gleichwertig. Ein solcher Textteil wird ggf. nicht nochmals als neuer Anfangsbuchstabe ausgegeben.
- VAB** n Zeichenfolgen, die (typenabhängig) vor dem wechselnden Anfangsbuchstaben ausgegeben werden sollen. [ II ]
- NAB** n Zeichenfolgen, die (typenabhängig) nach dem wechselnden Anfangsbuchstaben ausgegeben werden sollen. [ II ]

## Begrenzen der Satzlänge

Die folgenden Parameter wirken sich nur bei Ausgabe des Registers in die ZIEL-Datei (nicht im Protokoll) aus.

- SL** Satzlänge der Ausgabesätze. [ I ]
- Falls dieser Parameter nicht angegeben ist, wird eine Zeile (Registereintrag mit Referenzen) nur unterteilt, wenn sie länger als 64000 Zeichen lang ist (vgl. jedoch die Parameter NZB und NZ).

Es können zwei Zahlenwerte angegeben werden:

1. Zahl: Maximallänge für Sätze, in die diejenigen Zeilen unterteilt werden sollen, die länger sind, als mit der zweiten Zahl dieses Parameters angegeben ist. <100>
2. Zahl: Maximallänge für Zeilen, die nicht unterteilt werden sollen. <120 bzw. Wert der 1. Zahl, falls diese größer als 120 ist>

Vor der Ausgabe wird eine Zeile, die mehr Zeichen enthält, als mit dem 2. Zahlenwert angegeben ist, in mehrere Ausgabesätze unterteilt. Die Zeile wird so unterteilt, dass die Ausgabesätze maximal so lang sind, wie mit dem 1. Zahlenwert angegeben ist. Als Trennstelle wird ein Leerzeichen gewählt, vor dem kein »-« steht, das mit einem Silbentrennzeichen verwechselt werden kann. Falls innerhalb der mit dem 1. Zahlenwert angegebenen Anzahl von Zeichen keine solche Trennstelle vorhanden ist, wird die nächstmögliche Trennstelle gesucht.

## Ausgeben eines Endetextes

**zzz** Text, der in die ZIEL-Datei ausgegeben werden soll. Bei jedem Trennzeichen wird ein neuer Satz begonnen. [ II ]

## Alphabetisches Verzeichnis der Parameter

<b>AB</b>	Neuer Anfangsbuchstabe . . . . .	1033
<b>ABD</b>	Anfangsbuchstaben definieren . . . . .	1033
<b>ABE</b>	Ersatzzeichenfolge für Anfangsbuchstaben . . . . .	1033
<b>AH</b>	Stellenzahl für die absolute Häufigkeit . . . . .	1026
<b>AHH</b>	Absolute Häufigkeit: Höchstwert . . . . .	1023
<b>AHM</b>	Absolute Häufigkeit: Mindestwert . . . . .	1023
<b>AHP</b>	Position für absolute Häufigkeit . . . . .	1026
<b>AHW</b>	Wiederholung der absoluten Häufigkeit . . . . .	1026
<b>AHZ</b>	Position für absolute Häufigkeit – zusätzliche Angaben . . . . .	1026
<b>DR</b>	Druckausgabesteuerung . . . . .	1015
<b>DRA</b>	Druckausgabesteuerung bei Anfangsbuchstaben-Wechsel . . . . .	1019
<b>DRE</b>	Druckausgabesteuerung für neuen Eintrag . . . . .	1018
<b>DRT</b>	Druckertyp . . . . .	1018
<b>DRZ</b>	Druckausgabesteuerung – zusätzliche Angaben . . . . .	1016
<b>F</b>	Ergänzung nach einer mit »f« markierten Referenz . . . . .	1029
<b>F1</b>	Ergänzung nach einer Referenz mit 1 Nachfolger . . . . .	1029
<b>F2</b>	Ergänzung nach einer Referenz mit 2 Nachfolgern . . . . .	1029
<b>F3</b>	Ergänzung nach einer Referenz mit 3 oder mehr Nachfolgern . . . . .	1029
<b>FF</b>	Ergänzung nach einer mit »ff« markierten Referenz . . . . .	1029
<b>FT</b>	Fußtext . . . . .	1017
<b>GHF</b>	Gesamthäufigkeit . . . . .	1027
<b>GKU</b>	Groß- und Kleinbuchstaben unterscheiden . . . . .	1024
<b>HF</b>	Stellenzahl für die Referenzhäufigkeit . . . . .	1030

<b>HFE</b>	Ersatzzeichenfolge für Referenzhäufigkeit 1 . . . . .	1030
<b>HFL</b>	Häufigkeitsangabe: Länge . . . . .	1019
<b>HFP</b>	Position für Referenzhäufigkeit . . . . .	1030
<b>HFZ</b>	Position für Referenzhäufigkeit – zusätzliche Angaben . . . . .	1030
<b>IRL</b>	Interne Referenz: Länge . . . . .	1019
<b>ITT</b>	Ersatzzeichenfolge für einen identischen Textteil . . . . .	1024
<b>KT</b>	Kopftext . . . . .	1016
<b>KTZ</b>	Kopftext – zusätzliche Angaben . . . . .	1017
<b>LK</b>	Lebender Kolumnentitel . . . . .	1032
<b>LLK</b>	Ersatzzeichenfolge für einen leeren Kolumnentitel . . . . .	1032
<b>LN</b>	Stellenzahl für die laufende Nummer . . . . .	1025
<b>LNB</b>	Basiswert für die laufende Nummer . . . . .	1025
<b>LNN</b>	Neubeginn der laufenden Nummer . . . . .	1025
<b>LNP</b>	Positionen für laufende Nummer . . . . .	1025
<b>LNW</b>	Wiederholung der laufenden Nummer . . . . .	1025
<b>LNZ</b>	Positionen für laufende Nummer – zusätzliche Angaben . . . . .	1025
<b>LRT</b>	Ersatzzeichenfolge für einen leeren Referenzteil . . . . .	1028
<b>LTT</b>	Ersatzzeichenfolge für einen leeren Textteil . . . . .	1024
<b>MAX</b>	Maximum für Testzwecke . . . . .	1019
<b>MRF</b>	Markierte Referenzen ausgeben . . . . .	1028
<b>NAB</b>	Ergänzung nach einem neuem Anfangsbuchstaben . . . . .	1033
<b>NAH</b>	Ergänzung nach der absoluten Häufigkeit . . . . .	1026
<b>NHF</b>	Ergänzung nach der Referenzhäufigkeit . . . . .	1030
<b>NK</b>	Ergänzung nach dem Kontext . . . . .	1032
<b>NLK</b>	Ergänzung nach dem lebendem Kolumnentitel . . . . .	1032
<b>NLN</b>	Ergänzung nach der laufenden Nummer . . . . .	1025
<b>NR</b>	Nummerierung der Ausgabesätze . . . . .	1018
<b>NRF</b>	Ergänzung nach den Referenzen . . . . .	1027
<b>NRH</b>	Ergänzung nach der relativen Häufigkeit . . . . .	1027
<b>NRK</b>	Ergänzung nach Referenz/Kontext . . . . .	1032
<b>NRT</b>	Ergänzung nach Referenzteilen . . . . .	1029
<b>NSW</b>	Ergänzung nach dem Schlüsselwort . . . . .	1031
<b>NTT</b>	Ergänzung nach Textteilen . . . . .	1024
<b>NZ</b>	Neue Zeile nach einem Textteil . . . . .	1021
<b>NZB</b>	Neue Zeile bis zu einem bestimmten Textteil . . . . .	1021
<b>NZU</b>	Neue Zeile unterdrücken . . . . .	1021
<b>PAR</b>	Parameter-Konvention . . . . .	1014
<b>REF</b>	Referenzen mit best. Typen ausgeben / nicht ausgeben . . . . .	1020
<b>RFF</b>	Aufeinander folgende Referenzen zusammenfassen . . . . .	1028
<b>RFL</b>	Länge der einzelnen Referenzteile . . . . .	1028
<b>RFU</b>	Referenzen ab einer best. Häufigkeit unterdrücken . . . . .	1028
<b>RH</b>	Gesamtstellen für relative Häufigkeit . . . . .	1026
<b>RHB</b>	Bezug für relative Häufigkeit . . . . .	1026
<b>RHD</b>	Dezimalstellen für relative Häufigkeit . . . . .	1026
<b>RHP</b>	Position für relative Häufigkeit . . . . .	1027
<b>RHW</b>	Wiederholung der relativen Häufigkeit . . . . .	1027
<b>RHZ</b>	Position für relative Häufigkeit – zusätzliche Angaben . . . . .	1027
<b>ROM</b>	Römische Seitenzahlen . . . . .	1018

---

<b>RTE</b>	Ersatzzeichenfolge für einen Referenzteil . . . . .	1028
<b>RTP</b>	Position für Referenzteil . . . . .	1029
<b>RTU</b>	Referenzteile unterdrücken . . . . .	1028
<b>RTW</b>	Wiederholung von Referenzteilen . . . . .	1028
<b>RTZ</b>	Position für Referenzteil – zusätzliche Angaben . . . . .	1030
<b>SAB</b>	Suchen des Anfangsbuchstabens . . . . .	1033
<b>SL</b>	Satzlänge der Ausgabesätze . . . . .	1033
<b>SNL</b>	Sortiernummer: Länge . . . . .	1019
<b>SSL</b>	Länge der Sortierschlüssel . . . . .	1019
<b>SWP</b>	Position für Schlüsselwort . . . . .	1031
<b>SWZ</b>	Position für Schlüsselwort – zusätzliche Angaben . . . . .	1031
<b>T+</b>	Positiv-Auswahl über ganze Textteile . . . . .	1022
<b>T+N</b>	Positiv-Auswahl über ganze Textteile . . . . .	1022
<b>T+U</b>	Positiv-Auswahl über ganze Textteile . . . . .	1022
<b>T-</b>	Negativ-Auswahl über ganze Textteile . . . . .	1022
<b>T-N</b>	Negativ-Auswahl über ganze Textteile . . . . .	1022
<b>T-U</b>	Negativ-Auswahl über ganze Textteile . . . . .	1022
<b>TA+</b>	Positiv-Auswahl über den Anfang von Textteilen . . . . .	1023
<b>TA-</b>	Negativ-Auswahl über den Anfang von Textteilen . . . . .	1023
<b>TE+</b>	Positiv-Auswahl über das Ende von Textteilen . . . . .	1023
<b>TE-</b>	Negativ-Auswahl über das Ende von Textteilen . . . . .	1023
<b>TFR</b>	Umdefinieren der Typen für Referenzteile . . . . .	1020
<b>TFT</b>	Umdefinieren der Typen für Textteile . . . . .	1020
<b>TR</b>	Trennzeichen zwischen den Textteilen . . . . .	1021
<b>TRB</b>	Trennzeichen gilt bis zu einer bestimmten Stelle . . . . .	1021
<b>TRN</b>	Trennzeichennummer . . . . .	1021
<b>TRT</b>	Austauschen von Referenzteilen . . . . .	1028
<b>TRU</b>	Trennzeichen unterdrücken . . . . .	1021
<b>TRV</b>	Trennzeichen gilt von einer bestimmten Stelle an . . . . .	1021
<b>TT</b>	Anzahl der Textteile eines Registereintrags . . . . .	1021
<b>TTE</b>	Ersatzzeichenfolge für einen Textteil . . . . .	1024
<b>TTP</b>	Positionen für Textteile . . . . .	1025
<b>TTT</b>	Austauschen von Textteilen . . . . .	1024
<b>TTW</b>	Wiederholung von Textteilen . . . . .	1024
<b>TTZ</b>	Positionen für Textteile – zusätzliche Angaben . . . . .	1025
<b>TXT</b>	Auswahl der Einträge über Typen . . . . .	1020
<b>TYP</b>	Typ, der statt Typ 0 verwendet werden soll . . . . .	1020
<b>VAB</b>	Ergänzung vor einem neuem Anfangsbuchstaben . . . . .	1033
<b>VAH</b>	Ergänzung vor der absoluten Häufigkeit . . . . .	1026
<b>VHF</b>	Ergänzung vor der Referenzhäufigkeit . . . . .	1030
<b>VK</b>	Ergänzung vor dem Kontext . . . . .	1032
<b>VLK</b>	Ergänzung vor dem lebendem Kolummentitel . . . . .	1032
<b>VLN</b>	Ergänzung vor der laufenden Nummer . . . . .	1025
<b>VRF</b>	Ergänzung vor den Referenzen . . . . .	1027
<b>VRH</b>	Ergänzung vor der relativen Häufigkeit . . . . .	1027
<b>VRK</b>	Ergänzung vor Referenz/Kontext . . . . .	1032
<b>VRT</b>	Ergänzung vor Referenzteilen . . . . .	1029
<b>VSW</b>	Ergänzung vor dem Schlüsselwort . . . . .	1031

---

<b>VTT</b>	Ergänzung vor Textteilen . . . . .	1024
<b>XLK</b>	Ersetzen von Zeichenflg. im lebenden Kolumnentitel . . . . .	1032
<b>XRT</b>	Ersetzen von Zeichenfolgen in Referenzteilen . . . . .	1028
<b>XSW</b>	Ersetzen von Zeichenfolgen im Schlüsselwort . . . . .	1031
<b>XTT</b>	Ersetzen von Zeichenfolgen in Textteilen . . . . .	1024
<b>Z</b>	Ausgeben eines Anfangstextes . . . . .	1015
<b>ZF+</b>	Positiv-Auswahl über enthaltene Zeichenfolgen . . . . .	1023
<b>ZF-</b>	Negativ-Auswahl über enthaltene Zeichenfolgen . . . . .	1023
<b>ZRF</b>	Ergänzung zwischen den Referenzen . . . . .	1027
<b>ZRP</b>	Ergänzung zwischen Referenzpaaren . . . . .	1029
<b>ZZZ</b>	Ausgeben eines Endetextes . . . . .	1034

\*\*\*\*\*



**#RVORBEREITE**

---

Kommando . . . . .	1041
Leistung . . . . .	1042
Beispiel . . . . .	1042
Allgemeines . . . . .	1043
Parameter . . . . .	1045
Einstellen der Parameter-Konvention . . . . .	1045
Auswählen der Daten . . . . .	1045
Zusammenfassen mehrerer Sätze zu einer Texteinheit . . . . .	1046
Ersetzen von Zeichenfolgen bei der Eingabe . . . . .	1049
Auswählen der Textteile, die Registereinträge enthalten . . . . .	1049
Definieren der Registereinträge . . . . .	1050
Modifizieren der Registereinträge . . . . .	1051
Bestimmen des Typs der einzelnen Registereinträge . . . . .	1051
Bestimmen der Schlüsselwörter . . . . .	1052
Definieren der Referenz . . . . .	1053
Markieren der Referenz . . . . .	1057
Häufigkeitsangaben . . . . .	1059
Umdrehen der Registereinträge . . . . .	1060
Auswählen der Registereinträge bzw. Schlüsselwörter . . . . .	1061
Überprüfen der Länge der Registereinträge . . . . .	1063
Ergänzen der Registereinträge . . . . .	1063
Gruppenweise Sortieren . . . . .	1065
Bestimmen der ZIEL-Datei . . . . .	1065
Erstellen der Sortierschlüssel . . . . .	1066
Alphabetisches Verzeichnis der Parameter . . . . .	1069
Weiterverarbeitung der Daten . . . . .	1072
Aufbau eines Datensatzes . . . . .	1072



## Kommando

#RVORBEREITE

QUELLE	= <code>datei</code>	Name der Datei mit den Daten, aus denen Register- einträge extrahiert werden sollen bzw. von denen ein KWIC-Index vorbereitet werden soll.
	= <code>-STD-</code>	Die Daten, aus denen Register- einträge extrahiert werden sollen bzw. von denen ein KWIC-Index vor- bereitet werden soll, stehen in der Standard-Text- Datei.
ZIEL	= <code>datei</code>	Name der Datei für die Register- einträge bzw. für die Einträge für den KWIC-Index.
	= <code>-STD-</code>	Die Register- einträge bzw. die Einträge für den KWIC- Index in die Standard-Text-Datei ausgeben.
MODUS	= <code>+</code>	* Register- einträge mit Referenz erstellen.
	= <code>-</code>	Register- einträge ohne Referenz erstellen.
	= <code>KWIC</code>	Einträge für KWIC-Index erstellen.
LOESCHEN	= <code>-</code>	* Daten in der ZIEL-, in der DATEN- und in der PROTOKOLL-Datei nicht löschen.
	= <code>+</code>	Daten in der ZIEL-, in der DATEN- und in der PROTOKOLL-Datei zuerst löschen.
PARAMETER	= <code>datei</code>	Name der Datei mit den Parametern.
	= <code>*</code>	Die Parameter folgen auf das Kommando und sind mit *EOF abgeschlossen.
DATEN	= <code>-</code>	* Daten vollständig in die ZIEL-Datei ausgeben.
	= <code>datei</code>	Name der Datei für die zum Sortieren nicht notwen- digen Datenteile.
	= <code>-STD-</code>	Die zum Sortieren nicht notwendigen Datenteile in die Standard-Daten-Datei ausgeben.
PROTOKOLL	= <code>-</code>	* Kein Testprotokoll erstellen.
	= <code>+</code>	Testprotokoll ins Ablaufprotokoll ausgeben.
	= <code>-STD-</code>	Testprotokoll in die Standard-Protokoll-Datei ausge- ben.
	= <code>datei</code>	Name der Datei für das Testprotokoll.

## Leistung

Mit diesem Programm können

- Texte in sortierfähige und für Register geeignete Einträge zerlegt werden (z. B. für ein Wortformenregister oder einen KWIC-Index),
- aus Texten entsprechend gekennzeichnete Textteile als Registerinträge übernommen werden (z. B. für ein Autorenregister, Personenregister, Sachregister oder ein Ortsregister).

## Beispiel

Erstellen und Ausdrucken eines Registers aller kursiv geschriebenen Wörter. Der Text, der einzelne kursiv geschriebene Wörter enthält, steht in der Datei `text`. Ausgedruckt werden soll auf einem PostScript-Drucker, der den vom System vorgegebenen Druckernamen `ps1` hat:

```
#RV, text, -STD-, +, +, *
STR      1
EA       :#/+ :
EE       :#/- :
XS1      /ä/ae/ö/oe/ü/ue/ß/ss/%{1--2}{%} //
XS2      /ä/az/ö/oz/ü/uz/ß/sz/
A2       {|}%
SSL      20 20
*EOF
```

```
#SO, -STD-, -STD-, 17+40, +
```

```
#RA, -STD-, -, +, +, *
DR       2 10 32 4
DRT      PS-10
SSL      20 20
RFF      1
*EOF
```

```
#DR, , PS-10, ps1
```

Ein weiteres Beispiel mit dem Kommando #RVORBEREITE ist in der Beschreibung des Kommandos #RAUFBEREITE angegeben.

## Allgemeines

Registereinträge werden durch Zerlegen des Eingabetextes anhand von Trennzeichenfolgen und/oder durch Auswählen von Textteilen, die zwischen Anfangs- und Endkennungen stehen, gewonnen.

Registereinträge können aus mehreren Teilen zusammengesetzt werden. Zu diesem Zweck kann der Registerbeitrag durch weitere Zeichenfolgen aus dem Eingabetext ergänzt werden.

Aus dem Text des Registerbeitrags werden ein bis drei Sortierschlüssel aufgebaut. Dies ist deshalb notwendig, weil die Reihenfolge der Zeichen im internen Code des Rechners nicht der üblichen alphabetischen Ordnung entspricht. Die Sortierschlüssel werden nur zum Sortieren benutzt und werden danach wieder eliminiert. Für englische oder lateinische Texte, die weder Umlaute noch Akzente enthalten, reicht 1 Sortierschlüssel aus. In deutschen Texten muss nach DIN 5007 das ß für die Sortierung in ss umgewandelt werden; die Umlaute werden in einigen Wörterbüchern, Lexika etc. wie die entsprechenden Grundbuchstaben eingeordnet, in Namensverzeichnissen (Telefonbücher, Registraturen etc.) wie ae, oe, ue. Diese Gleichsetzungen (Umwandlungen) müssen im 1. Sortierschlüssel vorgenommen werden. Damit dabei Wörter wie »Maße« und »Masse«, »drucken« und »drücken« oder (in Namensverzeichnissen) Namen wie »Jäger« und »Jaeger« auseinander gehalten werden können, wird ein 2. Sortierschlüssel benötigt. Für ihn können ä, ö, ü und ß z. B. in az, oz, uz und sz umcodiert werden. Enthält der Text Akzente, so wird ebenfalls ein 2. Sortierschlüssel benötigt. Üblicherweise werden die Akzente für den 1. Sortierschlüssel eliminiert, damit zuerst nach dem Buchstabenbestand sortiert wird und die Akzente nur bei gleichem Buchstabenbestand die Reihenfolge beeinflussen. Für den 2. Sortierschlüssel können die Akzente z. B. in Dezimalzahlen umcodiert werden, damit sich für Wörter, die sich nur durch die Akzente unterscheiden, die gewünschte Reihenfolge ergibt. Wenn darüber hinaus noch Groß- und Kleinschreibung beim Sortieren von Bedeutung sind, wird ein dritter Sortierschlüssel benötigt. Falls keine Akzente und keine Umlaute vorkommen, kann die Groß- und Kleinschreibung im zweiten Sortierschlüssel berücksichtigt werden.

Falls die Registerbeiträge schon in der Eingabedatei in Gruppen unterteilt sind, ist es möglich, die Registerbeiträge jeweils nur innerhalb ihrer Gruppe zu sortieren. Dazu muss am Anfang jeder Gruppe eine eindeutige Kennung stehen. Es kann z. B. jeweils eine Zwischenüberschrift entsprechend gekennzeichnet sein. Die Gruppen werden intern durchnummeriert; die entsprechende Nummer wird jeweils unmittelbar vor dem Sortierschlüssel der einzelnen Registerbeiträge eingesetzt. Die Reihenfolge der Gruppen bleibt auf diese Weise unverändert.

Registereinträge können verschiedenen Typen zugeordnet werden, anhand derer sie später bei der Register-Aufbereitung auf vielfältige Weise (z. B. typographisch) differenziert werden können. Zu diesem Zweck wird jedem Registerbeitrag eine Typenkennnummer zugeordnet, deren Wert anhand von im Eingabetext enthaltenen Typenkennzeichen festgelegt werden kann. Dieses Typenkennzeichen kann im Eingabetext vor jedem einzelnen Registerbeitrag stehen oder als pauschales Typenkennzeichen angegeben werden. In diesem Fall gilt es jeweils für alle nachfolgenden Registerbeiträge.

Sollen im zu erstellenden Register die Registereinträge durch eine Referenz ergänzt werden, die die Fundstelle im zugrunde liegenden Text angibt, so muss jedem Registereintrag eine solche Referenz mitgegeben werden. Diese Referenz kann, wenn sie auf die Seiten- und ggf. Zeilen- und Wortnummer des durch das Register erschlossenen Textes verweisen soll, aus der Satznummer des Eingabetextes entnommen werden. Außerdem können entsprechend gekennzeichnete Textteile als Referenz übernommen werden. Dabei kann die Referenz auch aus mehreren Teilen zusammengesetzt werden.

Das Kommando #SORTIERE benötigt Speicherplatz zum Zwischenspeichern der zu sortierenden Daten. Bei sehr großen Datenmengen reicht der zur Verfügung stehende Platz unter Umständen nicht aus. Abhilfe kann dadurch geschaffen werden, dass die Sortierschlüssel und die eigentlichen Registereinträge auf verschiedene Dateien (ZIEL-Datei und DATEN-Datei) ausgegeben werden. Es genügt dann, die Datei mit den Sortierschlüsseln zu sortieren. Sollte der Speicherplatz zum Sortieren trotzdem nicht ausreichen, können die Sortierschlüssel auf mehrere Dateien verteilt werden. Diese müssen einzeln sortiert und dann zusammengemischt werden. Damit die Sortierschlüssel auf mehrere Dateien verteilt werden, genügt die Angabe mehrerer ZIEL-Dateien, zusätzlich Parameter sind dafür nicht erforderlich.

Bei MODUS=KWIC muss zur Spezifikation DATEN grundsätzlich -STD- bzw. der Name einer Datei angegeben werden. In die ZIEL-Datei werden bei diesem Modus die zum Sortieren notwendigen Datenteile und Zeiger (Pointer) auf das Schlüsselwort im Kontext ausgegeben, in die DATEN-Datei der dazugehörige Kontext.

## Parameter

Jeder Parameter ist einer Parameterart zugeordnet. Sie ist jeweils als römische Zahl in [ ] angegeben. Damit ist festgelegt, in welcher Form die Angaben vom Programm erwartet werden. Die einzelnen Parameterarten sind in der Beschreibung »TUSTEP-Grundlagen« im Kapitel »{}-Parameter« bzw. »<>-Parameter« definiert.

Für manche Parameter gibt es voreingestellte Werte, die angenommen werden, falls die entsprechenden Werte nicht angegeben sind. Diese Voreinstellungen sind in < > angegeben.

Mit bestimmten Parametern können über Anfangs- und/oder Endekennungen sowie über Kennungen, die als öffnende bzw. schließende Klammern dienen, Textteile für die weitere Bearbeitung ausgewählt werden. Die Funktionsweise dieser Parameter ist auch in den »TUSTEP-Grundlagen« am Ende des Kapitels »Parameter« beschrieben.

Außer den im folgenden beschriebenen Parametern sind auch Parameter zur Definition von Zeichen- und Stringgruppen möglich. [ V ]

Außer dem Parameter `SSL` muss bei `MODUS=KWIC` noch mindestens der Parameter `SWT` und bei allen anderen Modi noch mindestens einer der Parameter `EA`, `EE` oder `TR` angegeben werden.

### Einstellen der Parameter-Konvention

Ob die Parameter nach der »{}-Parameter-Konvention« oder nach der »<>-Parameter-Konvention« interpretiert werden, kann mit dem Kommando `#PARAMETER, {}` bzw. `#PARAMETER, <>` eingestellt werden. Darüber hinaus kann die Konvention auch mit folgendem Parameter eingestellt werden.

**PAR**            Parameter-Konvention einstellen.

    {}    Nachfolgende Parameter nach der »{}-Parameter-Konvention« interpretieren.

    <>    Nachfolgende Parameter nach der »<>-Parameter-Konvention« interpretieren.

Die Einstellung mit dem Parameter `PAR` hat Vorrang vor der mit dem Kommando `#PARAMETER` gewählten Einstellung; sie gilt jeweils nur für die nachfolgenden Parameter bis zum nächsten Parameter `PAR` bzw. bis zum Ende der Parameter dieses Programms.

### Auswählen der Daten

Soll die gesamte Datei bearbeitet werden, braucht keiner der folgenden Parameter angegeben zu werden.

**BER**            Angabe eines Bereichs (»Seite.Zeile–Seite.Zeile«, falls die Datei im Textmodus nummeriert ist; Zeile–Zeile, falls die Datei im Programmmodus nummeriert ist) oder einer Anfangsstelle (»Seite.Zeile« bzw. »Zeile«), falls nicht die ganze Datei verarbeitet werden soll. [ XI ]

Soll ein Segment einer Segment-Datei verarbeitet werden, kann anstelle des Bereichs der Name des Segments angegeben werden.

Dieser Parameter ist nur zugelassen, wenn die Satznummern in der Datei alle aufsteigend sind.

**MAX** Angabe für Testzwecke, aus wievielen Texteinheiten (= Eingabesätze, falls keiner der Parameter ANR, ALZ, AA oder AE angegeben ist) maximal Registereinträge erstellt werden sollen bzw. wieviele Registereinträge maximal erstellt werden sollen. [ 1 ]

Es können zwei Zahlenwerte angegeben werden:

1. Zahl einzulesende Texteinheiten <999999999>
2. Zahl auszugebende Registereinträge <999999999>

### Zusammenfassen mehrerer Sätze zu einer Texteinheit

Falls jeder Registereintrag und ggf. jeder als Referenz zu übernehmende Textteil vollständig in einem Eingabesatz enthalten ist, sollte keiner der folgenden Parameter angegeben werden. In diesem Fall bildet jeder Eingabesatz eine Texteinheit. Andernfalls können mit den Parametern ANR, ALZ, AA und/oder AE jeweils mehrere Sätze zu einer Texteinheit zusammengefasst werden.

Ist einer der Parameter ANR, ALZ, AA und AE angegeben, werden vor der Auswertung dieser Parameter evtl. am Anfang und am Ende des Eingabesatzes stehende Leerzeichen entfernt.

Beim Zusammenfassen wird zwischen den einzelnen Eingabesätzen je ein Leerzeichen ergänzt, jedoch nicht an den Stellen, an denen eine Silbentrennung aufgehoben wird (siehe Parameter STR).

**ANR** Angaben, ob aufeinander folgende Sätze, deren Satznummern teilweise oder vollständig übereinstimmen, zu einer Texteinheit zusammengefasst werden sollen. (Abschnitt auf Grund der Nummerierung). [ 1 ] <0>

Es kann ein Zahlenwert angegeben werden:

- 0 = kein Zusammenfassen von Sätzen auf Grund der Satznummer
- 1 = alle aufeinander folgenden Sätze mit der gleichen Seitennummer zu einer Texteinheit zusammenfassen.
- 2 = alle aufeinander folgenden Sätze mit der gleichen Seiten- und der gleichen Zeilennummer (ohne Berücksichtigung der Unterscheidungsnummer) zu einer Texteinheit zusammenfassen.
- 3 = alle aufeinander folgenden Sätze mit der gleichen Satznummer zu einer Texteinheit zusammenfassen.

Ist 0 angegeben (Voreinstellung), so wird die Zusammenfassung nur auf Grund der Parameter ALZ, AA und AE vorgenommen; ist einer der Werte 1 bis 3 angegeben, so ist eine weitere Unterteilung der so gebildeten Texteinheiten auf Grund der genannten Parameter möglich.

**ALZ** Angaben, ob Leerzeilen als Kennzeichen für den Anfang bzw. für das Ende einer Texteinheit dienen sollen. [ I ]

Es können zwei Zahlenwerte angegeben werden:

1. Zahl: Anfang einer Texteinheit <0>

0 = Leerzeilen sind kein Kennzeichen für den Anfang einer Texteinheit.

1 = Leerzeilen kennzeichnen den Anfang einer Texteinheit. Folgen mehrere Leerzeilen unmittelbar aufeinander, so beginnt mit jeder einzelnen Leerzeile eine Texteinheit.

2. Zahl: Ende einer Texteinheit <0>

0 = Leerzeilen sind kein Kennzeichen für das Ende einer Texteinheit.

1 = Leerzeilen kennzeichnen das Ende einer Texteinheit. Folgen mehrere Leerzeilen unmittelbar aufeinander, so endet mit jeder einzelnen Leerzeile eine Texteinheit.

Ist jeweils 0 angegeben (Voreinstellung), so wird die Zusammenfassung nur auf Grund der Parameter ANR, AA und AE vorgenommen; andernfalls ist eine weitere Unterteilung der so gebildeten Texteinheiten auf Grund der genannten Parameter möglich.

**AA** Zeichenfolgen, die am Anfang des Eingabesatzes den Anfang einer Texteinheit (Abschnittsanfang) kennzeichnen. [ VIII-a ]

Falls mit dem Parameter AAZ nichts anderes angegeben ist, werden vor der Überprüfung, ob ein Satz mit einer der angegebenen Zeichenfolgen beginnt, führende Leerzeichen entfernt.

Beginnen mit dem Parameter AA angegebene Zeichenfolgen mit Leerzeichen, so können diese nur dann einen Abschnittsanfang kennzeichnen, wenn mit dem Parameter AAZ angegeben wird, dass die führenden Leerzeichen erst nach der Überprüfung entfernt werden sollen; andernfalls ist die Angabe solcher Zeichenfolgen wirkungslos.

**AAZ** Zusatzangaben zum Parameter AA. [ I ] <0>

Es kann ein Zahlenwert angegeben werden:

0 = Leerzeichen am Anfang von Eingabesätzen vor der Auswertung des Parameters AA entfernen.

1 = Leerzeichen am Anfang von Eingabesätzen nach der Auswertung des Parameters AA entfernen.

2 = wie 0, jedoch zusätzlich auch die Zeichenfolgen »#[09]« und »#[0009]« entfernen.

**AE** Zeichenfolgen, die am Ende des Eingabesatzes das Ende einer Texteinheit (Abschnittsende) kennzeichnen. [ VIII-b ]

Falls mit dem Parameter AEZ nichts anderes angegeben ist, werden vor der Überprüfung, ob ein Satz mit einer der angegebenen Zeichenfolgen endet, abschließende Leerzeichen entfernt.

Enden mit dem Parameter **AE** angegebene Zeichenfolgen mit Leerzeichen, so können diese nur dann ein Abschnittsende kennzeichnen, wenn mit dem Parameter **AEZ** angegeben wird, dass die abschließenden Leerzeichen erst nach der Überprüfung entfernt werden sollen; andernfalls ist die Angabe solcher Zeichenfolgen wirkungslos.

**AEZ** Zusatzangaben zum Parameter **AE**. [ I ] <0>

Es kann ein Zahlenwert angegeben werden:

- 0 = Leerzeichen am Ende von Eingabesätzen vor der Auswertung des Parameters **AE** entfernen.
- 1 = Leerzeichen am Ende von Eingabesätzen nach der Auswertung des Parameters **AE** entfernen.

**STR** Angabe zur Silbentrennung. [ I ] <0>

Es kann ein Zahlenwert angegeben werden:

- 0 = Silbentrennungen nicht aufheben.
- 1 = Silbentrennung bei der Eingabe aufheben; dabei wird angenommen, dass im nachfolgenden Eingabesatz alle Zeichen bis zum ersten Leerzeichen zum getrennten Wort gehören.
- 2 = wie 1, jedoch Silbentrennung bei der Eingabe nur aufheben, falls entweder der letzte Buchstabe vor und der erste Buchstabe nach dem Silbentrennzeichen (im nachfolgenden Eingabesatz) Kleinbuchstaben sind oder der letzte Buchstabe vor und die ersten beiden Buchstaben nach dem Silbentrennzeichen (im nachfolgenden Eingabesatz) Großbuchstaben sind.
- 3 = wie 2, jedoch wird angenommen, dass im nachfolgenden Eingabesatz alle Zeichen bis zum ersten Leerzeichen bzw. bis zur ersten öffnenden spitzen Klammer zum getrennten Wort gehören.
- 4 = wie 2, jedoch wird angenommen, dass im nachfolgenden Eingabesatz alle Zeichen bis zum ersten Leerzeichen, das außerhalb von spitzen Klammern steht, zum getrennten Wort gehören.

Falls angegeben ist, dass die Silbentrennung bei der Eingabe aufgehoben werden soll, werden in allen Eingabesätzen führende und abschließende Leerzeichen entfernt.

Sollen bei der Eingabe Zeichenfolgen ersetzt werden (Parameter **x**), so wird nach dem Ersetzen festgestellt, ob der Satz mit einem Silbentrennzeichen endet.

Als Silbentrennzeichen gilt ein »-«, das (nachdem abschließende Leerzeichen entfernt wurden) als letztes Zeichen in einem Eingabesatz steht, wenn das zweitletzte Zeichen ebenfalls ein »-« ist, oder wenn das zweitletzte Zeichen ein Buchstabe und das drittletzte Zeichen kein Steuerzeichen (\$, &, @, \, \_, #, %) ist.

Beim Zusammenfassen mehrerer Eingabesätze zu einer Texteinheit werden Silbentrennungen nur innerhalb einer Texteinheit aufgehoben.



ben. Steht ein Silbentrennzeichen am Ende des letzten Eingabesatzes einer Texteinheit, wird an dieser Stelle die Silbentrennung nicht aufgehoben.

Beim Aufheben der Silbentrennung wird ein getrenntes »ck«, das als »k-« und »k« geschrieben ist, nicht wieder zu »ck«.

**STE** Zeichenfolge, die beim Aufheben einer Silbentrennung anstelle des Silbentrennzeichens eingefügt werden soll. [ II ]

## Ersetzen von Zeichenfolgen bei der Eingabe

**X** Zeichenfolgenpaare (und Ausnahmezeichenfolgen), deren jeweils erste Zeichenfolge durch die jeweils zweite Zeichenfolge bei der Eingabe ersetzt werden soll. [ X ]

Das Ersetzen erfolgt in jedem einzelnen Eingabesatz, auch wenn mehrere Sätze zu einer Texteinheit zusammengefasst werden (vgl. die Parameter ANR, AA und AE).

Vor dem Ersetzen werden evtl. am Anfang und am Ende des Satzes stehende Leerzeichen entfernt und dann am Anfang und am Ende je ein Leerzeichen ergänzt. Nach dem Ersetzen werden nochmals evtl. am Anfang und am Ende des Satzes stehende Leerzeichen entfernt.

Die Überprüfung, ob ein Satz mit einer zum Parameter AA angegebenen Zeichenfolge beginnt bzw. mit einer zum Parameter AE angegebenen endet, erfolgt vor dem Ersetzen.

Soll die Silbentrennung aufgehoben werden (Parameter STR), so wird nach dem Ersetzen festgestellt, ob der Satz mit einem Silbentrennzeichen endet.

## Auswählen der Textteile, die Registereinträge enthalten

Kommen auszugebende Registereinträge jeweils in der gesamten Texteinheit vor, so braucht keiner der folgenden Parameter angegeben zu werden.

**A** Zeichenfolgen, die den Anfang der Textteile kennzeichnen, die auszugebende Registereinträge enthalten. [ IX ]

**E** Zeichenfolgen, die das Ende der Textteile kennzeichnen, die auszugebende Registereinträge enthalten. [ IX ]

Der ausgewählte Textteil beginnt jeweils nach der Anfangskennung und endet vor der Endekennung bzw. am Ende der Texteinheit, falls keine Endekennung gefunden wird. Ist jedoch der Parameter EA angegeben, so gehört die Anfangskennung zum ausgewählten Textteil.

Sind beide Parameter A und E angegeben, so werden nacheinander alle mit A/E gekennzeichneten Textteile ausgewählt. Der zweite beginnt nach bzw. mit der Anfangskennung, die dem Ende des ersten folgt.

Dabei kann die Anfangskennung mit der Endekennung des vorangehenden Textteils zusammenfallen.

Ist nur der Parameter **A** angegeben, so endet der ausgewählte Textteil am Ende der Texteinheit; ist nur der Parameter **E** angegeben, so beginnt der ausgewählte Textteil am Anfang der Texteinheit.

## Definieren der Registereinträge

Registereinträge können auf zwei Arten definiert werden: Entweder durch die Angabe von Anfangs- und/oder Endekennungen, mit denen sie gekennzeichnet sind, mit den Parametern **EA** und **EE** oder durch die Angabe von Trennzeichenfolgen, die zwischen den Registereinträgen stehen, mit dem Parameter **TR**.

Wird der Parameter **TR** zusätzlich zu den Parametern **EA** und/oder **EE** angegeben, so werden die mit **EE/EA** ausgewählten Textteile anhand der mit dem Parameter **TR** angegebenen Trennzeichenfolgen in einzelne Registereinträge aufgeteilt.

Nach der Definition der Registereinträge durch die Parameter **EA**, **EE** und/oder **TR** werden führende und abschließende Leerzeichen von den Registereinträgen entfernt.

**EA** Zeichenfolgen, die den Anfang eines Registereintrags kennzeichnen.  
[ IX ]

**EE** Zeichenfolgen, die das Ende eines Registereintrags kennzeichnen.  
[ IX ]

Der ausgewählte Registereintrag beginnt jeweils nach der Anfangskennung und endet vor der Endekennung bzw. am Ende der Texteinheit (bzw. des mit **A/E** ausgewählten Textteils), falls keine Endekennung gefunden wird.

Sind beide Parameter **EA** und **EE** angegeben, so werden alle mit **EA/EE** gekennzeichneten Textteile als Registereinträge definiert. Der zweite beginnt nach der Anfangskennung, die dem Ende des ersten folgt. Dabei kann die Anfangskennung mit der Endekennung des vorangehenden Textteils zusammenfallen.

Ist nur der Parameter **EA** angegeben, so endet der ausgewählte Textteil, der als Registereintrag definiert wird, am Ende der Texteinheit; ist nur der Parameter **EE** angegeben, so beginnt der ausgewählte Textteil, der als Registereintrag definiert wird, am Anfang der Texteinheit.

**TR** Zeichenfolgen, die als Trennzeichen zwischen den einzelnen Registereinträgen stehen. [ IX ]

## Modifizieren der Registereinträge

(nicht bei `MODUS=KWIC`)

Die folgenden Parameter sind nur anzugeben, falls die Registereinträge nicht in unveränderter Form übernommen werden können.

**((** Zeichenfolgen, die beim Eliminieren von Teilen der Registereinträge als öffnende Klammer dienen sollen. [ IX ]

**)**) Zeichenfolgen, die beim Eliminieren von Teilen der Registereinträge als schließende Klammer dienen sollen. [ IX ]

Es werden die Teile (einschließlich der Klammern) eliminiert, die eingeklammert sind. Fehlende Klammern werden am Anfang bzw. am Ende der Registereinträge logisch ergänzt.

**xx** Zeichenfolgenpaare (und Ausnahmezeichenfolgen), deren jeweils erste Zeichenfolge durch die jeweils zweite Zeichenfolge ersetzt werden soll. [ X ]

Das Ersetzen erfolgt in jedem Registereintrag, nachdem ggf. Teile des Registereintrags durch die zuvor genannten Parameter eliminiert worden sind.

## Bestimmen des Typs der einzelnen Registereinträge

Falls alle Registereinträge dem gleichen Typ zugeordnet werden sollen, genügt es, diesen mit dem folgenden Parameter anzugeben.

**TYP** Typ der Registereinträge. [ I ] <0>

Falls die Typen der Registereinträge verschieden sein sollen, gibt es zwei Möglichkeiten, die Registereinträge dem jeweiligen Typ zuzuordnen, nämlich einzeln oder bereichsweise. Bei `MODUS=KWIC` ist nur bereichsweise Zuordnung möglich.

Bei Einzelzuordnung wird in jedem Registereintrag das erste (von Leerzeichen verschiedene) Zeichen als Typenkennzeichen interpretiert und anschließend eliminiert. Eine eigene Kennzeichnung dieses Typenkennzeichens ist nicht notwendig. Ein evtl. unmittelbar nachfolgendes Leerzeichen wird übergangen.

Bei bereichsweiser Zuordnung muss mit dem Parameter `TK` die Kennung angegeben werden, hinter der im Text das Typenkennzeichen (ohne Zwischenraum) steht, das jeweils für alle nachfolgenden Registereinträge gelten soll.

Die Zeichen, die als Typenkennzeichen vorkommen, müssen in beiden Fällen mit dem Parameter `TKZ` angegeben werden.

**TK** Zeichenfolgen, die zur Kennzeichnung der Typenkennzeichen dienen. [ IX ]

**TKZ** Zeichen, die als Typenkennzeichen vorkommen. [ VI ]

Die Zuordnung der Typenkennzeichen (ein einzelnes Zeichen) zum jeweiligen Typ (eine Nummer) kann mit dem Parameter `TKN` festgelegt werden. Mit ihm können

parallel zu den Typenkennzeichen, die mit dem Parameter TKZ angegeben sind, die entsprechenden Typen angegeben werden. Wird der Parameter TKN nicht angegeben, so wird dem 1., 2., 3., ... Typenkennzeichen des Parameters TKZ der Typ 1, 2, 3, ... zugeordnet.

**TKN** Typ (Nummer) für die einzelnen Typenkennzeichen. [ I ]

Die Zuordnung zu den Typenkennzeichen erfolgt entsprechend der Reihenfolge, in der die Zeichen mit dem Parameter TKZ angegeben sind.

Sind weniger Nummern als Typenkennzeichen mit dem Parameter TKZ angegeben, so wird für die restlichen Typenkennzeichen der Typ 0 eingesetzt.

## Bestimmen der Schlüsselwörter

(nur bei MODUS=KWIC)

Für den KWIC-Index bildet jeder »Registereintrag«, der durch die oben beschriebenen Parameter bestimmt wird, den Textteil, der die Schlüsselwörter enthält und der gleichzeitig Kontext für diese Schlüsselwörter ist.

Kommen Schlüsselwörter nur in bestimmten Teilen des »Registereintrags« (= Textteil, der als Kontext gilt) vor, so können diese Teile durch die folgenden Parameter abgegrenzt werden.

**ASW** Zeichenfolgen, die den Anfang der Textteile kennzeichnen, die Schlüsselwörter enthalten. [ IX ]

**ESW** Zeichenfolgen, die das Ende der Textteile kennzeichnen, die Schlüsselwörter enthalten. [ IX ]

Werden die Parameter ASW und/oder ESW angegeben, so werden nur die Schlüsselwörter übernommen, die innerhalb der ausgewählten Teile beginnen.

Die ausgewählten Textteile beginnen jeweils mit der Anfangskennung und enden vor der Endekennung bzw. am Ende des Registereintrags, falls keine Endekennung gefunden wird.

Sind beide Parameter ASW und ESW angegeben, so werden nacheinander aus allen mit ASW/ESW gekennzeichneten Textteilen die Schlüsselwörtern übernommen. Der zweite Textteil beginnt nach der Anfangskennung, die dem Ende des ersten folgt. Dabei kann die Anfangskennung mit der Endekennung des vorangehenden Textteils zusammenfallen.

Ist nur der Parameter ASW angegeben, so endet der ausgewählte Textteil, aus dem Schlüsselwörter übernommen werden, am Ende des Registereintrags; ist nur der Parameter ESW angegeben, so beginnt der ausgewählte Textteil, aus dem Schlüsselwörter übernommen werden, am Anfang des Registereintrags.

**(SW** Zeichenfolgen, die bei der Abgrenzung der Textteile, in denen Schlüsselwörter übernommen werden, (ggf. nach Abgrenzung durch ASW und/oder ESW) als öffnende Klammer dienen sollen. [ IX ]

**)SW** Zeichenfolgen, die bei der Abgrenzung der Textteile, aus denen Schlüsselwörter übernommen werden, (ggf. nach Abgrenzung durch ASW und/oder ESW) als schließende Klammer dienen sollen. [ IX ]

Werden die Parameter (SW und/oder )SW angegeben, so werden Schlüsselwörter, die vollständig innerhalb der eingeklammerten Teile stehen, nicht übernommen.

Werden diese Parameter zusätzlich zu den Parametern ASW und/oder ESW angegeben, so erfolgt die Überprüfung auf eingeklammerte Teile jeweils innerhalb der mit den genannten Parametern ausgewählten Teilen.

Die eingeklammerten Teile beginnen jeweils mit der Zeichenfolge, die als öffnende Klammer dient, und enden nach der Zeichenfolge, die als schließende Klammer dient.

Werden bei der Überprüfung auf eingeklammerte Teile unpaarige Klammern festgestellt, so werden die fehlenden Klammern jeweils am Anfang bzw. am Ende des überprüften Textteils logisch ergänzt. Dies gilt auch, wenn nur einer der Parameter (SW oder )SW angegeben ist.

Anmerkung: Der Kontext der Schlüsselwörter wird durch diese Parameter nicht verändert; eingeklammerte Teile bleiben also im Kontext stehen.

Die Schlüsselwörter werden durch Angabe von Trennzeichenfolgen, die zwischen den Schlüsselwörtern stehen, mit dem Parameter SWT bestimmt.

**SWT** Zeichenfolgen, die als Trennzeichen zwischen den einzelnen Schlüsselwörtern stehen. [ IX ]

## Definieren der Referenz

Die Registereinträge können mit einer »Standardreferenz« versehen werden, und/oder es können Textteile aus den Eingabedaten als Referenz übernommen werden. Soll nur die »Standardreferenz« eingesetzt werden, so darf keiner der folgenden Parameter (außer IRL) angegeben werden.

Die Standardreferenz gibt jeweils an, auf welcher Seite, in welcher Zeile und mit welchem Wort der Registereintrag beginnt. Seiten- und Zeilennummern (mit Unterscheidungsnummern) sind durch die Satznummern der Eingabedatei festgelegt, die Wortnummern ergeben sich durch Abzählen der Wörter innerhalb der Zeile. Für die Seitennummer werden 6, für die Zeilennummer 3, für die Unterscheidungsnummer ebenfalls 3 und für die Wortnummer 2 Stellen belegt.

Beim Zählen der Wörter gilt jede Zeichenfolge als ein Wort, die von Zeilenanfang und/oder Zeilenende und/oder einem oder mehreren Leerzeichen eingeschlossen ist. Satzzeichen und Sonderzeichen gehören also entweder zum Wort oder bilden, wenn

sie durch Leerzeichen davon getrennt sind, ein eigenes Wort. Fester Ausschluss (»\_«) gilt nicht als Leerzeichen.

Soll die Referenz aus dem Text entnommen werden, so müssen die Textteile, die als Referenz übernommen werden sollen, mit den Parametern `RFA` und/oder `RFE` bestimmt werden. Falls mit dem Parameter `RFL` nichts anderes angegeben ist, kann die Referenz bis zu 6 Zeichen lang sein. Eine aus dem Text entnommene Referenz gilt jeweils für alle (bis zur nächsten Referenz) folgenden Registereinträge (Ausnahme siehe Parameter `RFG`).

**RFA** Zeichenfolgen, die den Anfang des Textteils kennzeichnen, der als Referenz übernommen werden soll. [ IX ]

**RFE** Zeichenfolgen, die das Ende des Textteils kennzeichnen, der als Referenz übernommen werden soll. [ IX ]

Der ausgewählte Textteil beginnt jeweils nach der Anfangskennung und endet vor der Endekennung bzw. am Ende der Texteinheit, falls keine Endekennung gefunden wird. Enthält der ausgewählte Textteil führende und/oder abschließende Leerzeichen, so werden diese automatisch eliminiert.

Sind beide Parameter `RFA` und `RFE` angegeben, so werden nacheinander alle mit `RFA/RFE` gekennzeichneten Textteile jeweils als Referenz übernommen. Der zweite beginnt nach der Anfangskennung, die dem Ende des ersten folgt. Dabei kann die Anfangskennung mit der Endekennung des vorangehenden als Referenz zu übernehmenden Textteils zusammenfallen.

Ist nur der Parameter `RFA` angegeben, so endet der ausgewählte Textteil, der als Referenz übernommen wird, am Ende der Texteinheit; ist nur der Parameter `RFE` angegeben, so beginnt der ausgewählte Textteil, der als Referenz übernommen wird, am Anfang der Texteinheit.

Mit dem folgenden Parameter ist es möglich, in dem mit `RFA/RFE` ausgewählten Textteil Zeichenfolgen zu ersetzen, bevor er als Referenz übernommen (und bevor ggf. der Parameter `RFT` ausgewertet) wird.

**RFX** Zeichenfolgenpaare (und Ausnahmezeichenfolgen), deren jeweils erste Zeichenfolge durch die jeweils zweite Zeichenfolge in dem als Referenz zu übernehmenden Textteil ersetzt werden soll. [ X ]

Eine Referenz kann auch aus bis zu neun Teilen zusammengesetzt werden. Die einzelnen Referenzteile werden durchnummeriert. Um welchen Referenzteil es sich bei dem mit `RFA/RFE` ausgewählten Textteil jeweils handelt, wird durch die Anfangskennung dieser Textteile bestimmt. Zu diesem Zweck kann parallel zu den mit dem Parameter `RFA` angegebenen Zeichenfolgen mit dem Parameter `RFZ` angegeben werden, welchen Referenzteil die entsprechende Zeichenfolge jeweils kennzeichnet. Dieser Referenzteil wird in der Referenz ersetzt; nachgeordnete Referenzteile (das sind solche mit einer höheren Nummer) werden mit Leerzeichen belegt, falls mit dem Parameter `RFB` nichts anderes angegeben ist.

**RFZ** Nummer des Referenzteils, der mit den einzelnen Anfangskennungen gekennzeichnet ist. [ I ] <1, 1, 1, . . . >

Die Zuordnung der Nummern zu den Zeichenfolgen erfolgt entsprechend der Reihenfolge, in der die Zeichenfolgen mit dem Parameter RFA angegeben sind.

**RFB** Angabe, ob nachgeordnete Referenzteile beibehalten werden sollen oder mit Leerzeichen belegt (gelöscht) werden sollen. [ I ] <0>

0 = mit Leerzeichen belegen (löschen)

1 = andere Referenzteile unverändert lassen

Besteht die Referenz nicht nur aus einem einzigen Teil, so muss mit dem Parameter RFL angegeben werden, wie lang die einzelnen Referenzteile maximal sein dürfen. Durch die Anzahl der angegebenen Längenangaben wird gleichzeitig die Zahl der Referenzteile festgelegt. Besteht die Referenz aus einem einzigen Teil, so muss dieser Parameter nur angegeben werden, falls die voreingestellte Maximallänge (6 Zeichen) durch einen anderen Wert ersetzt werden soll. Ist die Gesamtlänge der Referenz (Summe aller Werte des Parameters RFL) größer als 14, so muss zusätzlich mit dem Parameter IRL ein entsprechender Wert angegeben werden.

**RFL** Länge der einzelnen Referenzteile. [ I ] <6>

Ein mit RFA/RFE ausgewählter Textteil kann auch mehr als einen Referenzteil enthalten. In diesem Fall müssen mit dem Parameter RFT die Zeichenfolgen, die die einzelnen Teile voneinander trennen, angegeben werden.

**RFT** Zeichenfolgen, die die einzelnen Referenzteile voneinander trennen. [ IX ]

Zwischen der Anfangskennung und der ersten Trennzeichenfolge steht der Referenzteil mit der Nummer 1 bzw. mit der durch den Parameter RFZ bestimmten Nummer. Nach den einzelnen Trennzeichenfolgen folgt jeweils der Referenzteil mit der nächsthöheren Nummer. Soll von dieser Regelung abgewichen werden, so kann mit dem Parameter RFN parallel zu den Trennzeichenfolgen des Parameters RFT angegeben werden, welcher Referenzteil mit welcher Trennzeichenfolge eingeleitet wird. Die einzelnen Referenzteile müssen aber in jedem Fall in aufsteigender Reihenfolge hintereinander stehen. Mit dem Parameter RFN kann also erreicht werden, dass fehlende Referenzteile erkannt werden; für sie werden Leerzeichen in die Referenz eingesetzt.

**RFN** Nummern der Referenzteile, die mit den einzelnen Trennzeichenfolgen eingeleitet werden. [ I ]

Die Zuordnung der Nummern zu den Trennzeichenfolgen erfolgt entsprechend der Reihenfolge, in der die Trennzeichenfolgen mit dem Parameter RFT angegeben sind.

Sind weniger Nummern als Trennzeichenfolgen mit dem Parameter RFT angegeben, so erhält ein Referenzteil, der mit einer der restlichen Trennzeichenfolgen eingeleitet wird, eine Nummer, die um 1 höher ist als die des vorangehenden Referenzteils.

Wird versucht, auf Grund der Angabe zum Parameter `RFN` einen Referenzteil einzuleiten, dessen Nummer gleich oder kleiner ist als die des vorangehenden Referenzteils, so bleibt die entsprechende Angabe zum Parameter `RFN` unberücksichtigt. Der Referenzteil erhält in diesem Fall eine Nummer, die um 1 höher ist als die des vorangehenden Referenzteils.

Für die einzelnen Referenzteile sind in den Ausgabesätzen feste Felder vorgesehen. Die Länge der einzelnen Felder wird durch den Parameter `RFL` festgelegt. Mit dem Parameter `RFS` kann für jeden Referenzteil bestimmt werden, ob er links- oder rechtsbündig in das entsprechende Feld abgespeichert werden soll. Von Bedeutung ist dies in der Regel nur für die Sortierung (falls für die Referenz kein eigener Sortierschlüssel aufgebaut wird).

**RFS** Angabe, ob die einzelnen Referenzteile links- oder rechtsbündig in die vorgesehenen Felder abgespeichert werden sollen. [ I ] <0, 0, 0, . . . >

0 = rechtsbündig abspeichern

1 = linksbündig abspeichern

Es empfiehlt sich, einen aus Buchstaben bestehenden Referenzteil linksbündig und einen aus Ziffern bestehenden rechtsbündig abzuspeichern.

Eine aus dem Text entnommene Referenz gilt jeweils für alle folgenden Registereinträge. Steht in einer Texteinheit ein Registereintrag vor dem ersten Textteil, der als Referenz übernommen werden soll, so gilt für diesen Registereintrag die zuletzt in einer vorangehenden Texteinheit vorgekommene Referenz. Soll in diesem Fall schon die Referenz gelten, die als erste in der Texteinheit vorkommt, so kann dies mit dem Parameter `RFG` angegeben werden. Für Registereinträge, die in einer Texteinheit nach einem Textteil stehen, der als Referenz übernommen werden soll, gilt in jedem Fall jeweils die zuletzt vorgekommene Referenz.

**RFG** Angabe, ob der erste als Referenz zu übernehmende Textteil in einer Texteinheit schon vom Anfang der Texteinheit an gelten soll. [ I ] <0>

0 = Er soll ab der Stelle gelten, an der er steht.

1 = Er soll vom Anfang der Texteinheit an gelten.

Soll die aus dem Text entnommene Referenz mit der Standardreferenz kombiniert werden, so sind die Angaben des folgenden Parameters erforderlich.

**RFR** Reihenfolge und Anzahl der einzelnen Referenzteile. [ I ]

Es können drei Zahlenwerte angegeben werden.

1. Zahl: Reihenfolge von Standard- und Textreferenz <0>

0 = Nur eine Referenz (falls Parameter `RFA` und/oder `RFE` angegeben ist, Textreferenz; andernfalls Standardreferenz)

1 = Textreferenz vor Standardreferenz

2 = Standardreferenz vor Textreferenz

2. Zahl: Anzahl der Referenzteile der an erster Stelle stehenden Referenz <0>



3. Zahl: Anzahl der Referenzteile der an zweiter Stelle stehenden Referenz <0>

Die Summe der mit der zweiten und dritten Zahl angegebenen Referenzteile muss zusammen der mit dem Parameter RFL angegebenen Anzahl von Referenzlängen entsprechen.

In den Ausgabesätzen sind (bei MODUS=+) für die Referenz 14 Stellen vorgesehen. Dieser Wert kann mit dem Parameter IRL verändert werden. Zu beachten ist dann aber, dass diese Änderung auch bei nachfolgenden Programmen, die diese Daten verarbeiten, berücksichtigt werden muss. Eine Änderung der Voreinstellung ist notwendig, wenn die Gesamtlänge der Referenz (= Summe der mit dem Parameter RFL angegebenen Einzelwerte) größer als 14 ist. Sie ist empfehlenswert, wenn die Gesamtlänge der Referenz bei großen Datenmengen kleiner als 14 ist, damit Plattenplatz gespart wird.

**IRL** Interne Referenzlänge; das ist die Anzahl der Stellen, die für die Referenz in den Ausgabesätzen vorgesehen werden sollen. [ I ] <14>

Falls die Standardreferenz benutzt wird, kann nur einer der Werte 6, 9, 12 und 14 eingestellt werden; es werden dann nur die Referenzteile eingesetzt, für die der Platz reicht.

## Markieren der Referenz

(nicht bei MODUS=KWIC)

Bei der Aufbereitung eines Registers mit dem Kommando #RAUFBEREITE können die Referenzen zu einzelnen Registereinträgen verschieden behandelt werden. So soll z. B. bei Verweiseinträgen die Referenz ganz fehlen; in anderen Fällen soll die Referenz z. B. mit dem Zusatz »f« oder »ff« versehen werden oder es sollen zwei aufeinander folgende Referenzen so ausgegeben werden, dass sie als Bereichsanfang und Bereichsende erkennbar sind. Dazu müssen die Referenzen solcher Registereinträge beim Vorbereiten des Registers mit einer entsprechenden Markierung versehen werden.

Soll keiner der Registereinträge eines Registers eine Referenz erhalten, so kann (beim Aufruf der Kommandos #RVORBEREITE und #RAUFBEREITE) MODUS=- angegeben werden. Sollen Registereinträge ohne Referenz mit solchen mit Referenz gemischt werden, so müssen die Registereinträge, die bei der Aufbereitung des Registers (mit dem Kommando #RAUFBEREITE) keine Referenz bekommen sollen, bei der Vorbereitung des Registers (mit dem Kommando #RVORBEREITE) trotzdem mit einer Referenz versehen werden. Diese muss entsprechend markiert werden. Dazu gibt es zwei Möglichkeiten: Sollen alle Registereinträge, die mit einem Aufruf des Kommandos #RVORBEREITE erzeugt werden, gleich behandelt werden, so kann mit dem Parameter RFM angegeben werden, ob die Registereinträge später eine Referenz erhalten sollen oder nicht. Sollen nur bestimmte Registereinträge später keine Referenz erhalten, so müssen diese mit einem eindeutigen Kennzeichen (keinesfalls Leerzeichen) versehen sein. Diese Kennzeichen müssen mit dem Parameter ORF angegeben werden.

**RFM** Angabe, ob die Registereinträge bei der Aufbereitung des Registers mit dem Kommando #RAUFBEREITE eine Referenz erhalten sollen. [ I ]  
<1>

0 = Registereinträge sollen keine Referenz erhalten

1 = Registereinträge sollen eine Referenz erhalten

**ORF** Zeichen, die zur Kennzeichnung der Registereinträge verwendet sind, die ohne Referenz ins Register sollen. [ VI ]

Das Kennzeichen muss am Anfang des jeweiligen Registereintrags (ggf. nach dem Typenkennzeichen) stehen. Es wird, nachdem es als solches erkannt ist, eliminiert. Ein eventuell unmittelbar nachfolgendes Leerzeichen wird ignoriert.

Sollen Referenzen mit einem Zusatz versehen werden, so müssen die entsprechenden Registereinträge mit einem eindeutigen Kennzeichen (keinesfalls Leerzeichen) versehen sein. Diese Kennzeichen müssen mit einem der beiden folgenden Parameter angegeben werden.

**MF** Zeichen, die zur Kennzeichnung der Registereinträge verwendet sind, nach deren Referenz das Zeichen »f« oder eine andere, mit dem Parameter F des Kommandos #RAUFBEREITE anzugebende Zeichenfolge ergänzt werden soll. [ VI ]

**MFf** Zeichen, die zur Kennzeichnung der Registereinträge verwendet sind, nach deren Referenz die Zeichen »ff« oder eine andere, mit dem Parameter FF des Kommandos #RAUFBEREITE anzugebende Zeichenfolge ergänzt werden soll. [ VI ]

Das Kennzeichen muss am Anfang des jeweiligen Registereintrags (ggf. nach dem Typenkennzeichen) stehen. Es wird, nachdem es als solches erkannt ist, eliminiert. Ein eventuell unmittelbar nachfolgendes Leerzeichen wird ignoriert.

Sollen die Referenzen den Anfang bzw. das Ende eines Bereichs angeben, so müssen die entsprechenden Registereinträge mit einem eindeutigen Kennzeichen (keinesfalls Leerzeichen) versehen sein. Diese Kennzeichen müssen mit dem Parameter VON bzw. mit dem Parameter BIS angegeben werden.

**VON** Zeichen, die zur Kennzeichnung der Registereinträge verwendet sind, deren Referenz als Anfang eines Bereichs gelten soll. [ VI ]

**BIS** Zeichen, die zur Kennzeichnung der Registereinträge verwendet sind, deren Referenz als Ende eines Bereichs gelten soll. [ VI ]

Das Kennzeichen muss am Anfang des jeweiligen Registereintrags (ggf. nach dem Typenkennzeichen) stehen. Es wird, nachdem es als solches erkannt ist, eliminiert. Ein eventuell unmittelbar nachfolgendes Leerzeichen wird ignoriert.

## Häufigkeitsangaben

(nicht bei MODUS=KWIC)

Bei der Aufbereitung des Registers mit dem Kommando #RAUFBEREITE können Häufigkeitsangaben ausgegeben werden. Dazu werden die jeweils gleichen Register-einträge beim Zusammenfassen gezählt, d. h. jeder vom Kommando #RVORBEREITE erzeugte Register-eintrag hat implizit die Häufigkeit 1. Es gibt jedoch die Möglichkeit, Register-einträge mit einer expliziten Häufigkeit zu versehen. Dazu muss der entsprechende Zahlenwert jeweils vor bzw. hinter einer eindeutigen Zeichenfolge am Anfang bzw. Ende des Register-eintrags angegeben sein. Diese Zeichenfolge und die Häufigkeitsangabe werden nach der Auswertung eliminiert. Für Register-einträge ohne eine solche Häufigkeitsangabe gilt die Häufigkeit 1. Die Zeichenfolgen, mit denen die Häufigkeit gekennzeichnet ist, müssen mit dem Parameter HFE bzw. HFA angegeben werden. Außerdem muss mit dem Parameter HFL die Anzahl der Dezimalstellen angegeben werden, die für die Häufigkeit maximal benötigt werden.

**HFE** Zeichenfolgen, die das Ende der expliziten Häufigkeitsangabe in den Register-einträgen kennzeichnen, falls die Häufigkeitsangabe am Anfang der Einträge steht. [ IX ]

**HFA** Zeichenfolgen, die den Anfang der expliziten Häufigkeitsangabe in den Register-einträgen kennzeichnen, falls die Häufigkeitsangabe am Ende der Einträge steht. [ IX ]

**HFL** Anzahl der Dezimalstellen, die in jedem erzeugten Register-eintrag für die Häufigkeitsangabe vorgesehen werden sollen. [ I ] <0>

Register-einträge, die bei der Berechnung der Häufigkeit (mit dem Kommando #RAUFBEREITE) nicht mitgezählt werden sollen, aber dennoch im Register erscheinen sollen (z. B. Verweise auf andere Einträge) können entweder mit der expliziten Häufigkeitsangabe 0 (s. o.) oder mit einem eindeutigen Kennzeichen (keinesfalls Leerzeichen) versehen werden. Diese Kennzeichen müssen mit dem Parameter OHF angegeben werden.

**OHF** Zeichen, die zur Kennzeichnung der Register-einträge verwendet sind, die bei der Berechnung der Häufigkeit nicht mitgezählt werden sollen. [ VI ]

Das Kennzeichen muss am Anfang des jeweiligen Register-eintrags (ggf. nach dem Typenkennzeichen) stehen. Es wird, nachdem es solches erkannt ist, (und ggf. auch für die Parameter ORF, MF, MFF, VON, BIS berücksichtigt wurde), eliminiert. Ein evtl. unmittelbar nachfolgendes Leerzeichen wird ignoriert.

## Umdrehen der Registereinträge

(nicht bei MODUS=KWIC)

Registereinträge können in sich umgedreht werden (z. B. wird dabei aus »Friedrich von Schiller« »Schiller, Friedrich von«). Dazu wird der Registereintrag an einer mit dem Parameter UMP angegebenen Zeichenfolge bzw. am letzten Wortzwischenraum (ein oder mehrere Leerzeichen) unterteilt und dann die beiden Teile vertauscht; zusätzlich wird zwischen den beiden Teilen die Zeichenfolge », « eingefügt, falls mit dem Parameter UME nichts anderes angegeben ist.

**UMD** Angabe, ob die Registereinträge umgedreht werden sollen. Falls mit dem Parameter TKZ Typenkennzeichen definiert werden, muss ggf. für jedes Typenkennzeichen ein Wert angegeben werden. [ I ] <0 bzw. 0, 0, 0, . . . >

- 0 = Registereinträge sollen nicht umgedreht werden.
- 1 = Registereinträge sollen in umgedrehter Form ins Register, falls sie eine mit dem Parameter UMP angegebene Zeichenfolge oder ein Leerzeichen enthalten, andernfalls sollen sie in normaler Form ins Register.
- 2 = Registereinträge sollen in normaler Form ins Register und zusätzlich in umgedrehter Form, falls sie eine mit dem Parameter UMP angegebene Zeichenfolge oder ein Leerzeichen enthalten.
- 3 = Registereinträge sollen in umgedrehter Form ins Register, falls sie eine mit dem Parameter UMP angegebene Zeichenfolge enthalten, andernfalls sollen sie in normaler Form ins Register.
- 4 = Registereinträge sollen in normaler Form ins Register und zusätzlich in umgedrehter Form, falls sie eine mit dem Parameter UMP angegebene Zeichenfolge enthalten.

Die Angaben 1 und 2 unterscheiden sich von den Angaben 3 und 4 dadurch, dass ggf. am letzten Leerzeichen umgedreht wird, falls der Registereintrag nicht an einer mit dem Parameter UMP angegebenen Zeichenfolge umgedreht werden kann.

Sind die Registereinträge mit Typenkennzeichen versehen, die mit dem Parameter TKZ angegeben sind, so kann zu jedem Typenkennzeichen ein Wert mit dem Parameter UMD angegeben werden. Die Zuordnung zu den Typenkennzeichen erfolgt entsprechend der Reihenfolge, in der die Zeichen mit dem Parameter TKZ angegeben sind. Sind weniger Zahlenwerte als Typenkennzeichen angegeben, so wird für die restlichen Typenkennzeichen der Wert 0 angenommen.

**UMP** Zeichenfolgen, an denen die Registereinträge zum Umdrehen unterteilt werden sollen. [ IX ]

Soll ein Registereintrag umgedreht werden, so wird er von links nach rechts nach einer mit dem Parameter UMP angegebenen Zeichenfolge durchsucht. Wird eine solche gefunden, so wird der Registereintrag an dieser Stelle umgedreht, falls diese Zeichenfolge nicht unmittelbar am Anfang oder am Ende des Registereintrags steht. Wird keine der ange-

gegebenen Zeichenfolgen gefunden und ist die dazugehörige Angabe mit dem Parameter UMD 1 oder 2, so wird der Registereintrag ggf. am letzten Wortzwischenraum (ein oder mehrere Leerzeichen) umgedreht. Die gefundene Zeichenfolge bzw. das Leerzeichen wird beim Umdrehen eliminiert. Ein Leerzeichen, das unmittelbar vor oder nach der gefundenen Zeichenfolge steht, wird ebenfalls eliminiert.

**UME** Textteile, die beim Umdrehen der Registereinträge zwischen den beiden vertauschten Teilen eingefügt werden sollen. [ II ] <, >

Sind die Registereinträge mit Typenkennzeichen versehen, die mit dem Parameter TKZ angegeben sind, so kann zu jedem Typenkennzeichen eine Zeichenfolge mit dem Parameter UME angegeben werden. Die Zuordnung zu den Typenkennzeichen erfolgt entsprechend der Reihenfolge, in der die Zeichen mit dem Parameter TKZ angegeben sind. Sind weniger Zeichenfolgen als Typenkennzeichen angegeben, so wird für die restlichen Typenkennzeichen die voreingestellte Zeichenfolge (», «) eingefügt.

## Auswählen der Registereinträge bzw. Schlüsselwörter

Falls nicht alle Schlüsselwörter (bei MODUS=KWIC) bzw. Registereinträge (bei allen anderen Modi) ausgegeben werden sollen, so können sie mit den folgenden Parametern ausgewählt werden.

Stimmt ein Registereintrag/Schlüsselwort mit einem zum Parameter T+N, T+U oder T+ angegebenen überein, wird er ins Register übernommen. In diesem Fall unterbleiben weitere Überprüfungen auf Grund der übrigen Parameter dieses Abschnitts. Sind keine anderen Parameter dieses Abschnitts angegeben, so werden nur diese Registereinträge/Schlüsselwörter übernommen.

**T+N** Registereinträge/Schlüsselwörter, die ins Register übernommen werden sollen. [ III ]

Groß- und Kleinbuchstaben werden nicht unterschieden.

**T+U** Registereinträge/Schlüsselwörter, die ins Register übernommen werden sollen. [ III ]

Groß- und Kleinbuchstaben werden unterschieden.

**T+** Registereinträge/Schlüsselwörter, die ins Register übernommen werden sollen. [ III ]

Beim Vergleich wird die Sonder-Sortierfolge zugrunde gelegt; d. h. die Sonderzeichen, die mit x bzw. ^x codiert sind (z. B. ! und ^!, & und ^&) sowie die Groß- und Kleinbuchstaben werden nicht unterschieden.

Stimmt ein Registereintrag/Schlüsselwort mit einem zum Parameter T-N, T-U oder T- angegebenen überein, wird er nicht ins Register übernommen. In diesem Fall unterbleiben weitere Überprüfungen auf Grund der übrigen Parameter dieses Abschnitts.

- T-N** Registereinträge/Schlüsselwörter, die wegfallen sollen. [ III ]  
Groß- und Kleinbuchstaben werden nicht unterschieden.
- T-U** Registereinträge/Schlüsselwörter, die wegfallen sollen. [ III ]  
Groß- und Kleinbuchstaben werden unterschieden.
- T-** Registereinträge/Schlüsselwörter, die wegfallen sollen. [ III ]  
Beim Vergleich wird die Sonder-Sortierfolge zugrunde gelegt; d. h. die Sonderzeichen, die mit x bzw. ^x codiert sind (z. B. ! und ^!, & und ^&) sowie die Groß- und Kleinbuchstaben werden nicht unterschieden.

Mit den Parametern ZF+, TA+ und TE+ können Bedingungen angegeben werden, unter denen ein Registereintrag/Schlüsselwort ins Register übernommen werden soll. Falls von diesen Parametern einer oder mehrere angegeben sind, muss ein Registereintrag/Schlüsselwort mindestens eine dieser Bedingungen erfüllen, um nicht wegzufallen.

- ZF+** Zeichenfolgen, von denen mindestens eine im Registereintrag/Schlüsselwort vorkommen muss, damit er/es ins Register übernommen wird. [ IX ]
- TA+** Zeichenfolgen, von denen mindestens eine mit dem Anfang des Registereintrags/Schlüsselwortes übereinstimmen muss, damit er/es ins Register übernommen wird. [ VIII-a ]
- TE+** Zeichenfolgen, von denen mindestens eine mit dem Ende des Registereintrags/Schlüsselwortes übereinstimmen muss, damit er/es ins Register übernommen wird. [ VIII-b ]

Mit den Parameter ZF-, TA- und TE- können Bedingungen angegeben werden, unter denen ein Registereintrags/Schlüsselwort nicht ins Register übernommen werden soll. Falls von diesen Parametern einer oder mehrere angegeben sind, genügt es, wenn ein Registereintrags/Schlüsselwort eine dieser Bedingungen erfüllt, um wegzufallen.

- ZF-** Zeichenfolgen, von denen keine im Registereintrag/Schlüsselwort vorkommen darf, damit er/es ins Register übernommen wird. [ IX ]
- TA-** Zeichenfolgen, von denen keine mit dem Anfang des Registereintrags/Schlüsselwortes übereinstimmen darf, damit er/es ins Register übernommen wird. [ VIII-a ]
- TE-** Zeichenfolgen, von denen keine mit dem Ende des Registereintrags/Schlüsselwortes übereinstimmen darf, damit er/es ins Register übernommen wird. [ VIII-b ]

Sind sowohl einer oder mehrere der Parameter ZF+, TA+ und TE+ als auch einer oder mehrere der Parameter ZF-, TA- und TE- angegeben, so muss ein Registereintrag/Schlüsselwort mindestens eine der Bedingungen der Parameter ZF+, TA+ und TE+ erfüllen und darf keine der Bedingungen der Parameter ZF-, TA- und TE- erfüllen, um ins Register übernommen zu werden.

## Überprüfen der Länge der Registereinträge

- MTL** Maximale Länge eines Registereintrags. [ I ] <999999>  
Ist ein Registereintrag länger als angegeben, so wird er mit einer entsprechenden Fehlermeldung ausgegeben.

## Ergänzen der Registereinträge

(nicht bei MODUS=KWIC)

Vor und/oder nach jedem Registereintrag kann ein Ergänzungstext (z. B. ein Oberbegriff, unter dem ein Registereintrag im Register erscheinen soll) hinzugefügt werden. Der Ergänzungstext wird aus entsprechend im Text gekennzeichneten Teilen gebildet und gilt jeweils für alle folgenden Registereinträge. Bei den im folgenden beschriebenen Parametern beziehen sich die mit »EV« beginnenden Parameter auf den Ergänzungstext, der vor jedem Registereintrag ergänzt wird, und die mit »EN« beginnenden auf den, der nach jedem Registereintrag ergänzt wird.

- EVK / ENK** Textteil, der zwischen Ergänzungstext und Registereintrag eingefügt werden soll. [ II ]

Der mit diesem Parameter angegebene Textteil kann auch als konstanter Ergänzungstext angesehen werden, falls der Ergänzungstext nicht durch Angabe einer der folgenden Parameter aus den Eingabedaten bestimmt wird.

- EVA / ENA** Zeichenfolgen, die den Anfang des Textteils kennzeichnen, der als Ergänzungstext übernommen werden soll. [ IX ]

- EVE / ENE** Zeichenfolgen, die das Ende des Textteils kennzeichnen, der als Ergänzungstext übernommen werden soll. [ IX ]

Der ausgewählte Textteil beginnt jeweils nach der Anfangskennung und endet vor der Endekennung bzw. am Ende der Texteinheit, falls keine Endekennung gefunden wird.

Sind beide Parameter EVA und EVE bzw. ENA und ENE angegeben, so werden nacheinander alle mit EVA/EVE bzw. mit ENA/ENE gekennzeichneten Textteile jeweils als Ergänzungstext übernommen. Der zweite beginnt nach der Anfangskennung, die dem Ende des ersten folgt. Dabei kann die Anfangskennung mit der Endekennung des vorangehenden als Ergänzungstext zu übernehmenden Textteils zusammenfallen.

Ist nur der Parameter EVA bzw. ENA angegeben, so endet der ausgewählte Textteil, der als Ergänzungstext übernommen wird, am Ende der Texteinheit; ist nur der Parameter EVE bzw. ENE angegeben, so beginnt der ausgewählte Textteil, der als Ergänzungstext übernommen wird, am Anfang der Texteinheit.

Mit dem folgenden Parameter ist es möglich, in dem mit EVA/EVE bzw. mit ENA/ENE ausgewählten Textteil Zeichenfolgen zu ersetzen, bevor er als Ergänzungstext übernommen (und bevor ggf. der Parameter EVT bzw. ENT ausgewertet) wird.

**EVX / ENX** Zeichenfolgenpaare (und Ausnahmezeichenfolgen), deren jeweils erste Zeichenfolge durch die jeweils zweite Zeichenfolge in dem als Ergänzungstext zu übernehmenden Textteil ersetzt werden soll. [ X ]

Ein Ergänzungstext kann auch aus mehreren Teilen zusammengesetzt werden (z. B. bei hierarchischen Oberbegriffen). Die einzelnen Teile des Ergänzungstextes werden durchnummeriert. Um welchen Teil des Ergänzungstextes es sich bei dem mit EVA/EVE bzw. mit ENA/ENE ausgewählten Textteil jeweils handelt, wird durch die Anfangskennung dieser Textteile bestimmt. Zu diesem Zweck kann parallel zu den mit dem Parameter EVA bzw. ENA angegebenen Zeichenfolgen mit dem Parameter EVZ bzw. ENZ angegeben werden, welchen Teil des Ergänzungstextes die entsprechende Zeichenfolge jeweils kennzeichnet. Dieser Teil des Ergänzungstextes wird im Ergänzungstext ersetzt; nachgeordnete Teile des Ergänzungstextes (das sind solche mit einer höheren Nummer) werden gelöscht, falls mit den Parameter EVB/ENB nichts anderes angegeben ist.

**EVZ / ENZ** Nummer des Ergänzungstext-Teils, der mit den einzelnen Anfangskennungen gekennzeichnet ist. [ I ] <1, 1, 1, . . . >

Die Zuordnung der Nummern zu den Zeichenfolgen erfolgt entsprechend der Reihenfolge, in der die Zeichenfolgen mit dem Parameter EVA bzw. ENA angegeben sind.

**EVB / ENB** Angabe, ob nachgeordnete Ergänzungstext-Teile beibehalten werden sollen oder gelöscht werden sollen. [ I ] <0>

0 = nachgeordnete Ergänzungstext-Teile löschen  
1 = andere Ergänzungstext-Teile unverändert lassen

Ein mit EVA/EVE bzw. mit ENA/ENE ausgewählter Textteil kann auch mehr als einen Teil des Ergänzungstextes enthalten. In diesem Fall müssen mit dem Parameter EVT bzw. ENT die Zeichenfolgen, die die einzelnen Teile voneinander trennen, angegeben werden.

**EVT / ENT** Zeichenfolgen, die die einzelnen Ergänzungstext-Teile voneinander trennen. [ IX ]

Zwischen der Anfangskennung und der ersten Trennzeichenfolge steht der Ergänzungstext-Teil mit der Nummer 1 bzw. mit der durch den Parameter EVZ bzw. ENZ bestimmten Nummer. Nach den einzelnen Trennzeichenfolgen folgt jeweils der Ergänzungstext-Teil mit der nächsthöheren Nummer. Soll von dieser Regelung abgewichen werden, so kann mit dem Parameter EVN bzw. ENN parallel zu den Trennzeichenfolgen zum Parameter EVT bzw. ENT angegeben werden, welcher Ergänzungstext-Teil mit welcher Trennzeichenfolge eingeleitet wird. Die einzelnen Ergänzungstext-Teile müssen aber in jedem Fall in aufsteigender Reihenfolge hintereinander stehen.

**EVN / ENN** Nummern der Ergänzungstext-Teile, die mit den einzelnen Trennzeichenfolgen eingeleitet werden. [ I ]

Die Zuordnung der Nummern zu den Trennzeichenfolgen erfolgt entsprechend der Reihenfolge, in der die Trennzeichenfolgen mit dem Parameter EVT bzw. ENT angegeben sind.



Sind weniger Nummern als Trennzeichenfolgen mit dem Parameter `EVT` bzw. `ENT` angegeben, so erhält ein Ergänzungstext-Teil, der mit einer der restlichen Trennzeichenfolgen eingeleitet wird, eine Nummer, die um 1 höher ist als die des vorangehenden Ergänzungstext-Teils.

Wird versucht, auf Grund der Angabe zum Parameter `EVN` bzw. `ENN` einen Ergänzungstext-Teil einzuleiten, dessen Nummer gleich oder kleiner ist als die des vorangehenden Ergänzungstext-Teils, so bleibt die entsprechende Angabe zum Parameter `EVN` bzw. `ENN` unberücksichtigt. Der Ergänzungstext-Teil erhält in diesem Fall eine Nummer, die um 1 höher ist als die des vorangehenden Ergänzungstext-Teils.

## Gruppenweise Sortieren

- NSN** Zeichenfolgen, die am Anfang eines Registereintrags den Beginn einer neuen Gruppe von Registereinträgen kennzeichnen, also die Stelle, ab der die Registereinträge die nächste Sortiernummer erhalten. [ VIII-a ]
- Die Anzahl der Stellen, die für die Sortiernummer vorgesehen werden sollen, muss mit dem Parameter `SNL` angegeben werden.
- SNL** Sortiernummerlänge (maximale Stellenzahl). [ I ] <0>

## Bestimmen der ZIEL-Datei

Mit einem Aufruf des Kommandos `#RVORBEREITE` können auch Registereinträge für verschiedene Register erstellt werden, falls die einzelnen Registereinträge mit Typenkennzeichen versehen sind, die mit dem Parameter `TKZ` angegeben sind. Dazu kann für jedes Register im Aufruf eine eigene `ZIEL`-Datei angegeben werden; mit dem folgenden Parameter muss dann angegeben werden, welche Registereinträge in welche `ZIEL`-Datei ausgegeben werden sollen.

- ZD** Nummer der `ZIEL`-Datei, in die die Registereinträge in Abhängigkeit von ihrem Typenkennzeichen ausgegeben werden sollen. [ I ]
- Die Nummern der `ZIEL`-Dateien werden durch Abzählen bestimmt. Die erste erhält die Nummer 1, die zweite die Nummer 2, usw.
- Die Zuordnung der Nummern zu den Typenkennzeichen erfolgt entsprechend der Reihenfolge, in der die Zeichen mit dem Parameter `TKZ` angegeben sind.
- Wird statt der Nummer einer `ZIEL`-Datei der Wert 0 angegeben, so werden die Registereinträge mit dem entsprechenden Typenkennzeichen nicht ausgegeben.
- Sind weniger Nummern als Typenkennzeichen mit dem Parameter `TKZ` angegeben, so werden die Registereinträge mit den restlichen Typenkennzeichen nicht ausgegeben.

Mit dem Parameter ZDU wird die ZIEL-Datei für Registereinträge mit undefiniertem Typenkennzeichen (d. h. mit einem Typenkennzeichen, das mit dem Parameter TKZ nicht angegeben ist) festgelegt.

**ZDU** Nummer der ZIEL-Datei, in die die Registereinträge mit undefiniertem Typenkennzeichen ausgegeben werden sollen. [ I ] <0>

Die Nummern der ZIEL-Dateien werden durch Abzählen bestimmt. Die erste erhält die Nummer 1, die zweite die Nummer 2, usw.

Wird statt der Nummer einer ZIEL-Datei der Wert 0 angegeben, so werden die Registereinträge mit undefiniertem Typenkennzeichen nicht ausgegeben.

## Erstellen der Sortierschlüssel

Es sind insgesamt neun Sortierschlüssel möglich. Sie sind von 1 bis 9 durchnummeriert. Bei den folgenden Parametern, bei denen Zahlenwerte erwartet werden, kann für jeden Sortierschlüssel ein Zahlenwert angegeben werden; dabei gilt jeweils die n-te Zahl für den n-ten Sortierschlüssel. Bei allen anderen Parametern ist das letzte Zeichen der Parameterkennung eine Ziffer (im folgenden steht ein n für diese Ziffer), die die Nummer des Sortierschlüssels angibt, für den der Parameter gilt. Die Sortierschlüssel können wie folgt verwendet werden:

- bei MODUS=-  
Sortierschlüssel 1 bis 3 für den Registereintrag
- bei MODUS=+  
Sortierschlüssel 1 bis 3 für den Registereintrag  
Sortierschlüssel 7 bis 9 für die Referenz
- bei MODUS=KWIC  
Sortierschlüssel 1 bis 3 für das Schlüsselwort  
Sortierschlüssel 4 bis 6 für den Kontext  
Sortierschlüssel 7 bis 9 für die Referenz

Im folgenden steht die Bezeichnung Sortiertext für den Text, aus dem der jeweilige Sortierschlüssel nach den Angaben der folgenden Parameter gebildet wird, also für Registereintrag, Schlüsselwort, Kontext, Referenz, je nachdem, welcher Sortierschlüssel erzeugt wird.

Für den Fall, dass die gewünschte Reihenfolge der Referenzen im Register mit der Reihenfolge der Referenzen in den Eingabedaten übereinstimmt, braucht die Referenz im allgemeinen beim Sortieren nicht berücksichtigt zu werden.

Ein Sortierschlüssel für die Referenz ist nur dann notwendig, wenn sie nicht unverändert als Sortierfeld (beim Kommando #SORTIERE bzw. #MISCHE) benutzt werden kann, d. h. wenn damit nicht die gewünschte Reihenfolge erreicht wird.

Die einzelnen Teile der Referenz werden zum Sortiertext lückenlos aneinander gehängt. Falls zum Erstellen der Sortierschlüssel eine Abgrenzung der einzelnen Teile

notwendig ist, kann vor jedem Referenzteil mit dem folgenden Parameter eine Kennung eingefügt werden.

**RSK** Textteile, die zur Kennzeichnung der einzelnen Referenzteile im Sortiertext eingefügt werden sollen. Der erste angegebene Textteil wird vor dem ersten Referenzteil eingefügt, der zweite vor dem zweiten usw. [ II ]

Dieser Parameter ist nur zur Erstellung des Sortiertextes aus der Referenz für die Sortierschlüssel 7 bis 9 von Bedeutung. Die Referenz selbst wird dadurch nicht verändert.

Die folgenden Parameter zum Erstellen der Sortierschlüssel werden in der angegebenen Reihenfolge abgearbeitet. Dies muss z. B. beachtet werden, wenn mit dem Parameter **DEZ** Zahlen mit führenden Nullen ergänzt werden und dann in Such- oder Ausnahmezeichenfolgen des entsprechenden Parameters **XSn** Zahlen angegeben werden.

Die folgenden Auswahlparameter (**ASn** bis **KL5**) sind nur notwendig, falls nicht der ganze Sortiertext im jeweiligen Sortierschlüssel berücksichtigt werden soll.

**ASn** Zeichenfolgen, die den Anfang des Teils des Sortiertextes kennzeichnen, der im n-ten Sortierschlüssel berücksichtigt werden soll. [ IX ]

**ESn** Zeichenfolgen, die das Ende des Teils des Sortiertextes kennzeichnen, der im n-ten Sortierschlüssel berücksichtigt werden soll. [ IX ]

**AES** Index für die Parameter **ASn** und **ESn**. [ I ] <1, 1, 1, 1, 1, 1, 1, 1, 1>

Es können ein bis neun Zahlenwerte angegeben werden:

1 = Es wird der erste mit A/E gekennzeichnete Textteil (er beginnt mit einer Anfangskennung und endet vor der nachfolgenden Endekennung bzw. am Ende des Sortiertextes) ausgewählt.

Ist nur A angegeben, so endet der ausgewählte Textteil am Ende des Sortiertextes; ist nur E angegeben, so beginnt der ausgewählte Textteil am Anfang des Sortiertextes.

0 = Es wird der Teil des Sortiertextes ausgewählt, der bei 1 wegfallen würde.

3 = Wie 1, jedoch wird nicht nur der erste, sondern es werden alle mit A/E gekennzeichneten Textteile (der zweite beginnt mit der Anfangskennung, die dem Ende des ersten mit A/E gekennzeichneten Textteils folgt) ausgewählt.

Ist nur A angegeben, so beginnt der ausgewählte Textteil mit der letzten in der Texteinheit vorkommenden Anfangskennung und endet am Ende des Sortiertextes; ist nur E angegeben, so beginnt der ausgewählte Textteil am Anfang des Sortiertextes und endet vor der letzten im Sortiertext vorkommenden Endekennung.

2 = Es wird der Teil des Sortiertextes ausgewählt, der bei 3 wegfallen würde.

Bei den Werten 0 bis 3 wird die Anfangskennung jeweils zum nachfolgenden Text gerechnet, während die Endekennung nicht mehr

zum davor stehenden Text gerechnet wird. Durch Aufaddieren von 10 und/oder 20 kann diese Regelung für die Anfangs- und/oder Endekennung umgekehrt werden. Wird zu den Werten 10 aufaddiert (also 10 bis 13 angegeben), so wird die Anfangskennung jeweils nicht zum nachfolgenden Text gerechnet; wird 20 addiert, so wird die Endekennung jeweils noch zum davor stehenden Text gerechnet; wird 30 addiert, so wird die Anfangskennung nicht zum nachfolgenden Text und die Endekennung zum davor stehenden Text gerechnet.

Bei den Werten 2 und 3 wird die nächste Anfangskennung ab der ersten Position der zuletzt gefundenen Endekennung gesucht, da sie nicht mehr zum davor stehenden Textteil gehört. Anfangs- und Endekennung können sich also im Text überschneiden. Wurde auf diese Werte 20 oder 30 aufaddiert, so wird die nächste Anfangskennung erst ab der Position gesucht, die auf die letzte Position der zuletzt gefundenen Endekennung folgt, da diese noch zum davor stehenden Textteil gehört.

**(Sn** Zeichenfolgen, die bei der Auswahl des Teils des Sortiertextes (falls  $AS_n$  und/oder  $ES_n$  nicht angegeben) bzw. bei der Eliminierung von Textteilen aus dem bereits mit  $AS_n$  und/oder  $ES_n$  ausgewählten Teil des Sortiertextes, der im  $n$ -ten Sortierschlüssel zu berücksichtigt ist, als öffnende Klammer dienen sollen. [ IX ]

**) Sn** Zeichenfolgen, die bei der Auswahl des Teils des Sortiertextes (falls  $AS_n$  und/oder  $ES_n$  nicht angegeben) bzw. bei der Eliminierung von Textteilen aus dem bereits mit  $AS_n$  und/oder  $ES_n$  ausgewählten Teil des Sortiertextes, der im  $n$ -ten Sortierschlüssel zu berücksichtigt ist, als schließende Klammer dienen sollen. [ IX ]

**KL** Index für die Parameter (Sn und ) Sn. [ I ] <0, 0, 0, 0, 0, 0, 0, 0, 0, 0>

Es können ein bis neun Zahlenwerte angegeben werden:

- 0 = Es werden die Teile (einschließlich der Klammern) eliminiert, die eingeklammert sind. Fehlende Klammern werden am Anfang bzw. am Ende des Sortiertextes bzw. des bereits ausgewählten Textteils logisch ergänzt.
- 1 = Es werden die Teile ausgewählt, die bei 0 wegfallen würden.
- 2 = Wie 0, jedoch werden fehlende Klammern nicht ergänzt, sondern die unpaarigen Klammern werden ignoriert.
- 3 = Es werden die Teile ausgewählt, die bei 2 wegfallen würden.

Bei den Werten 0 bis 3 werden die Klammern jeweils zum eingeklammerten Text gerechnet und werden mit ihm eliminiert bzw. bleiben mit ihm erhalten. Durch Aufaddieren von 10 und/oder 20 kann diese Regelung für die öffnenden und/oder schließenden Klammern umgekehrt werden. Wird auf diese Werte 10 aufaddiert (also 10 bis 13 angegeben), so werden die öffnenden Klammern nicht zum eingeklammerten Text gerechnet; wird 20 addiert, so werden die schließenden Klammern nicht zum eingeklammerten Text gerechnet; wird 30 addiert, so werden beide Klammern nicht zum eingeklammerten Text gerechnet.

<b>Nn</b>	Zeichenfolgen, mit denen Nicht-Sortierwörter für den n-ten Sortierschlüssel gekennzeichnet sind. Im jeweiligen Sortierschlüssel werden diese Kennung und die nachfolgenden Zeichen bis einschließlich dem nachfolgenden Leerzeichen bzw. Apostroph jeweils eliminiert. [ IX ]
<b>DEZ</b>	Anzahl der Stellen, auf die Zahlen im Sortierschlüssel mit führenden Nullen aufgefüllt werden sollen. Zahlen, die schon aus entsprechend vielen oder mehr Ziffern bestehen, bleiben unverändert. [ I ] <1, 1, 1, 1, 1, 1, 1, 1, 1>
<b>Rn</b>	Zeichenfolgen, die römische Zahlen kennzeichnen, die für den n-ten Sortierschlüssel in eine Folge von 4 arabischen Ziffern umgewandelt werden sollen. Die Zeichenfolgen müssen jeweils unmittelbar vor der römischen Zahl stehen. [ IX ]
<b>XSn</b>	Zeichenfolgenpaare (und Ausnahmezeichenfolgen), deren jeweils erste Zeichenfolge durch die jeweils zweite Zeichenfolge im n-ten Sortierschlüssel ersetzt werden soll. [ X ]
<b>An</b>	Sortieralphabet für den n-ten Sortierschlüssel. [ VII ]
<b>SSL</b>	Sortierschlüssellängen. [ I ] <0, 0, 0, 0, 0, 0, 0, 0, 0>
<b>RLF</b>	Angabe, ob die Sortierschlüssel rückläufig aufgebaut werden sollen. [ I ] <0, 0, 0, 0, 0, 0, 0, 0, 0>

Es kann für jeden Sortierschlüssel ein Zahlenwert angegeben werden:

- 0 = Sortierschlüssel normal aufbauen
- 1 = Sortierschlüssel rückläufig aufbauen

Die Sortierschlüssel 4 bis 6 für den Kontext (nur bei MODUS=KWIC) können nur entweder alle normal oder alle rückläufig aufgebaut werden. Werden sie normal aufgebaut, wird der Teil vom Kontext als Sortiertext genommen, der rechts vom Schlüsselwort steht; werden sie rückläufig aufgebaut, wird der Teil vom Kontext als Sortiertext genommen, der links vom Schlüsselwort steht.

## Alphabetisches Verzeichnis der Parameter

Das Zeichen »n« in den Kennungen der Parameter steht für die Ziffern 1 bis 9 (z. B. steht Rn für R1, ..., R9).

((	Ausklammern von Teilen der Registereinträge . . . . .	1051
(Sn	Ausklammern von Textteilen im Sortierschlüssel . . . . .	1068
(SW	Ausklammern von Textteilen ohne Schlüsselwörter . . . . .	1053
))	Ausklammern von Teilen der Registereinträge . . . . .	1051
)Sn	Ausklammern von Textteilen im Sortierschlüssel . . . . .	1068
)SW	Ausklammern von Textteilen ohne Schlüsselwörter . . . . .	1053

<b>A</b>	Anfang der Textteile, die Registereinträge enthalten . . . . .	1049
<b>An</b>	Sortieralphabet . . . . .	1069
<b>AA</b>	Anfang einer Texteinheit (Abschnittsanfang) . . . . .	1047
<b>AAZ</b>	Zusatzangaben zum Parameter AA . . . . .	1047
<b>AE</b>	Ende einer Texteinheit (Abschnittsende) . . . . .	1047
<b>AEZ</b>	Zusatzangaben zum Parameter AE . . . . .	1048
<b>AES</b>	Index für AS <sub>n</sub> und ES <sub>n</sub> . . . . .	1067
<b>ALZ</b>	Bilden einer Texteinheit (Abschnitt) bei Leerzeilen . . . . .	1047
<b>ANR</b>	Bilden einer Texteinheit (Abschnitt) nach Nummern . . . . .	1046
<b>AS<sub>n</sub></b>	Anfangskennung für Sortierschlüssel . . . . .	1067
<b>ASW</b>	Anfang der Textteile, die Schlüsselwörter enthalten . . . . .	1052
<b>BER</b>	Auswahl eines Bereichs aus der QUELL-Datei . . . . .	1045
<b>BIS</b>	Kennzeichen für das Ende eines Referenzbereichs . . . . .	1058
<b>DEZ</b>	Dezimalstellen für Zahlen im Sortierschlüssel . . . . .	1069
<b>E</b>	Ende der Textteile, die Registereinträge enthalten . . . . .	1049
<b>EA</b>	(Register-) Eintragsanfang . . . . .	1050
<b>EE</b>	(Register-) Eintragsende . . . . .	1050
<b>ENA</b>	Ergänzung nach Registereinträgen: Anfangskennung . . . . .	1063
<b>ENB</b>	Ergänzung nach Registereinträgen: Beibehalten/löschen . . . . .	1064
<b>ENE</b>	Ergänzung nach Registereinträgen: Endekennung . . . . .	1063
<b>ENK</b>	Ergänzung nach Registereinträgen: Konstanter Text . . . . .	1063
<b>ENN</b>	Ergänzung nach Registereinträgen: Nummern zu ENT . . . . .	1064
<b>ENT</b>	Ergänzung nach Registereinträgen: Trennzeichenfolgen . . . . .	1064
<b>ENX</b>	Ergänzung nach Registereinträgen: Ersetzen von Zf. . . . .	1064
<b>ENZ</b>	Ergänzung nach Registereinträgen: Zusatz zu ENA . . . . .	1064
<b>ES<sub>n</sub></b>	Endekennung für Sortierschlüssel . . . . .	1067
<b>ESW</b>	Ende der Textteile, die Schlüsselwörter enthalten . . . . .	1052
<b>EVA</b>	Ergänzung vor Registereinträgen: Anfangskennung . . . . .	1063
<b>EVB</b>	Ergänzung vor Registereinträgen: Beibehalten/löschen . . . . .	1064
<b>EVE</b>	Ergänzung vor Registereinträgen: Endekennung . . . . .	1063
<b>EVK</b>	Ergänzung vor Registereinträgen: Konstanter Text . . . . .	1063
<b>EVN</b>	Ergänzung vor Registereinträgen: Nummern zu EVT . . . . .	1064
<b>EVT</b>	Ergänzung vor Registereinträgen: Trennzeichenfolgen . . . . .	1064
<b>EVX</b>	Ergänzung vor Registereinträgen: Ersetzen von Zf. . . . .	1064
<b>EVZ</b>	Ergänzung vor Registereinträgen: Zusatz zu EVA . . . . .	1064
<b>HFA</b>	Häufigkeitsangabe: Anfangskennung . . . . .	1059
<b>HFE</b>	Häufigkeitsangabe: Endekennung . . . . .	1059
<b>HFL</b>	Häufigkeitsangabe: Länge (Dezimalstellen) . . . . .	1059
<b>IRL</b>	Interne Referenzlänge . . . . .	1057
<b>KLS</b>	Index für (S <sub>n</sub> und ) S <sub>n</sub> . . . . .	1068
<b>MAX</b>	Maximum für Testzwecke . . . . .	1046
<b>MF</b>	Kennzeichen für eine Referenz mit f . . . . .	1058
<b>MFF</b>	Kennzeichen für eine Referenz mit ff . . . . .	1058
<b>MTL</b>	Max. Länge eines Sortiertextes (Textlänge) . . . . .	1063
<b>Nn</b>	Kennzeichen für Nicht-Sortierwörter . . . . .	1069
<b>NSN</b>	Kennzeichen für neue Sortiernummer . . . . .	1065
<b>OHF</b>	Kennzeichen für Registereinträge ohne Häufigkeit . . . . .	1059
<b>ORF</b>	Kennzeichen für Registereinträge ohne Referenz . . . . .	1058

<b>PAR</b>	Parameter-Konvention . . . . .	1045
<b>Rn</b>	Kennzeichen für römische Zahlen im Sortierschlüssel . . . . .	1069
<b>RFA</b>	Referenz: Anfangskennung . . . . .	1054
<b>RFB</b>	Referenz: Beibehalten/löschen . . . . .	1055
<b>RFE</b>	Referenz: Endekennung . . . . .	1054
<b>RFG</b>	Referenz: Gültigkeit . . . . .	1056
<b>RFL</b>	Referenz: Länge der einzelnen Teile . . . . .	1055
<b>RFM</b>	Referenz: Markierung . . . . .	1058
<b>RFN</b>	Referenz: Nummern zu RFT . . . . .	1055
<b>RFR</b>	Referenz: Reihenfolge . . . . .	1056
<b>RFS</b>	Referenz: Sortierung . . . . .	1056
<b>RFT</b>	Referenz: Trennzeichenfolgen zwischen den Teilen . . . . .	1055
<b>RFX</b>	Referenz: Zeichenfolgen ersetzen . . . . .	1054
<b>RFZ</b>	Referenz: Zusatz zu RFA . . . . .	1055
<b>RLF</b>	Rückläufig sortieren . . . . .	1069
<b>RSK</b>	Referenz: Sortierkennungen . . . . .	1067
<b>SNL</b>	Länge der Sortiernummer . . . . .	1065
<b>SSL</b>	Länge der Sortierschlüssel . . . . .	1069
<b>STE</b>	Silbentrennzeichen-Ersatz . . . . .	1049
<b>STR</b>	Silbentrennung aufheben . . . . .	1048
<b>SWT</b>	Trennzeichenfolgen zwischen Schlüsselwörter . . . . .	1053
<b>T+</b>	Positiv-Auswahl über ganze Registereinträge . . . . .	1061
<b>T+N</b>	Positiv-Auswahl über ganze Registereinträge . . . . .	1061
<b>T+U</b>	Positiv-Auswahl über ganze Registereinträge . . . . .	1061
<b>T-</b>	Negativ-Auswahl über ganze Registereinträge . . . . .	1062
<b>T-N</b>	Negativ-Auswahl über ganze Registereinträge . . . . .	1062
<b>T-U</b>	Negativ-Auswahl über ganze Registereinträge . . . . .	1062
<b>TA+</b>	Positiv-Auswahl über Anfänge von Registereinträgen . . . . .	1062
<b>TA-</b>	Negativ-Auswahl über Anfänge von Registereinträgen . . . . .	1062
<b>TE+</b>	Positiv-Auswahl über Enden von Registereinträgen . . . . .	1062
<b>TE-</b>	Negativ-Auswahl über Enden von Registereinträgen . . . . .	1062
<b>TK</b>	Kennung der Typenkennzeichen . . . . .	1051
<b>TKN</b>	Typenkennnummern für die Typenkennzeichen . . . . .	1052
<b>TKZ</b>	Typenkennzeichen . . . . .	1051
<b>TR</b>	Trennzeichenfolgen zwischen Registereinträgen . . . . .	1050
<b>TYP</b>	Typ der Registereinträge . . . . .	1051
<b>UMD</b>	Umdrehen von Registereinträgen . . . . .	1060
<b>UME</b>	Ergänzung am Umdrehpunkt in einem Registereintrag . . . . .	1061
<b>UMP</b>	Umdrehpunkt in einem Registereintrag . . . . .	1060
<b>VON</b>	Kennzeichen für den Anfang eines Referenzbereichs . . . . .	1058
<b>X</b>	Ersetzen von Zeichenfolgen bei der Eingabe . . . . .	1049
<b>XX</b>	Ersetzen von Zeichenfolgen in Registereinträgen . . . . .	1051
<b>XSn</b>	Ersetzen von Zeichenfolgen im Sortierschlüssel . . . . .	1069
<b>ZD</b>	Bestimmen der ZIEL-Datei . . . . .	1065
<b>ZDU</b>	ZIEL-Datei bei undefinierten Typenkennzeichen . . . . .	1066
<b>ZF+</b>	Positiv-Auswahl über enthaltene Zeichenfolgen . . . . .	1062
<b>ZF-</b>	Negativ-Auswahl über enthaltene Zeichenfolgen . . . . .	1062

## Weiterverarbeitung der Daten

Nachdem die Daten mit diesem Programm zum Sortieren vorbereitet sind, müssen sie mit dem Kommando #SORTIERE sortiert werden. Wurden mehrere ZIEL-Dateien angegeben, so müssen diese einzeln sortiert werden. Falls die Einträge für ein Register in mehrere ZIEL-Dateien ausgegeben wurden, müssen die zu einem Register gehörenden Dateien mit dem Kommando #MISCHE in eine Datei zusammengemischt werden.

Nach dem Sortieren bzw. Mischen können die Registereinträge bzw. die Schlüsselwörter und der dazugehörige Kontext mit dem Kommando #RAUFBEREITE zur Ausgabe aufbereitet werden.

Wurden die Sortierschlüssel und die eigentlichen Registereinträge auf verschiedene Dateien ausgegeben, so müssen diese wieder zusammengeführt werden. Dies geschieht dadurch, dass die zur Spezifikation DATEN angegebene Datei (bzw. -STD- für die Standard-Daten-Datei) auch beim Aufruf des Kommandos #RAUFBEREITE zur Spezifikation DATEN angegeben wird.

## Aufbau eines Datensatzes

REF	TYP	STB	SN	SS	HF	Text ...
-----	-----	-----	----	----	----	----------

```

MODUS = -                +---+ +-----+ +---+ =====
MODUS = +  *****  *****  *****  +---+ +-----+ +---+ =====

```

- REF Referenz (normalerweise 14 Zeichen)
- TYP Typ (1 Zeichen)
- STB Steuerbits (1 Zeichen)
- SN Sortiernummer
- SS Sortierschlüssel
- HF Häufigkeit

Bei der Sortiernummer, dem Sortierschlüssel und der Häufigkeit ist die Länge abhängig von den Angaben zu den Parametern SNL, HFL, SSL; bei der Referenz kann die Standardlänge mit dem Parameter IRL geändert werden.

- === Eingabedaten bzw. bei MODUS=KWIC  
Pointer auf Schlüsselwort (10 Zeichen)
- \*\*\* Daten, die hinzugefügt werden
- +++ Daten, die bei Angabe entsprechender  
Parameter hinzugefügt werden.

\*\*\*\*\*



**#SPRUEFE**

Kommando . . . . .	1075
Leistung . . . . .	1075
Beispiel . . . . .	1076
Arbeitsweise des Programms . . . . .	1077
Parameter . . . . .	1079

## Kommando

#SPRUEFE

QUELLE	= <code>datei</code>	Name der Datei mit den sortierten Daten.
	= <code>-STD-</code>	Die sortierten Daten stehen in der Standard-Text-Datei.
ZIEL	= <code>datei</code>	Name der Datei für die möglicherweise falsch sortierten Daten.
	= <code>-STD-</code>	Die möglicherweise falsch sortierten Daten in die Standard-Text-Datei ausgeben.
MODUS	= <code>-</code>	Daten enthalten keine Referenzen.
	= <code>+</code>	* Daten enthalten Referenzen.
	= <code>R</code>	Daten enthalten Referenzen, für die auch Sortierschlüssel vorhanden sind.
	= <code>K</code>	Daten sind Korrekturanweisungen.
LOESCHEN	= <code>-</code>	* Daten in der ZIEL- und in der PROTOKOLL-Datei nicht löschen.
	= <code>+</code>	Daten in der ZIEL- und in der PROTOKOLL-Datei zuerst löschen.
PARAMETER	= <code>datei</code>	Name der Datei mit den Parametern.
	= <code>*</code>	Die Parameter folgen auf das Kommando und sind mit <code>*EOF</code> abgeschlossen.
DATEN	= <code>-</code>	* Daten stehen vollständig in der QUELL-Datei.
	= <code>datei</code>	Name der Datei mit den zum Sortieren nicht erforderlichen Datenteilen.
	= <code>-STD-</code>	Die zum Sortieren nicht erforderlichen Datenteile stehen in der Standard-Daten-Datei.
PROTOKOLL	= <code>-</code>	* Kein Testprotokoll erstellen.
	= <code>+</code>	Testprotokoll ins Ablaufprotokoll ausgeben.
	= <code>-STD-</code>	Testprotokoll in die Standard-Protokoll-Datei ausgeben.
	= <code>datei</code>	Name der Datei für das Testprotokoll.

## Leistung

Mit diesem Programm kann nach dem Sortieren geprüft werden, ob die einzelnen Sortierschlüssel lang genug sind, um die gewünschte Reihenfolge der einzelnen Sätze zu erreichen.

## Beispiel

Die Daten stehen in der Datei d und bestehen aus einzelnen Einheiten, die alle das folgende Format haben:

```
@n laufende Nummer
@a Autor1; Autor2
@t Titel
@x ...
...
```

Es soll ein Autorenregister erstellt werden. Dabei soll geprüft werden, ob die einzelnen Sortierschlüssel lang genug sind.

```
#RV, d, -STD-, +, +, *
AA      /@n /
RFA     /@n /
RFE     / /
EA      /@a /
EE      /@t /
TR      /; /
XS1     /ä/ae/ö/oe/ü/ue/ß/ss/{1--2}{%} //
XS2     /ä/az/ö/oz/ü/uz/ß/sz/
A2      {|}%
SSL     20 20
*EOF
```

```
#SO, -STD-, -STD-, 17+40, +
```

```
#SP, -STD-, , +, +, *
XS1     /ä/ae/ö/oe/ü/ue/ß/ss/{1--2}{%} //
XS2     /ä/az/ö/oz/ü/uz/ß/sz/
A2      {|}%
SSL     20 20
*EOF
```

```
#RA, -STD-, , +, +, *
SSL     20 20
DRT     WIN-10
*EOF
```

```
#DR, , WIN-10, +
```

## Arbeitsweise des Programms

Für jede Sortiereinheit wird der Sortiertext gebildet und daraus werden die einzelnen Sortierschlüssel erstellt. Die Sortierschlüssel werden dabei nicht auf die mit dem Parameter `SSL` angegebene Zeichenzahl begrenzt.

Für jede Sortiereinheit wird geprüft,

- ob die erstellten Sortierschlüssel mit denen in den sortierten Daten übereinstimmen, wobei jeweils nur die mit dem Parameter `SSL` angegebene Zeichenzahl berücksichtigt wird. Stimmt ein Sortierschlüssel nicht überein, wird das Programm mit einer entsprechenden Fehlermeldung abgebrochen.
- ob die Reihenfolge der jeweils aktuellen Sortiereinheit und der vorangehenden Sortiereinheit den Sortierschlüsseln entspricht (d. h. ob die Daten nach den Sortierschlüsseln sortiert sind).
- wie lang die einzelnen Sortierschlüssel sein müssen. Dazu werden die vollständigen Sortierschlüssel (die mit dem Parameter `SSL` angegebenen Zeichenzahl bleibt unberücksichtigt) der jeweils aktuellen Sortiereinheit einzeln mit denen der vorangehenden Sortiereinheit verglichen. Stimmen zwei Sortierschlüssel überein, werden jeweils auch die beiden nachfolgenden miteinander verglichen. Stimmen zwei Sortierschlüssel nicht überein, wird für den jeweiligen Sortierschlüssel gemerkt, wie lang er mindestens sein muss, damit sich die zwei unterscheiden; die restlichen Sortierschlüssel werden in diesem Fall nicht mehr verglichen.
- wie lang die einzelnen Sortierschlüssel maximal werden, wenn sie nicht auf die mit dem Parameter `SSL` angegebene Zeichenzahl begrenzt werden.

Wenn alle Sortiereinheiten abgearbeitet sind, wird für jeden Sortierschlüssel aufgelistet, wie lang er ist (Wert von Parameter `SSL`), wie lang er mindestens sein muss, damit die Daten dem vollständigen Sortierschlüssel entsprechend sortiert werden können, und wie lang er maximal wird.

Sind alle Sortierschlüssel lang genug, stimmen diese Angaben. Ist aber ein Sortierschlüssel zu kurz, sind die Daten möglicherweise nicht richtig sortiert. Dies hat zur Folge, dass die Angabe zur Mindestlänge eventuell zu klein angegeben ist, da jeweils nur die Sortierschlüssel von zwei unmittelbar aufeinander folgenden Sortiereinheiten miteinander verglichen werden.

Dies soll am folgenden Beispiel mit einem einzigen, aus 15 Zeichen bestehenden Sortierschlüssel erläutert werden. Der Einfachheit halber wird angenommen, dass keine Parameter angegeben sind, die beim Erstellen des Sortierschlüssels aus dem Sortiertext Veränderungen vornehmen, so dass der ungekürzte Sortierschlüssel jeweils mit dem Sortiertext übereinstimmt.

Sortierschlüssel	Sortiertext
schmidberger, p	Schmidberger, Petra
schmidberger, p	Schmidberger, Paul
schmidberger, p	Schmidberger, Peter
123456789012345	12345678901234567890

Die Daten sind nach dem Sortierschlüssel sortiert, aber offensichtlich ist der Sortierschlüssel nicht lang genug, um beim Sortieren die richtige Reihenfolge zu erhalten. Das Programm vergleicht die erste mit der zweiten Sortiereinheit sowie die zweite mit der dritten Sortiereinheit und stellt fest, dass 16 Zeichen für die richtige Sortierung notwendig sind. Da die erste und die dritte Sortiereinheit nicht miteinander verglichen werden, bemerkt das Programm nicht, dass tatsächlich 18 Zeichen erforderlich sind. Der längste ungekürzte Sortierschlüssel ist 19 Zeichen lang. Bei Programmende wird folglich angezeigt, dass der Sortierschlüssel 16–19 Zeichen lang sein sollte.

Wird mit einem 16 Zeichen langen Sortierschlüssel sortiert, kann sich folgende Reihenfolge ergeben:

Sortierschlüssel	Sortiertext
schmidberger, pa	Schmidberger, Paul
schmidberger, pe	Schmidberger, Petra
schmidberger, pe	Schmidberger, Peter
1234567890123456	12345678901234567890

Hier stellt das Programm beim Vergleich der jeweils unmittelbar aufeinander folgenden Sortiereinheiten fest, dass 18 Zeichen für die richtige Sortierung notwendig sind, und zeigt an, dass der Sortierschlüssel 18–19 Zeichen lang sein sollte.

## Parameter

Es können die gleichen Parameter wie beim Kommando #SVORBEREITE angegeben werden; ausgenommen davon sind folgende Parameter:

- Auswahl der Daten: BER
- Zusammenfassen von Sätzen: ANR, ALZ, AA, AAZ, AE, AEZ
- Silbentrennung: STR, STE
- Zusammenfassen von Texteinheiten: FS
- Gruppenweises Sortieren: NSN
- Ändern der Daten: TYP, RFM, SW

Außerdem können mit dem Parameter RFL mehrere Werte angegeben werden; sie haben dieselbe Bedeutung wie beim Parameter RFL des Kommandos #RVORBEREITE.

Wurden die Sortierschlüssel mit dem Kommando #SVORBEREITE erstellt, sind alle Parameter erforderlich, die dabei angegeben waren, mit Ausnahme der oben genannten.

Wurden die Sortierschlüssel mit dem Kommando #RVORBEREITE erstellt, sind alle Parameter erforderlich, die zum »Erstellen des Sortierschlüssels« (siehe dort) angegeben waren. Wurden für die Referenz ebenfalls Sortierschlüssel erstellt, müssen auch die Parameter RSK und RSS (siehe Beschreibung des Kommandos #SVORBEREITE) angegeben werden.

Der Parameter IRL muss angegeben werden, wenn damit ein von der Voreinstellung abweichender Wert angegeben war.

\* \* \* \* \*





**#SVORBEREITE**

---

Kommando . . . . .	1083
Leistung . . . . .	1084
Beispiel . . . . .	1085
Allgemeines . . . . .	1086
Die einzelnen Modi und ihre speziellen Parameter . . . . .	1088
MODUS = - . . . . .	1088
MODUS = + . . . . .	1088
MODUS = R . . . . .	1088
MODUS = K . . . . .	1089
MODUS = S . . . . .	1089
Parameter . . . . .	1090
Einstellen der Parameter-Konvention . . . . .	1090
Auswählen der Daten . . . . .	1090
Zusammenfassen mehrerer Sätze zu einer Texteinheit . . . . .	1091
Überprüfen der Länge der Texteinheiten . . . . .	1094
Zusammenfassen von Texteinheiten zu einer Sortiereinheit . . . . .	1094
Gruppenweises Sortieren . . . . .	1094
Erstellen des Sortiertextes . . . . .	1095
Erstellen der Sortierschlüssel . . . . .	1098
Alphabetisches Verzeichnis der Parameter . . . . .	1100
Weiterverarbeitung der Daten . . . . .	1102
Aufbau eines Datensatzes . . . . .	1102
Aufbau des Korrekturschlüssels . . . . .	1103

## Kommando

#SVORBEREITE

QUELLE	= <code>datei</code>	Name der Datei mit den Daten, die zum Sortieren vorbereitet werden sollen.
	= <code>-STD-</code>	Die Daten, die zum Sortieren vorbereitet werden sollen, stehen in der Standard-Text-Datei.
ZIEL	= <code>datei</code>	Name der Datei für die zum Sortieren vorbereiteten Daten.
		Es können bis zu zehn Dateinamen angegeben werden; sie müssen durch Apostroph getrennt sein.
	= <code>-STD-</code>	Die zum Sortieren vorbereiteten Daten in die Standard-Text-Datei ausgeben.
MODUS	= <code>-</code>	Nur Sortierschlüssel (keine Referenzen) hinzufügen.
	= <code>+</code>	* Referenzen und Sortierschlüssel hinzufügen.
	= <code>R</code>	Eingabedaten enthalten schon Referenzen; Sortierschlüssel hinzufügen.
	= <code>K</code>	Eingabedaten sind Korrekturanweisungen; Korrekturschlüssel und ggf. Sortierschlüssel hinzufügen.
	= <code>S</code>	Eingabedaten sind Korrekturanweisungen mit Korrekturschlüssel; Sortierschlüssel hinzufügen.
LOESCHEN	= <code>-</code>	* Daten in der ZIEL-, in der DATEN- und in der PROTOKOLL-Datei nicht löschen.
	= <code>+</code>	Daten in der ZIEL-, in der DATEN- und in der PROTOKOLL-Datei zuerst löschen.
PARAMETER	= <code>datei</code>	Name der Datei mit den Parametern.
	= <code>*</code>	Die Parameter folgen auf das Kommando und sind mit <code>*EOF</code> abgeschlossen.
DATEN	= <code>-</code>	* Daten vollständig in die ZIEL-Datei ausgeben.
	= <code>datei</code>	Name der Datei für die zum Sortieren nicht notwendigen Datenteile.
	= <code>-STD-</code>	Die zum Sortieren nicht notwendigen Datenteile in die Standard-Daten-Datei ausgeben.
PROTOKOLL	= <code>-</code>	* Kein Testprotokoll erstellen.
	= <code>+</code>	Testprotokoll ins Ablaufprotokoll ausgeben.
	= <code>-STD-</code>	Testprotokoll in die Standard-Protokoll-Datei ausgeben.
	= <code>datei</code>	Name der Datei für das Testprotokoll.

## Leistung

Mit diesem Programm können Texteinheiten (bestehend aus einem oder mehreren Eingabesätzen, auch Korrekturanweisungen) zum Sortieren vorbereitet werden. Dabei kann angegeben werden, nach welchen Kriterien sortiert werden soll.

## Beispiel

Die zu sortierenden Daten bestehen aus einzelnen Einheiten, die alle das folgende Format haben:

```
@n laufende Nummer
@a Autor
@t Titel
@x ...
...
```

Die Daten stehen in der Datei `altdat`. Sie werden nach dem Autor und bei gleichem Autor nach dem Titel sortiert. Anschließend werden die Einheiten neu durchnummeriert und nach jeder Einheit wird eine Leerzeile eingefügt. Das Ergebnis wird in die Datei `neudat` ausgegeben.

Die Einheiten werden nach den Regeln für das alphabetische Ordnen in Bibliothekskatalogen sortiert. Bei den Titeln handelt es sich um deutschsprachige Titel. Bestimmte und unbestimmte Artikel werden am Anfang des Titels zum Sortieren eliminiert. Die Behandlung der Satzzeichen und der Sonderzeichen ist in diesem Beispiel nicht berücksichtigt. Akzentbuchstaben werden beim Sortieren jedoch berücksichtigt.

```
#SV,altdat,-STD-,-,+,*
AA      /@n/
AK1     /@a /
EK1     /@t /
AK2     /@t /
EK2     /@x /
XX2     /{[]Der //{}Die //{}Das //{}Ein //{}Eine //
XX2     /{[]Des //{}Dem //{}Den //
XX2     /{[]Eines //{}Einem //{}Einer //{}Einen //
AEI     11 11
STB     1 1
SSZ     2 2
XS1     /ä/ae/ö/oe/ü/ue/ß/ss/{1--2}{%} //
XS2     /ä/az/ö/oz/ü/uz/ß/sz/
A2      {}%
XS3     = XS1
XS4     = XS2
A4      = A2
SSL     30 30 90 90
*EOF
```

```
#SO,-STD-,-STD-,1+240,+,1+240
```

```
#KO,-STD-,neudat,+,+,*
LNR     /@n /
ZA      /@<% /
SA      /@n /
LZV     /@n /
*EOF
```

## Allgemeines

In der Regel werden die Eingabedaten so in Texteinheiten eingeteilt (siehe »Zusammenfassen mehrerer Sätze zu einer Texteinheit«), dass jede Texteinheit eine Sortiereinheit bildet.

Soll nicht einfach nach dem Wortlaut der Texteinheit sortiert werden, sondern nach bestimmten Textteilen dieser Texteinheit, so kann über Auswahlparameter (siehe »Erstellen des Sortiertextes«) angegeben werden, welche Textteile in welcher Reihenfolge berücksichtigt werden sollen. Diese werden in der entsprechenden Reihenfolge, jeweils mit zwei zusätzlichen Leerzeichen dazwischen, zum Sortiertext zusammengestellt. Werden keine Auswahlparameter angegeben, so ist der Sortiertext mit dem Text der Texteinheit identisch.

Aus dem Sortiertext werden ein bis drei Sortierschlüssel aufgebaut. Dies ist deshalb notwendig, weil die Reihenfolge der Zeichen im internen Code des Rechners nicht der üblichen alphabetischen Ordnung entspricht. Die Sortierschlüssel werden nur zum Sortieren benutzt und werden danach wieder eliminiert. Für englische oder lateinische Texte, die weder Umlaute noch Akzente enthalten, reicht 1 Sortierschlüssel aus. In deutschen Texten muss nach DIN 5007 das ß für die Sortierung in ss umgewandelt werden; die Umlaute werden in einigen Wörterbüchern, Lexika etc. wie die entsprechenden Grundbuchstaben eingeordnet, in Namensverzeichnissen (Telefonbücher, Registraturen etc.) wie ae, oe, ue. Diese Gleichsetzungen (Umwandlungen) müssen im 1. Sortierschlüssel vorgenommen werden. Damit dabei Wörter wie »Maße« und »Masse«, »drucken« und »drücken« oder (in Namensverzeichnissen) Namen wie »Jäger« und »Jaeger« auseinander gehalten werden können, wird ein 2. Sortierschlüssel benötigt. Für ihn können ä, ö, ü und ß z. B. in az, oz, uz und sz umcodiert werden. Enthält der Text Akzente, so wird ebenfalls ein 2. Sortierschlüssel benötigt. Üblicherweise werden die Akzente für den 1. Sortierschlüssel eliminiert, damit zuerst nach dem Buchstabenbestand sortiert wird und die Akzente nur bei gleichem Buchstabenbestand die Reihenfolge beeinflussen. Für den 2. Sortierschlüssel können die Akzente z. B. in Dezimalzahlen umcodiert werden, damit sich für Wörter, die sich nur durch die Akzente unterscheiden, die gewünschte Reihenfolge ergibt. Wenn darüber hinaus noch Groß- und Kleinschreibung beim Sortieren von Bedeutung sind, wird ein dritter Sortierschlüssel benötigt. Falls keine Akzente und keine Umlaute vorkommen, kann die Groß- und Kleinschreibung im zweiten Sortierschlüssel berücksichtigt werden.

Setzt sich der Sortiertext aus mehreren Textteilen zusammen, und erfordert mindestens ein Textteil mehr als einen Sortierschlüssel (dies ist z. B. der Fall, wenn der erste Textteil Autorennamen mit Umlauten/Akzenten und der zweite Textteil Titel mit Umlauten/Akzenten enthält), so genügt es in der Regel nicht mehr, die einzelnen Sortierschlüssel jeweils aus dem gesamten Sortiertext aufzubauen. In diesem Fall ist es erforderlich, den Sortiertext in Bereiche einzuteilen. Dabei können ein einzelner Textteil oder mehrere aufeinander folgende Textteile jeweils einen Sortiertextbereich bilden, wobei jedoch zumindest nach jedem Textteil, der mehr als einen Sortierschlüssel erfordert, ein neuer Bereich beginnen muss. Aus jedem dieser Sortiertextbereiche können ein bis drei Sortierschlüssel aufgebaut werden.

Falls die Texteinheiten der Eingabedatei in Gruppen unterteilt sind, ist es möglich, die Texteinheiten jeweils nur innerhalb ihrer Gruppe zu sortieren. Dazu muss am Anfang jeder Gruppe eine eindeutige Kennung stehen. Es kann z. B. jeweils eine Zwischenüberschrift entsprechend gekennzeichnet sein. Die Gruppen werden intern durchnummeriert; die entsprechende Nummer wird jeweils unmittelbar vor dem Sortierschlüssel der einzelnen Texteinheiten eingesetzt. Die Reihenfolge der Gruppen bleibt auf diese Weise unverändert.

Da der Speicher im Programm begrenzt ist, dürfen die Texteinheiten nicht beliebig groß sein. Reicht dieser für extrem große Texteinheiten nicht aus, so kann diese Begrenzung dadurch umgangen werden, dass kleinere Texteinheiten gebildet werden und diese zu einer Sortiereinheit, deren Größe unbeschränkt ist, zusammengefasst werden. Es müssen dazu die Texteinheiten, die zur gleichen Sortiereinheit wie die vorangehende Texteinheit gehören, mit einer entsprechenden Kennzeichnung versehen sein. Die Sortiermerkmale müssen jedoch alle in der jeweils ersten Texteinheit enthalten sein.

Das Kommando #SORTIERE benötigt Speicherplatz zum Zwischenspeichern der zu sortierenden Daten. Bei sehr großen Datenmengen reicht der zur Verfügung stehende Platz unter Umständen nicht aus. Abhilfe kann dadurch geschaffen werden, dass die Sortierschlüssel und die eigentlichen Textdaten auf verschiedene Dateien (ZIEL-Datei und DATEN-Datei) ausgegeben werden. Es genügt dann, die Datei mit den Sortierschlüsseln (ZIEL-Datei) zu sortieren. Sollte der Speicherplatz zum Sortieren trotzdem nicht ausreichen, können die Sortierschlüssel auf mehrere Dateien verteilt werden. Diese müssen einzeln sortiert und dann (mit dem Kommando #MISCHE) zusammengemischt werden. Damit die Sortierschlüssel auf mehrere Dateien verteilt werden, genügt die Angabe mehrerer ZIEL-Dateien, zusätzliche Parameter sind dafür nicht erforderlich.

## Die einzelnen Modi und ihre speziellen Parameter

MODUS = -

Für die Texteinheiten (die jeweils aus mehreren Eingabesätzen bestehen können) werden nur Sortierschlüssel aufgebaut.

MODUS = +

Wie MODUS=-. Zusätzlich werden bei jeder Texteinheit noch eine Referenz (REF), ein Typ (TYP) und Steuerbits (STB) vor dem Sortierschlüssel bzw. der Sortiernummer eingefügt. Die Referenz gibt an, wo die Texteinheit in der Eingabedatei steht. Sie ist im Normalfall 14 Zeichen lang: 6 Zeichen für die Seitennummer, 3 Zeichen für die Zeilennummer, 3 Zeichen für die Unterscheidungsnummer, 2 Leerzeichen (reserviert für Wortnummer; wird von anderen Programmen benutzt). Besteht eine Texteinheit aus mehreren Zeilen, so wird jeweils die Satznummer der ersten Zeile übernommen. Der Typ (1 Zeichen) wird mit dem Parameter TYP festgelegt. Er kann zur Steuerung der Ausgabe mit dem Kommando #RAUFBEREITE benutzt werden. Von den Steuerbits (1 Zeichen) kann nur eines durch den Parameter RFM gesetzt bzw. gelöscht werden; es gibt an, ob bei der Ausgabe mit dem Kommando #RAUFBEREITE die Referenz ausgegeben werden soll (Bit gesetzt) oder nicht (Bit gelöscht).

**TYP** Typ für alle Sortiereinheiten. [ 1 ] <0 = undefiniert>

**RFM** Angabe, ob die Einträge bei der Aufbereitung mit dem Kommando #RAUFBEREITE eine Referenz erhalten sollen. [ 1 ] <1>

0 = Einträge sollen keine Referenz erhalten

1 = Einträge sollen eine Referenz erhalten

**IRL** Anzahl der Zeichen, die für die Referenz (REF) vorgesehen sind. [ 1 ] <14>

MODUS = R

Wie MODUS=+, jedoch enthält jeder Eingabesatz eine (vollständige) Texteinheit (Parameter ANR, AA und AE nicht möglich), und REF/TYP/STB sind schon vorhanden. Typ bzw. Referenz-Bit können mit den Parametern TYP bzw. RFM geändert werden.

Für den Fall, dass die gewünschte Reihenfolge der Referenzen im Register mit der Reihenfolge der Referenzen in den Eingabedaten übereinstimmt, braucht die Referenz im allgemeinen beim Sortieren nicht berücksichtigt zu werden.

Die drei folgenden Parameter sind nur erforderlich, wenn Sortierschlüssel für die Referenz erstellt werden sollen. Diese sind nur dann notwendig, wenn die Referenz nicht unverändert als Sortierfeld (beim Kommando #SORTIERE bzw. #MISCHE) benutzt werden kann, d. h. wenn damit nicht die gewünschte Reihenfolge erreicht wird.

Besteht die Referenz nicht aus einem einzigen Teil, so muss mit dem Parameter RFL



angegeben werden, wie lang die einzelnen Referenzteile sind. Durch die Anzahl der angegebenen Längenangaben wird gleichzeitig die Zahl der Referenzteile festgelegt. Besteht die Referenz aus einem einzigen Teil, so muss dieser Parameter nur angegeben werden, falls die voreingestellte Länge (6 Zeichen) durch einen anderen Wert ersetzt werden soll.

**RFL** Länge der einzelnen Referenzteile. [ 1 ] <6>

Die einzelnen Teile der Referenz werden zum Referenztext lückenlos aneinander gehängt. Falls zum Erstellen der Sortierschlüssel eine Abgrenzung der einzelnen Teile notwendig ist, kann vor jedem Referenzteil mit dem folgenden Parameter eine Kennung eingefügt werden.

**RSK** Textteile, die zur Kennzeichnung der einzelnen Referenzteile im Referenztext eingefügt werden sollen. Der erste angegebene Textteil wird vor dem ersten Referenzteil eingefügt, der zweite vor dem zweiten usw. [ II ]

Dieser Parameter ist nur zum Erstellen des Referenztextes für die mit dem folgenden Parameter angegebenen Sortierschlüssel von Bedeutung. Die Referenz selbst wird dadurch nicht verändert.

**RSS** Anzahl der Sortierschlüssel für die Referenz. [ 1 ] <0>

Wie lang die einzelnen Sortierschlüssel sein sollen, die aus dem Referenztext erstellt werden, muss am Ende des Parameters `SSL` angegeben werden. Die am Anfang stehenden Angaben gelten für die Sortierschlüssel, die aus dem Sortiertext erstellt werden. Insgesamt sind bis zu neun Sortierschlüssel möglich.

MODUS = K

Jeder Eingabesatz enthält eine Korrekturanweisung (z. B. mit dem Kommando `#VERGLEICHE` erzeugt), für die Korrekturschlüssel und ggf. (abhängig vom Parameter `SSL`) Sortierschlüssel aufgebaut werden sollen. Der Korrekturschlüssel ist eine interne Verschlüsselung der Korrekturanweisung ohne den Korrekturtext und ist 44 Zeichen lang. Der Aufbau ist identisch mit dem Korrekturschlüssel, der mit dem Kommando `#VERGLEICHE` (durch den Parameter `SW`) aufgebaut wird. Falls die Korrekturanweisungen ein Versionskennzeichen enthalten (z. B. mit dem Parameter `VKZ` im Kommando `VERGLEICHE` eingesetzt), kann dieses als Referenz übernommen werden (durch Angabe des Parameters `RFL`). Dabei werden wie bei `MODUS=#REF/TYP/STB` ergänzt.

**RFL** Maximale Länge des Versionskennzeichens der Korrekturanweisungen, das als Referenz übernommen werden soll. [ 1 ] <0>

**SW** Sortierwert, der in den Korrekturschlüssel eingesetzt werden soll. [ 1 ] <0>

MODUS = S

Wie `MODUS=K`, jedoch ist der Korrekturschlüssel schon vorhanden (vom Kommando `VERGLEICHE` mit Parameter `SW`). Der Sortierwert kann durch den Parameter `SW` geändert werden.

## Parameter

Jeder Parameter ist einer Parameterart zugeordnet. Sie ist jeweils als römische Zahl in [ ] angegeben. Damit ist festgelegt, in welcher Form die Angaben vom Programm erwartet werden. Die einzelnen Parameterarten sind in der Beschreibung »TUSTEP-Grundlagen« im Kapitel »{}-Parameter« bzw. »<>-Parameter« definiert.

Für manche Parameter gibt es voreingestellte Werte, die angenommen werden, falls die entsprechenden Werte nicht angegeben sind. Diese Voreinstellungen sind in < > angegeben.

Mit bestimmten Parametern können über Anfangs- und/oder Endekennungen sowie über Kennungen, die als öffnende bzw. schließende Klammern dienen, Textteile für die weitere Bearbeitung ausgewählt werden. Die Funktionsweise dieser Parameter ist auch in den »TUSTEP-Grundlagen« am Ende des Kapitels »Parameter« beschrieben.

Außer den im folgenden beschriebenen Parametern sind auch Parameter zur Definition von Zeichen- und Stringgruppen möglich. [ V ]

Es muss mindestens der Parameter SSL angegeben werden.

### Einstellen der Parameter-Konvention

Ob die Parameter nach der »{}-Parameter-Konvention« oder nach der »<>-Parameter-Konvention« interpretiert werden, kann mit dem Kommando #PARAMETER, {} bzw. #PARAMETER, <> eingestellt werden. Darüber hinaus kann die Konvention auch mit folgendem Parameter eingestellt werden.

- PAR**            Parameter-Konvention einstellen.
- { }    Nachfolgende Parameter nach der »{}-Parameter-Konvention« interpretieren.
  - <>    Nachfolgende Parameter nach der »<>-Parameter-Konvention« interpretieren.

Die Einstellung mit dem Parameter PAR hat Vorrang vor der mit dem Kommando #PARAMETER gewählten Einstellung; sie gilt jeweils nur für die nachfolgenden Parameter bis zum nächsten Parameter PAR bzw. bis zum Ende der Parameter dieses Programms.

### Auswählen der Daten

Soll die gesamte Datei bearbeitet werden, braucht keiner der folgenden Parameter angegeben zu werden.

- BER**            Angabe eines Bereichs (»Seite.Zeile–Seite.Zeile«, falls die Datei im Textmodus nummeriert ist; Zeile–Zeile, falls die Datei im Programmmodus nummeriert ist) oder einer Anfangsstelle (»Seite.Zeile« bzw. »Zeile«), falls nicht die ganze Datei verarbeitet werden soll. [ XI ]

Soll ein Segment einer Segment-Datei verarbeitet werden, kann anstelle des Bereichs der Name des Segments angegeben werden.

Dieser Parameter ist nur zugelassen, wenn die Satznummern in der Datei alle aufsteigend sind.

**MAX** Angabe für Testzwecke, wieviele Texteinheiten (= Eingabesätze, falls keiner der Parameter ANR, ALZ, AA oder AE angegeben ist) maximal zum Sortieren vorbereitet werden sollen. [ 1 ] <999999999>

## Zusammenfassen mehrerer Sätze zu einer Texteinheit

Falls jeder Eingabesatz schon eine vollständige Texteinheit (Sortiereinheit) enthält, sollte keiner der folgenden Parameter angegeben werden. Andernfalls können mit den Parametern ANR, ALZ, AA und/oder AE jeweils mehrere Sätze zu einer Texteinheit zusammengefasst werden.

Ist einer der Parameter ANR, ALZ, AA und AE angegeben, werden vor der Auswertung dieser Parameter evtl. am Anfang und am Ende des Eingabesatzes stehende Leerzeichen entfernt.

Beim Zusammenfassen wird zwischen den einzelnen Eingabesätzen je ein Leerzeichen ergänzt, jedoch nicht an den Stellen, an denen eine Silbentrennung aufgehoben wird (siehe Parameter STR).

**ANR** Angaben, ob aufeinander folgende Sätze, deren Satznummern teilweise oder vollständig übereinstimmen, zu einer Texteinheit zusammengefasst werden sollen. (Abschnitt auf Grund der Nummerierung). [ 1 ] <0>

Es kann ein Zahlenwert angegeben werden:

- 0 = kein Zusammenfassen von Sätzen auf Grund der Satznummer
- 1 = alle aufeinander folgenden Sätze mit der gleichen Seitennummer zu einer Texteinheit zusammenfassen.
- 2 = alle aufeinander folgenden Sätze mit der gleichen Seiten- und der gleichen Zeilennummer (ohne Berücksichtigung der Unterscheidungsnummer) zu einer Texteinheit zusammenfassen.
- 3 = alle aufeinander folgenden Sätze mit der gleichen Satznummer zu einer Texteinheit zusammenfassen.

Ist 0 angegeben (Voreinstellung), so wird die Zusammenfassung nur auf Grund der Parameter ALZ, AE und EE vorgenommen; ist einer der Werte 1 bis 3 angegeben, so ist eine weitere Unterteilung der so gebildeten Texteinheiten auf Grund der genannten Parameter möglich.

**ALZ** Angaben, ob Leerzeilen als Kennzeichen für den Anfang bzw. für das Ende einer Texteinheit dienen sollen. [ 1 ]

Es können zwei Zahlenwerte angegeben werden:

- 1. Zahl: Anfang einer Texteinheit <0>

- 0 = Leerzeilen sind kein Kennzeichen für den Anfang einer Texteinheit.
- 1 = Leerzeilen kennzeichnen den Anfang einer Texteinheit. Folgen mehrere Leerzeilen unmittelbar aufeinander, so beginnt mit jeder einzelnen Leerzeile eine Texteinheit.

## 2. Zahl: Ende einer Texteinheit <0>

- 0 = Leerzeilen sind kein Kennzeichen für das Ende einer Texteinheit.
- 1 = Leerzeilen kennzeichnen das Ende einer Texteinheit. Folgen mehrere Leerzeilen unmittelbar aufeinander, so endet mit jeder einzelnen Leerzeile eine Texteinheit.

Ist jeweils 0 angegeben (Voreinstellung), so wird die Zusammenfassung nur auf Grund der Parameter ANR, AA und AE vorgenommen; andernfalls ist eine weitere Unterteilung der so gebildeten Texteinheiten auf Grund der genannten Parameter möglich.

**AA** Zeichenfolgen, die am Anfang des Eingabesatzes den Anfang einer Texteinheit (Abschnittsanfang) kennzeichnen. [ VIII-a ]

Falls mit dem Parameter AAZ nichts anderes angegeben ist, werden vor der Überprüfung, ob ein Satz mit einer der angegebenen Zeichenfolgen beginnt, führende Leerzeichen entfernt.

Beginnen mit dem Parameter AA angegebene Zeichenfolgen mit Leerzeichen, so können diese nur dann einen Abschnittsanfang kennzeichnen, wenn mit dem Parameter AAZ angegeben wird, dass die führenden Leerzeichen erst nach der Überprüfung entfernt werden sollen; andernfalls ist die Angabe solcher Zeichenfolgen wirkungslos.

**AAZ** Zusatzangaben zum Parameter AA. [ 1 ] <0>

Es kann ein Zahlenwert angegeben werden:

- 0 = Leerzeichen am Anfang von Eingabesätzen vor der Auswertung des Parameters AA entfernen.
- 1 = Leerzeichen am Anfang von Eingabesätzen nach der Auswertung des Parameters AA entfernen.
- 2 = wie 0, jedoch zusätzlich auch die Zeichenfolgen »#[09]« und »#[0009]« entfernen.

**AE** Zeichenfolgen, die am Ende des Eingabesatzes das Ende einer Texteinheit (Abschnittsende) kennzeichnen. [ VIII-b ]

Falls mit dem Parameter AEZ nichts anderes angegeben ist, werden vor der Überprüfung, ob ein Satz mit einer der angegebenen Zeichenfolgen endet, abschließende Leerzeichen entfernt.

Enden mit dem Parameter AE angegebene Zeichenfolgen mit Leerzeichen, so können diese nur dann ein Abschnittsende kennzeichnen, wenn mit dem Parameter AEZ angegeben wird, dass die abschließenden Leerzeichen erst nach der Überprüfung entfernt werden sollen; andernfalls ist die Angabe solcher Zeichenfolgen wirkungslos.

- AEZ** Zusatzangaben zum Parameter AE. [ I ] <0>  
 Es kann ein Zahlenwert angegeben werden:
- 0 = Leerzeichen am Ende von Eingabesätzen vor der Auswertung des Parameters AE entfernen.
  - 1 = Leerzeichen am Ende von Eingabesätzen nach der Auswertung des Parameters AE entfernen.
- STR** Angabe zur Silbentrennung. [ I ] <0>  
 Es kann ein Zahlenwert angegeben werden:
- 0 = Silbentrennungen nicht aufheben.
  - 1 = Silbentrennung bei der Eingabe aufheben; dabei wird angenommen, dass im nachfolgenden Eingabesatz alle Zeichen bis zum ersten Leerzeichen zum getrennten Wort gehören.
  - 2 = wie 1, jedoch Silbentrennung bei der Eingabe nur aufheben, falls entweder der letzte Buchstabe vor und der erste Buchstabe nach dem Silbentrennzeichen (im nachfolgenden Eingabesatz) Kleinbuchstaben sind oder der letzte Buchstabe vor und die ersten beiden Buchstaben nach dem Silbentrennzeichen (im nachfolgenden Eingabesatz) Großbuchstaben sind.
  - 3 = wie 2, jedoch wird angenommen, dass im nachfolgenden Eingabesatz alle Zeichen bis zum ersten Leerzeichen bzw. bis zum ersten öffnenden spitzen Klammer zum getrennten Wort gehören.
  - 4 = wie 2, jedoch wird angenommen, dass im nachfolgenden Eingabesatz alle Zeichen bis zum ersten Leerzeichen, das außerhalb von spitzen Klammern steht, zum getrennten Wort gehören.
- Falls angegeben ist, dass die Silbentrennung bei der Eingabe aufgehoben werden soll, werden in allen Eingabesätzen führende und abschließende Leerzeichen entfernt.
- Als Silbentrennzeichen gilt ein »-«, das (nachdem abschließende Leerzeichen entfernt wurden) als letztes Zeichen in einem Eingabesatz steht, wenn das zweitletzte Zeichen ebenfalls ein »-« ist, oder wenn das zweitletzte Zeichen ein Buchstabe und das drittletzte Zeichen kein Steuerzeichen (\$, &, @, \, \_, #, %) ist.
- Beim Zusammenfassen mehrerer Eingabesätze zu einer Texteinheit werden Silbentrennungen nur innerhalb einer Texteinheit aufgehoben. Steht ein Silbentrennzeichen am Ende des letzten Eingabesatzes einer Texteinheit, wird an dieser Stelle die Silbentrennung nicht aufgehoben.
- Beim Aufheben der Silbentrennung wird ein getrenntes »ck«, das als »k-« und »k« geschrieben ist, nicht wieder zu »ck«.
- STE** Zeichenfolge, die beim Aufheben einer Silbentrennung anstelle des Silbentrennzeichens eingefügt werden soll. [ II ]

## Überprüfen der Länge der Texteinheiten

- MTL** Maximale Länge einer Texteinheit. [ I ] <999999>  
Ist eine Texteinheit länger als angegeben, so wird sie mit einer entsprechenden Fehlermeldung ausgegeben.

## Zusammenfassen von Texteinheiten zu einer Sortiereinheit

- FS** Zeichenfolgen, die zur Kennzeichnung von Fortsetzungseinheiten bzw. Sortiereinheiten dienen. Welche Texteinheiten gekennzeichnet sind, wird mit dem nachfolgend beschriebenen Parameter FSZ festgelegt. [ VIII-a ]

Die Parameter FS ist nur bei MODUS=- erlaubt. Außerdem ist die Angabe einer Datei (bzw. -STD- für die Standard-Daten-Datei) zur Spezifikation DATEN obligat.

- FSZ** Zusatzangaben zum Parameter FS. [ I ] <0>

Es kann ein Zahlenwert angegeben werden:

- 0 = Beginnt eine Texteinheit mit einer mit dem Parameter FS angegebenen Zeichenfolge, so bildet diese Texteinheit keine eigene Sortiereinheit, sondern wird als Fortsetzung der vorangehenden Sortiereinheit behandelt.  
1 = Beginnt eine Texteinheit mit einer mit dem Parameter FS angegebenen Zeichenfolge, so bildet diese Texteinheit eine Sortiereinheit; alle anderen Texteinheiten werden als Fortsetzung der vorangehenden Sortiereinheit behandelt.

## Gruppenweise sortieren

- NSN** Zeichenfolgen, die am Anfang einer Texteinheit den Beginn einer neuen Gruppe von Texteinheiten kennzeichnen, also die Stelle, ab der die Texteinheiten die nächste Sortiernummer erhalten. [ VIII-a ]

Die Anzahl der Stellen, die für die Sortiernummer vorgesehen werden sollen, muss mit dem Parameter SNL angegeben werden.

- SNL** Sortiernummerlänge (maximale Stellenzahl). [ I ] <0>

## Erstellen des Sortiertextes

- AK<sub>n</sub>** Zeichenfolgen, die den Anfang des  $n$ -ten ( $n=1, 2, \dots, 9$ ) Teils des Sortiertextes kennzeichnen. [ IX ]
- EK<sub>n</sub>** Zeichenfolgen, die das Ende des  $n$ -ten ( $n=1, 2, \dots, 9$ ) Teils des Sortiertextes kennzeichnen. [ IX ]
- AEI** Index für die Parameter AK<sub>n</sub> und EK<sub>n</sub>. [ I ]  $\langle 1, 1, 1, 1, 1, 1, 1, 1, 1 \rangle$

Es können ein bis neun Zahlenwerte angegeben werden:

- 1 = Es wird der erste mit A/E gekennzeichnete Textteil (er beginnt mit einer Anfangskennung und endet vor der nachfolgenden Endekennung bzw. am Ende der Texteinheit) ausgewählt.  
Ist nur A angegeben, so endet der ausgewählte Textteil am Ende der Texteinheit; ist nur E angegeben, so beginnt der ausgewählte Textteil am Anfang der Texteinheit.
- 0 = Es wird der Teil der Texteinheit ausgewählt, der bei 1 wegfallen würde.
- 3 = Wie 1, jedoch wird nicht nur der erste, sondern es werden alle mit A/E gekennzeichneten Textteile (der zweite beginnt mit der Anfangskennung, die dem Ende des ersten mit A/E gekennzeichneten Textteils folgt) ausgewählt.  
Ist nur A angegeben, so beginnt der ausgewählte Textteil mit der letzten in der Texteinheit vorkommenden Anfangskennung und endet am Ende der Texteinheit; ist nur E angegeben, so beginnt der ausgewählte Textteil am Anfang der Texteinheit und endet vor der letzten in der Texteinheit vorkommenden Endekennung.
- 2 = Es wird der Teil der Texteinheit ausgewählt, der bei 3 wegfallen würde.

Bei den Werten 0 bis 3 wird die Anfangskennung jeweils zum nachfolgenden Text gerechnet, während die Endekennung nicht mehr zum davor stehenden Text gerechnet wird. Durch Aufaddieren von 10 und/oder 20 kann diese Regelung für die Anfangs- und/oder Endekennung umgekehrt werden. Wird zu den Werten 10 aufaddiert (also 10 bis 13 angegeben), so wird die Anfangskennung jeweils nicht zum nachfolgenden Text gerechnet; wird 20 addiert, so wird die Endekennung jeweils noch zum davor stehenden Text gerechnet; wird 30 addiert, so wird die Anfangskennung nicht zum nachfolgenden Text und die Endekennung zum davor stehenden Text gerechnet.

Bei den Werten 2 und 3 wird die nächste Anfangskennung ab der ersten Position der zuletzt gefundenen Endekennung gesucht, da sie nicht mehr zum davor stehenden Textteil gehört. Anfangs- und Endekennung können sich also im Text überschneiden. Wurde auf diese Werte 20 oder 30 aufaddiert, so wird die nächste Anfangskennung erst ab der Position gesucht, die auf die letzte Position der zuletzt gefundenen Endekennung folgt, da diese noch zum davor stehenden Textteil gehört.

**(Kn** Zeichenfolgen, die bei der Auswahl des  $n$ -ten ( $n=1, 2, \dots, 9$ ) Teils des Sortiertextes (falls  $AK_n$  und/oder  $EK_n$  nicht angegeben) bzw. die bei der Eliminierung von Textteilen aus dem bereits mit  $AK_n$  und/oder  $EK_n$  ausgewählten Teil des Sortiertextes als öffnende Klammer dienen sollen. [ IX ]

Die Wirkung der öffnenden Klammer wird mit dem  $n$ -ten Zahlenwert des Parameters  $KLI$  bestimmt.

**)Kn** Zeichenfolgen, die bei der Auswahl des  $n$ -ten ( $n=1, 2, \dots, 9$ ) Teils des Sortiertextes (falls  $AK_n$  und/oder  $EK_n$  nicht angegeben) bzw. die bei der Eliminierung von Textteilen aus dem bereits mit  $AK_n$  und/oder  $EK_n$  ausgewählten Teil des Sortiertextes als schließende Klammer dienen sollen. [ IX ]

Die Wirkung der schließenden Klammer wird mit dem  $n$ -ten Zahlenwert des Parameters  $KLI$  bestimmt.

**KLI** Index für die Parameter  $(Kn$  und  $)Kn$ . [ I ]  $<0, 0, 0, 0, 0, 0, 0, 0, 0>$

Es können ein bis neun Zahlenwerte angegeben werden:

0 = Es werden die Teile (einschließlich der Klammern) eliminiert, die eingeklammert sind. Fehlende Klammern werden am Anfang bzw. am Ende der Texteinheit bzw. des bereits ausgewählten Textteils logisch ergänzt.

1 = Es werden die Teile ausgewählt, die bei 0 wegfallen würden.

2 = Wie 0, jedoch werden fehlende Klammern nicht ergänzt, sondern die unpaarigen Klammern werden ignoriert.

3 = Es werden die Teile ausgewählt, die bei 2 wegfallen würden.

Bei den Werten 0 bis 3 werden die Klammern jeweils zum eingeklammerten Text gerechnet und werden mit ihm eliminiert bzw. bleiben mit ihm erhalten. Durch Aufaddieren von 10 und/oder 20 kann diese Regelung für die öffnenden und/oder schließenden Klammern umgekehrt werden. Wird auf diese Werte 10 aufaddiert (also 10 bis 13 angegeben), so werden die öffnenden Klammern nicht zum eingeklammerten Text gerechnet; wird 20 addiert, so werden die schließenden Klammern nicht zum eingeklammerten Text gerechnet; wird 30 addiert, so werden beide Klammern nicht zum eingeklammerten Text gerechnet.

**( (n** Zeichenfolgen, die bei der Auswahl des  $n$ -ten ( $n=1, 2, \dots, 9$ ) Teils des Sortiertextes (falls  $AK_n$ ,  $EK_n$ ,  $(Kn$  oder  $)Kn$  nicht angegeben) bzw. die bei der Eliminierung von Textteilen aus dem bereits mit  $AK_n$ ,  $EK_n$ ,  $(Kn$  oder  $)Kn$  ausgewählten Teil des Sortiertextes als öffnende Klammer dienen sollen. [ IX ]

Die Wirkung der öffnenden Klammer wird mit dem  $n$ -ten Zahlenwert des Parameters  $DKI$  bestimmt.



- ) n** Zeichenfolgen, die bei der Auswahl des  $n$ -ten ( $n=1, 2, \dots, 9$ ) Teils des Sortiertextes (falls  $AK_n, EK_n, (K_n$  oder  $)K_n$  nicht angegeben) bzw. die bei der Eliminierung von Textteilen aus dem bereits mit  $AK_n, EK_n, (K_n$  oder  $)K_n$  ausgewählten Teil des Sortiertextes als schließende Klammer dienen sollen. [ IX ]
- Die Wirkung der öffnenden Klammer wird mit dem  $n$ -ten Zahlenwert des Parameters  $DKI$  bestimmt.
- DKI** Index für die Parameter  $(n$  und  $)n$ . [ I ]  $\langle 0, 0, 0, 0, 0, 0, 0, 0, 0 \rangle$
- Angaben analog zum Parameter  $KL_I$  (siehe Seite 1096)
- XXn** Zeichenfolgenpaare (und Ausnahmezeichenfolgen), deren jeweils erste Zeichenfolge durch die jeweils zweite Zeichenfolge im  $n$ -ten ( $n=1, 2, \dots, 9$ ) Teil des Sortiertextes ersetzt werden soll. [ X ]
- EV** Textteile, die im Sortiertext vor den einzelnen Sortiertextteilen ergänzt werden sollen. [ II ]
- EN** Textteile, die im Sortiertext nach den einzelnen Sortiertextteilen ergänzt werden sollen. [ II ]
- ERG** Textteile, die vor dem ersten Teil des Sortiertextes, zwischen den einzelnen Teilen und nach dem letzten Teil des Sortiertextes ergänzt werden sollen. [ II ]  $\langle$ vor dem ersten und nach dem letzten Teil nichts ergänzen, zwischen den einzelnen Teilen je zwei Leerzeichen ergänzen $\rangle$
- Dieser Parameter ist nur erlaubt, wenn keiner der zuvor beschriebenen Parameter  $EV$  und  $EN$  angegeben ist. Außerdem darf er nicht verwendet werden, wenn der Sortiertext in Bereiche eingeteilt wird (d. h. die Parameter  $STB$  und  $SSZ$  angegeben werden). In diesem Fall müssen zum Ergänzen von Textteilen im Sortiertext die Parameter  $EV$  und/oder  $EN$  verwendet werden.
- MLS** Maximallängen für die einzelnen Teile (maximal 9) des Sortiertextes. Die Begrenzung eines Teils auf eine bestimmte Länge ist dann sinnvoll, wenn die jeweils entsprechenden Teile des Sortiertextes sich einerseits bereits nach einer bestimmten Anzahl von Zeichen unterscheiden, andererseits aber so lang sind, dass der Sortiertext unnötig lang wird oder gar so lang wird, dass nachfolgende Teile des Sortiertextes im Sortierschlüssel nicht mehr berücksichtigt werden können, weil der Sortierschlüssel schon von dem überlangen Teil aufgefüllt wird. [ I ]  $\langle 999999, \dots, 999999 \rangle$

## Erstellen der Sortierschlüssel

Aus dem Sortiertext und dem Referenztext (vgl. »MODUS=R« Seite 1088) können insgesamt bis zu neun Sortierschlüssel aufgebaut werden.

Die beiden folgenden Parameter sind nur erforderlich, wenn der Sortiertext in mehrere Bereiche eingeteilt werden soll. Werden diese beiden Parameter nicht angegeben, bildet der gesamte Sortiertext einen einzigen Sortiertextbereich, auch wenn dieser aus mehreren Teilen besteht.

**STB** Größe der Sortiertextbereiche. [ I ]

Es können bis zu neun Zahlenwerte angegeben werden. Die Anzahl der Zahlenwerte ist zugleich die Anzahl der Sortiertextbereiche. Die einzelnen Zahlenwerte bestimmen jeweils die Anzahl der aufeinander folgenden Sortiertextteile (beginnend mit dem ersten), die zu einem Sortiertextbereich zusammengefasst werden sollen. Die Summe aller Zahlenwerte muss mit der Anzahl der Sortiertextteile übereinstimmen, die mit den zu »Erstellen des Sortiertextes« angegebenen Parametern bestimmt wurden.

**SSZ** Anzahl der Sortierschlüssel für die einzelnen Sortiertextbereiche. [ I ]

Parallel zu den Angaben des Parameters **STB** muss für jeden Sortiertextbereich die Anzahl der jeweils aufzubauenden Sortierschlüssel angegeben werden. Insgesamt können bis zu neun Sortierschlüssel aufgebaut werden.

Die folgenden Parameter zum Erstellen der Sortierschlüssel gelten in gleicher Weise für Sortier- und Referenztext. Der besseren Lesbarkeit wegen wird in der Beschreibung jedoch immer nur der Sortiertext genannt. Die Parameter werden in der angegebenen Reihenfolge abgearbeitet. Dies muss z. B. beachtet werden, wenn mit dem Parameter **DEZ** Zahlen mit führenden Nullen ergänzt werden und dann in Such- oder Ausnahmezeichenfolgen des entsprechenden Parameters **xSn** Zahlen angegeben werden.

Die folgenden Auswahlparameter **AS<sub>n</sub>** bis **N<sub>n</sub>** sind nur notwendig, falls nicht der gesamte Sortiertext bzw. der gesamte Text des entsprechenden Sortiertextbereichs im jeweiligen Sortierschlüssel berücksichtigt werden soll.

**AS<sub>n</sub>** Zeichenfolgen, die den Anfang des Teils des Sortiertextes kennzeichnen, der im *n*-ten Sortierschlüssel berücksichtigt werden soll. [ IX ]

**ES<sub>n</sub>** Zeichenfolgen, die das Ende des Teils des Sortiertextes kennzeichnen, der im *n*-ten Sortierschlüssel berücksichtigt werden soll. [ IX ]

**AES** Zusatzangaben für die Parameter **AS<sub>n</sub>** und **ES<sub>n</sub>**. [ I ]

<1, 1, 1, 1, 1, 1, 1, 1, 1>

Angaben analog zum Parameter **AET** (siehe Seite 1095)

**(S<sub>n</sub>** Zeichenfolgen, die bei der Auswahl des Teils des Sortiertextes (falls **AS<sub>n</sub>** und/oder **ES<sub>n</sub>** nicht angegeben) bzw. bei der Eliminierung von Textteilen aus dem bereits mit **AS<sub>n</sub>** und/oder **ES<sub>n</sub>** ausgewählten Teil des

Sortiertextes, der im n-ten ( $n=1, 2, \dots, 9$ ) Sortierschlüssel zu berücksichtigt ist, als öffnende Klammer dienen sollen. [ IX ]

Die Wirkung der öffnenden Klammer wird mit dem n-ten Zahlenwert des Parameters  $KL_S$  bestimmt.

**) S<sub>n</sub>** Zeichenfolgen, die bei der Auswahl des Teils des Sortiertextes (falls  $AS_n$  und/oder  $ES_n$  nicht angegeben) bzw. bei der Eliminierung von Textteilen aus dem bereits mit  $AS_n$  und/oder  $ES_n$  ausgewählten Teil des Sortiertextes, der im n-ten ( $n=1, 2, \dots, 9$ ) Sortierschlüssel zu berücksichtigt ist, als schließende Klammer dienen sollen. [ IX ]

Die Wirkung der öffnenden Klammer wird mit dem n-ten Zahlenwert des Parameters  $KL_S$  bestimmt.

**KL<sub>S</sub>** Index für die Parameter ( $S_n$  und  $) S_n$ . [ I ]  $\langle 0, 0, 0, 0, 0, 0, 0, 0, 0 \rangle$   
Angaben analog zum Parameter  $KL_I$  (siehe Seite 1096)

**N<sub>n</sub>** Zeichenfolgen, mit denen Nicht-Sortierwörter für den n-ten Sortierschlüssel gekennzeichnet sind. Im jeweiligen Sortierschlüssel werden diese Kennung und die nachfolgenden Zeichen bis einschließlich dem nachfolgenden Leerzeichen bzw. Apostroph jeweils eliminiert. [ IX ]

**DEZ** Anzahl der Stellen, auf die Zahlen in den einzelnen Sortierschlüssel mit führenden Nullen aufgefüllt werden sollen. Zahlen, die schon aus entsprechend vielen oder mehr Ziffern bestehen, bleiben unverändert. [ I ]  $\langle 1, 1, 1, 1, 1, 1, 1, 1, 1 \rangle$

**R<sub>n</sub>** Zeichenfolgen, die römische Zahlen kennzeichnen, die für den n-ten Sortierschlüssel in eine Folge von 4 arabischen Ziffern umgewandelt werden sollen. Die Zeichenfolgen müssen jeweils unmittelbar vor der römischen Zahl stehen. [ IX ]

**X<sub>S<sub>n</sub></sub>** Zeichenfolgenpaare (und Ausnahmezeichenfolgen), deren jeweils erste Zeichenfolge durch die jeweils zweite Zeichenfolge im n-ten Sortierschlüssel ersetzt werden soll. [ X ]

**A<sub>n</sub>** Sortieralphabet für den n-ten Sortierschlüssel. [ VII ]

**SSL** Sortierschlüssellängen. [ I ]  $\langle 0, 0, 0, 0, 0, 0, 0, 0, 0 \rangle$

Dieser Parameter ist obligat; für jeden Sortierschlüssel, der aufgebaut werden soll, muss ein Zahlenwert ungleich Null angegeben werden.

**RLF** Angabe, ob die Sortierschlüssel rückläufig aufgebaut werden sollen. [ I ]  $\langle 0, 0, 0, 0, 0, 0, 0, 0, 0 \rangle$

Es kann für jeden Sortierschlüssel ein Zahlenwert angegeben werden:

0 = Sortierschlüssel normal aufbauen

1 = Sortierschlüssel rückläufig aufbauen

## Alphabetisches Verzeichnis der Parameter

Das Zeichen »n« in den Kennungen der Parameter steht für die Ziffern 1 bis 9 (z. B. steht R<sub>n</sub> für R1, ..., R9).

( (n	Ausklammern von Textteilen im Sortiertext . . . . .	1096
(Kn	Ausklammern von Textteilen im Sortiertext . . . . .	1096
(Sn	Ausklammern von Textteilen im Sortierschlüssel . . . . .	1098
) )n	Ausklammern von Textteilen im Sortiertext . . . . .	1097
) Kn	Ausklammern von Textteilen im Sortiertext . . . . .	1096
) Sn	Ausklammern von Textteilen im Sortierschlüssel . . . . .	1099
An	Sortieralphabet . . . . .	1099
AA	Anfang einer Texteinheit (Abschnittsanfang) . . . . .	1092
AAZ	Zusatzangaben zum Parameter AA . . . . .	1092
AE	Ende einer Texteinheit (Abschnittsende) . . . . .	1092
AEZ	Zusatzangaben zum Parameter AE . . . . .	1093
AEI	Index für AK <sub>n</sub> und EK <sub>n</sub> . . . . .	1095
AES	Index für AS <sub>n</sub> und ES <sub>n</sub> . . . . .	1098
AK <sub>n</sub>	Anfangskennung für Sortiertext . . . . .	1095
ALZ	Bilden einer Texteinheit (Abschnitt) bei Leerzeilen . . . . .	1091
ANR	Bilden einer Texteinheit (Abschnitt) nach Nummern . . . . .	1091
AS <sub>n</sub>	Anfangskennung für Sortierschlüssel . . . . .	1098
BER	Auswählen eines Bereichs aus der QUELL-Datei . . . . .	1090
DEZ	Dezimalstellen für Zahlen im Sortierschlüssel . . . . .	1099
DKI	Index für ( (n und ) )n . . . . .	1097
EK <sub>n</sub>	Endekennung für Sortiertext . . . . .	1095
EN	Ergänzungen nach Sortiertextteilen . . . . .	1097
ERG	Ergänzungen für den Sortiertext . . . . .	1097
EV	Ergänzungen vor Sortiertextteilen . . . . .	1097
ES <sub>n</sub>	Endekennung für Sortierschlüssel . . . . .	1098
FS	Kennzeichen für Fortsetzungs-Texteinheiten . . . . .	1094
FSZ	Zusatzangaben zum Parameter FS . . . . .	1094
IRL	Interne Referenzlänge . . . . .	1088
KLI	Index für (Kn und ) Kn . . . . .	1096
KLS	Index für (Sn und ) Sn . . . . .	1099
MAX	Maximum für Testzwecke . . . . .	1091
MLS	Maximallängen für Teile des Sortiertextes . . . . .	1097
MTL	Max. Länge einer Texteinheit (Textlänge) . . . . .	1094
Nn	Kennzeichen für Nicht-Sortierwörter . . . . .	1099
NSN	Kennzeichen für neue Sortiernummer . . . . .	1094
PAR	Parameter-Konvention . . . . .	1090
Rn	Kennzeichen für römische Zahlen im Sortierschlüssel . . . . .	1099
RFL	Länge der einzelnen Referenzteile . . . . .	1089
RFL	Länge des als Referenz zu übernehmenden Kennzeichens . . . . .	1089
RFM	Markierung für Referenz . . . . .	1088
RLF	Rückläufig sortieren . . . . .	1099
RSK	Referenz: Sortierkennungen . . . . .	1089
RSS	Referenz: Anzahl der Sortierschlüssel . . . . .	1089

---

<b>SNL</b>	Länge der Sortiernummer . . . . .	1094
<b>SSL</b>	Länge der Sortierschlüssel . . . . .	1099
<b>SSZ</b>	Anzahl der Sortierschlüssel . . . . .	1098
<b>STB</b>	Größe der Sortiertextbereiche . . . . .	1098
<b>STE</b>	Silbentrennzeichen-Ersatz . . . . .	1093
<b>STR</b>	Silbentrennung aufheben . . . . .	1093
<b>SW</b>	Sortierwert . . . . .	1089
<b>TYP</b>	Typ der Texteinheiten . . . . .	1088
<b>XS<sub>n</sub></b>	Ersetzen von Zeichenfolgen im Sortierschlüssel . . . . .	1099
<b>XX<sub>n</sub></b>	Ersetzen von Zeichenfolgen im Sortiertext . . . . .	1097

## Weiterverarbeitung der Daten

Nachdem die Daten mit diesem Programm zum Sortieren vorbereitet sind, müssen sie mit dem Kommando #SORTIERE sortiert werden. Wurden mehrere ZIEL-Dateien angegeben, so müssen diese einzeln sortiert werden und dann mit dem Kommando #MISCHE in eine Datei zusammengemischt werden.

Nach dem Sortieren bzw. Mischen können die Texteinheiten mit dem Kommando #KOPIERE wieder in die gewohnte Form gebracht oder mit dem Kommando #RAUFBEREITE zur Ausgabe aufbereitet werden.

Wurden die Sortierschlüssel und die eigentlichen Textdaten auf verschiedene Dateien ausgegeben, so müssen diese wieder zusammengeführt werden. Dies geschieht dadurch, dass die zur Spezifikation DATEN angegebene Datei (bzw. -STD- für die Standard-Daten-Datei) auch beim Aufruf des Kommandos #KOPIERE bzw. #RAUFBEREITE zur Spezifikation DATEN angegeben wird.

Werden mehrere Texteinheiten zu einer Sortiereinheit zusammengefasst, so dürfen die Daten nur mit dem Kommando KOPIERE im Modus S weiterverarbeitet werden, falls die zur Spezifikation DATEN angegebene Datei (bzw. -STD- für die Standard-Daten-Datei) nicht schon beim Kommando #SORTIERE angegeben wurde.

## Aufbau eines Datensatzes

REF	TYP	STB	KS	SN	SS	Text ...
-----	-----	-----	----	----	----	----------

```

MODUS = -                ++++++ ++++++++ =====
MODUS = +    *****   *****   *****                ++++++ ++++++++ =====
MODUS = R    =====   =====   =====                ++++++ ++++++++ =====
MODUS = K    ++++++   ++++++   ++++++   *****   ++++++ ++++++++ =====
MODUS = S    ++++++   ++++++   ++++++   =====   ++++++ ++++++++ =====

```

- REF Referenz (normalerweise 14 Zeichen)
- TYP Typ (1 Zeichen)
- STB Steuerbits (1 Zeichen)
- KS Korrekturschlüssel (44 Zeichen)
- SN Sortiernummer
- SS Sortierschlüssel

Bei der Sortiernummer und dem Sortierschlüssel ist die Länge abhängig von den Angaben zu den Parametern SNL und SSL; bei der Referenz kann die Standardlänge mit dem Parameter IRL geändert werden.

- === Eingabedaten
- \*\*\* Daten, die hinzugefügt werden
- +++ Daten, die bei Angabe entsprechender Parameter hinzugefügt werden

## Aufbau des Korrekturschlüssels

Der Korrekturschlüssel ist insgesamt 44 Zeichen lang und ist wie folgt aufgebaut:

APO	EPO	KA	SW	FNR	POS
-----	-----	----	----	-----	-----

APO	1	17	Anfangsposition des Korrekturtextes		
	1	6	Seitennummer		
	7	3	Zeilennummer		
	10	3	Unterscheidungsnummer		
	13	2	Wortnummer		
	15	3	Zeichennummer		
EPO	18	17	Endposition des Korrekturtextes		
	18	6	Seitennummer		
	24	3	Zeilennummer		
	27	3	Unterscheidungsnummer		
	30	2	Wortnummer		
	32	3	Zeichennummer		
KA	35	2	Korrekturart:		
	35	1	0 = Fehler		
			1 = Seite-Zeile-Wort-Zeichen		
			2 = Seite-Zeile-Wort		
			3 = Seite-Zeile		
	36	1	0 = Fehler		
			1 = Kommentar (*)		
			2 = Löschen (-)		
			3 = Ersetzen (=)		
			4 = Einfügen (+)		
SW	37	2	Sortierwert (Versionsnummer)		
FNR	39	3	Folgenummer (für Fortsetzungszeilen)		
POS	42	3	Position des Korrekturzeichens (*, -, =, +) in der Korrekturanweisung		

Die erste Zahl gibt jeweils die Zeichenposition innerhalb des Korrekturschlüssels an, die zweite Zahl die Anzahl der Zeichen.

\*\*\*\*\*





**#VAUFBEREITE**

---

Kommando . . . . .	1107
Leistung . . . . .	1107
Beispiele . . . . .	1107
Allgemeines . . . . .	1109
MODUS = -STD- . . . . .	1109
MODUS = KUMULIERT . . . . .	1109
Parameter . . . . .	1110
Einstellen der Parameter-Konvention . . . . .	1110
Auswählen der Daten . . . . .	1110
Angaben zum Protokoll . . . . .	1111
Gestaltung des zeilensynoptischen Teils . . . . .	1113
Abbruch bei Fehler . . . . .	1115
Alphabetisches Verzeichnis der Parameter . . . . .	1115

## Kommando

#VAUFBEREITE

QUELLE	= <code>datei</code>	Name der Datei mit dem Grundtext.
MODUS	= <code>--STD-</code>	* Jede KORREKTUR-Datei enthält jeweils die Korrekturanweisungen zu einer Textversion.
	= <code>KUMULIERT</code>	Die KORREKTUR-Datei enthält die Korrekturanweisungen zu allen Textversionen.
LOESCHEN	= <code>-</code>	* Daten in der PROTOKOLL-Datei nicht löschen.
	= <code>+</code>	Daten in der PROTOKOLL-Datei zuerst löschen.
PARAMETER	= <code>datei</code>	Name der Datei mit den Parametern.
	= <code>*</code>	Die Parameter folgen auf das Kommando und sind mit <code>*EOF</code> abgeschlossen.
KORREKTUR	= <code>datei</code>	Name der Datei mit den Korrekturanweisungen (mit Korrekturschlüssel).  Bei <code>MODUS=-STD-</code> können auch mehrere Dateinamen angegeben werden; sie müssen durch Apostroph getrennt sein.
PROTOKOLL	= <code>--STD-</code>	* Protokoll in die Standard-Protokoll-Datei ausgeben.
	= <code>datei</code>	Name der Datei für das Protokoll.

## Leistung

Mit diesem Programm können Abweichungen einer oder mehrerer Textversionen von einem gemeinsamen Grundtext zusammen mit dem Grundtext zeilensynoptisch zum Drucken aufbereitet werden. Die Abweichungen müssen in Form von Korrekturanweisungen vorliegen, die mit einem Korrekturschlüssel versehen sind, wie er von den Kommandos #VERGLEICHE oder #SVORBEREITE erzeugt wird. Dabei wird zeilenweise unter den Wörtern des Grundtextes, für die eine Abweichung festgestellt wurde, der abweichende Wortlaut (Variante) ausgegeben. Übereinstimmungen zwischen dem Grundtext und den übrigen Textversionen, sowie Übereinstimmungen zwischen den Varianten selbst, werden markiert.

## Beispiele

Der Grundtext in Datei `basis` soll mit den beiden Textversionen in den Dateien `vers1` und `vers2` verglichen und die Abweichungen im Querformat auf einem HP-Laserjet, der den vom System vorgegebenen Namen `pr1` hat, ausgedruckt werden:

```
#VE,basis,vers1,W,+,*,korr1
SW          1
VKZ        /eins/
*EOF
```

```
#VE,basis,vers2,W,+,*,korr2
SW          2
VKZ        /zwei/
*EOF
```

```
#VA,basis,,+,*,korr1'korr2
DRT        HP-LP
*EOF
```

```
#DR,,HP-LP,pr1
```

Das im obigen Beispiel gezeigte Vorgehen ist mit bis zu neun Textversionen möglich. Sollen die Abweichungen von mehr als neun Textversionen zusammen ausgedruckt werden, so ist das folgende (im diesem Beispiel nur mit drei Textversionen gezeigte) Vorgehen notwendig:

```
#VE,basis,vers1,W,+,*,korr
SW          1
*EOF
```

```
#VE,basis,vers2,W,-,*,korr
SW          2
*EOF
```

```
#VE,basis,vers3,W,-,*,korr
SW          3
*EOF
```

```
#SO,korr,korr,1+44,+
```

```
#VA,basis,KUMULIERT,+,*,korr
DRT        HP-LP
SW          1    2    3
VKZ        /eins/zwei/drei/
*EOF
```

```
#DR,,HP-LP,pr1
```

## Allgemeines

Um einen Überblick über die Unterschiede von mehreren Versionen des gleichen Grundtextes zu erhalten, kann es sinnvoll sein, diese in einer gemeinsamen Liste zusammenzustellen. Dazu müssen die Abweichungen dieser Versionen vom Grundtext in Form von Korrekturanweisungen vorliegen.

Das Programm erwartet als Eingabe die Datei mit dem Grundtext und die Datei(en) mit den zu diesem Grundtext festgestellten Textabweichungen in Form von Korrekturanweisungen. Die Korrekturanweisungen müssen mit einem Korrekturschlüssel versehen sein. Dieser kann entweder vom Kommando #VERGLEICHE zusammen mit den Korrekturanweisungen erstellt werden oder er kann mit dem Kommando #SVORBEREITE vorhandenen Korrekturanweisungen hinzugefügt werden.

Das Protokoll enthält jeweils eine Zeile des Grundtextes und darunter zeilenweise den evtl. abweichenden Text der übrigen Textversionen. Dabei wird zeilenweise unter den Wörtern des Grundtextes, für die eine Abweichung festgestellt wurde, der abweichende Wortlaut (Variante) ausgegeben. Übereinstimmungen zwischen dem Grundtext und dem Text der übrigen Versionen werden markiert; Übereinstimmungen zwischen den Varianten selbst können gekennzeichnet werden.

Darüber hinaus können Übereinstimmungen zwischen dem Wortlaut verschiedener Versionen durch einen Verweis auf die erste Version, die diesen Wortlaut enthält, markiert werden.

MODUS = -STD-

Bei diesem Modus werden alle zu einer Textversion gehörenden Korrekturanweisungen jeweils in einer eigenen Datei erwartet. Es können bis zu neun Dateien angegeben werden. Die Reihenfolge der Zeilen mit dem vom Grundtext abweichenden Text wird durch die Reihenfolge bestimmt, in der die entsprechenden Dateien angegeben werden.

MODUS = KUMULIERT

Bei diesem Modus werden die Korrekturanweisungen zu den einzelnen (max. 50) Textversionen alle in einer einzigen Datei erwartet. Die Korrekturanweisungen müssen nach dem Korrekturschlüssel aufsteigend sortiert sein. Mit dem Parameter *sw* müssen die Sortierwerte derjenigen Textversionen angegeben werden, die berücksichtigt werden sollen. Die Reihenfolge der Zeilen mit dem vom Grundtext abweichenden Text wird durch die Reihenfolge bestimmt, in der die entsprechenden Sortierwerte angegeben werden. Mit dem Parameter *vkz* müssen außerdem Kennzeichen für alle mit dem Parameter *sw* angegebenen Textversionen angegeben werden.

## Parameter

Jeder Parameter ist einer Parameterart zugeordnet. Sie ist jeweils als römische Zahl in [ ] angegeben. Damit ist festgelegt, in welcher Form die Angaben vom Programm erwartet werden. Die einzelnen Parameterarten sind in der Beschreibung »TUSTEP-Grundlagen« im Kapitel »{}-Parameter« bzw. »<>-Parameter« definiert.

Für manche Parameter gibt es voreingestellte Werte, die angenommen werden, falls die entsprechenden Werte nicht angegeben sind. Diese Voreinstellungen sind in < > angegeben.

### Einstellen der Parameter-Konvention

Ob die Parameter nach der »{}-Parameter-Konvention« oder nach der »<>-Parameter-Konvention« interpretiert werden, kann mit dem Kommando #PARAMETER, {} bzw. #PARAMETER, <> eingestellt werden. Darüber hinaus kann die Konvention auch mit folgendem Parameter eingestellt werden.

- PAR**            Parameter-Konvention einstellen.
- { }    Nachfolgende Parameter nach der »{}-Parameter-Konvention« interpretieren.
  - <>    Nachfolgende Parameter nach der »<>-Parameter-Konvention« interpretieren.

Die Einstellung mit dem Parameter PAR hat Vorrang vor der mit dem Kommando #PARAMETER gewählten Einstellung; sie gilt jeweils nur für die nachfolgenden Parameter bis zum nächsten Parameter PAR bzw. bis zum Ende der Parameter dieses Programms.

### Auswählen der Daten

Sollen die gesamten Eingabedaten zu einem Protokoll aufbereitet werden, so braucht keiner der folgenden Parameter angegeben zu werden.

Falls nur ein bestimmter Bereich des Grundtextes berücksichtigt werden soll, kann dies mit den folgenden Parametern angegeben werden.

- BER**            Angabe eines Bereichs (»Seite.Zeile–Seite.Zeile«) oder einer Anfangsstelle (»Seite.Zeile«), falls nicht der gesamte Grundtext berücksichtigt werden soll. [ XI ]
- MAX**            Angabe für Testzwecke, wieviele Zeilen des Grundtextes maximal verarbeitet werden sollen. [ I ] <999999999>

## Angaben zum Protokoll

Von den folgenden Parametern muss der Parameter DRT in jedem Fall angegeben werden.

**DR** Angaben zur Druckausgabesteuerung. [ I ]

Es können vier Zahlenwerte angegeben werden:

1. Zahl: Spalten <1>

Anzahl der Spalten, die auf jeder Seite nebeneinander gedruckt werden sollen

2. Zahl: Rand <0>

Anzahl der Leerstellen links der ersten Spalte

3. Zahl: Breite <132>

Anzahl der Zeichen je Spalte

4. Zahl: Zwischenraum <0>

Anzahl der Leerstellen zwischen den Spalten

**DRZ** Zusätzliche Angaben zur Druckausgabesteuerung. [ I ]

Es können drei Zahlenwerte angegeben werden:

1. Zahl: Kopftext <3>

Anzahl der Zeilen für den Kopftext, einschließlich der Leerzeilen

2. Zahl: Höhe <60>

Anzahl der Zeilen je Reihe, ohne die Zeilen für den Kopf- und Fußtext

3. Zahl: Fußtext <0>

Anzahl der Zeilen für den Fußtext, einschließlich der Leerzeilen

**KT** Textteile, die als Kopftext oben auf jeder Seite gedruckt werden sollen.  
[ II ] <:&Q3 @/ &D2 &U2 &#6:>

In den Textteilen werden folgende Steueranweisungen durch die entsprechenden aktuellen Werte ersetzt:

&Q1 Projektname der QUELL-Datei (ohne Dateiname)

&Q2 Dateiname der QUELL-Datei (ohne Projektname)

&Q3 Projekt- und Dateiname der QUELL-Datei

&D0 Wochentag (z. B. Sonntag)

&D1 Datum xx.xx.xx (z. B. 02.04.96)

&D2 Datum xx. xxx. xxxx (z. B. 2. Apr. 2008)

&D3 Datum xx. xxxxxxxxxxxx xxxx (z. B. 2. April 2008)

&U1 Uhrzeit xx.xx (z. B. 12.00)

&U2 Uhrzeit xx:xx (z. B. 12:00)

&#n Seitennummer mit maximal n (n=1 bis 6) Stellen

Die Seitennummer kann nur einmal eingesetzt werden. Wird für die Seitennummer »- &#n -« (n=1 bis 6) angegeben, so wird die Seitennummer in die Mitte zwischen die Minuszeichen eingesetzt; die Minuszeichen werden bis auf ein Leerzeichen als Zwischenraum nach rechts bzw. links zur Seitennummer hin verschoben.

Jeder der Textteile kann durch die Formatieranweisungen »@z« und »@/« in drei Teile gegliedert sein:

linksbündig @z auf Mitte zentriert @/ rechtsbündig

Diese einzelnen Teile werden linksbündig, auf Mitte zentriert und rechtsbündig eingesetzt. Jeder einzelne Teil kann (bei Teil zwei und drei einschließlich der davor stehenden Formatieranweisung) fehlen.

Jeder Textteil wird in eine neue Zeile des Kopftextes gedruckt. Bei mehrspaltigem Druck können auch Textteile für die einzelnen Spalten angegeben werden. Dazu gibt es folgende Regelung:

Beginnt ein Textteil mit »\* :«, so wird der Rest des Textteils über jede Spalte in den Kopftext eingetragen. Steht anstelle des Sterns eine Zahl, so wird der Rest des Textteils über die durch die Zahl bezeichnete Spalte eingetragen. Hat die Zahl den Wert 0, so gilt der Rest des Textteils für die ganze Zeile. Beginnt ein Textteil nicht in der beschriebenen Weise, so wird »0 :« angenommen (Normalfall).

Mit einem Textteil, der für eine ganze Zeile des Kopftextes gilt, wird immer eine neue Zeile begonnen. Ein Textteil, der über einer bestimmten Spalte stehen soll, wird in die gleiche Zeile wie der vorangehende Textteil eingetragen, falls diese Zeile nicht schon einen Text für die ganze Zeile oder für diese oder eine weiter rechts stehende Spalte enthält; andernfalls wird mit diesem Textteil eine neue Zeile begonnen.

**FT** Textteile, die als Fußtext unten auf jeder Seite gedruckt werden sollen. [ II ] <>

Es gelten die gleichen Regelungen wie für den Kopftext (Parameter  $\kappa T$ ). Die Seitennummer kann jedoch nicht in den Fußtext eingesetzt werden, wenn sie schon in den Kopftext eingesetzt worden ist.

**PR** Angabe, ob alle Zeilen des Grundtextes oder nur die Zeilen, zu denen in den KORREKTUR-Dateien mindestens eine Korrekturanweisung vorhanden ist, protokolliert werden sollen. [ I ] <0>

0 = Nur die Zeilen des Grundtextes protokollieren, zu denen mindestens eine Korrekturanweisung vorhanden ist.

1 = Alle Zeilen des Grundtextes protokollieren.

**NSB** Neue Seite beginnen, falls für Grundtext und die einzelnen Versionen weniger als die angegebene Anzahl Zeilen auf der Seite frei sind. [ I ] <1 + Anzahl der Versionen>



**DRT** Druckertyp, für den die Daten aufbereitet werden. [ XI ]

Dieser Parameter ist obligat.

Hinweis: Mit dem Kommando #LISTE, DRUCKERTYPEN werden die definierten Druckertypen aufgelistet.

## Gestaltung des zeilensynoptischen Teils

Falls zu einer Zeile des Grundtextes eine Korrekturanweisung vorhanden ist, wird im Protokoll jeweils die Zeile des Grundtextes einschließlich Seiten-Zeilen-Nummer und darunter zeilenweise der Text der einzelnen Versionen ausgegeben.

Beispiel:

```
1.1      Beispiel für      3 Fassungen   eines kurzen  Textes
[eins]   =====   == die = Abschriften =====
[zwei]   =====   ==      = Abschriften ===== simplen =====
```

Bei MODUS=-STD- wird vor den Zeilen, die den Text der Versionen enthalten, das Versionskennzeichen ausgegeben. Falls in der Korrekturanweisung kein Versionskennzeichen enthalten ist, wird die Nummer der KORREKTUR-Datei, aus der die Korrekturanweisung stammt, ausgegeben. Die Nummern der KORREKTUR-Dateien werden durch Abzählen bestimmt. Die erste erhält die Nummer 1, die zweite die Nummer 2, usw.

Bei MODUS=KUMULIERT wird vor den Zeilen, die den Text der einzelnen Versionen enthalten, das mit den beiden folgenden Parametern bestimmte Versionskennzeichen ausgegeben. Die Korrekturanweisungen werden auf Grund des im Korrekturschlüssel enthaltenen Sortierwertes den einzelnen Textversionen zugeordnet. Für jede vorkommende Textversion muss mit dem Parameter SW der Sortierwert und mit dem Parameter VKZ das entsprechende Versionskennzeichen angegeben werden. Ein evtl. in den Korrekturanweisungen enthaltenes Versionskennzeichen bleibt unberücksichtigt.

**SW** Sortierwerte, die in den Korrekturschlüsseln vorkommen. [ I ]

Korrekturanweisungen mit einem nicht angegebenen Sortierwert werden übergangen; in diesem Fall wird am Ende eine entsprechende Warnung ausgegeben.

Dieser Parameter ist bei MODUS=KUMULIERT obligat.

**VKZ** Textteile (parallel zum Parameter SW), die als Versionskennzeichen verwendet werden sollen. [ II ]

Dieser Parameter ist bei MODUS=KUMULIERT obligat.

Falls zu einer Zeile des Grundtextes (mindestens) eine Korrekturanweisung zu irgend einer mit dem Parameter SW angegebenen Version vorhanden ist, wird im Protokoll die Zeile des Grundtextes und darunter zeilenweise der Text der einzelnen Versionen ausgegeben. Die Ausgabe der Zeilen, die die einzelnen Versionen betreffen, kann unter bestimmten Bedingungen eingeschränkt werden:

**PRE** Protokoll-Ausgabe für einzelne Versionen einschränken (Angaben parallel zu `SW`). [ I ] <0, 0, 0, . . . >

0 = Zeile mit der jeweiligen Version immer ausgegeben.

1 = Zeile mit der jeweiligen Version nur ausgegeben, wenn eine Korrekturanweisung für die Zeile vorhanden ist, oder wenn die Zeile zu einem mit dem Parameter `VB` (siehe Kommando #VERGLEICHE) angegebenen Bereich gehört.

Dieser Parameter ist nur bei `MODUS=KUMULIERT` erlaubt.

An den Stellen, an denen der Text einer Version wortweise mit dem Grundtext übereinstimmt, werden (statt des Textes der Version) Gleichheitszeichen »=« ausgegeben.

Soll anstelle des Gleichheitszeichens ein anderes Zeichen oder der übereinstimmende Wortlaut ausgegeben werden, so kann dies mit dem folgenden Parameter angegeben werden.

**GLT** Zeichen, das bei Übereinstimmung von Grundtext und Text der Version ausgegeben werden soll. [ XI ] <=>

Wird der Parameter `GLT` ohne Angabe eines Zeichens angegeben, so wird bei Übereinstimmung von Grundtext und Text der Version das übereinstimmende Wort ausgegeben.

Soll nur bei bestimmten Versionen auch bei Übereinstimmung von Versionstext und Grundtext (statt des Gleichheitszeichens bzw. des mit dem Parameter `GLT` angegebenen Zeichens) den übereinstimmenden Text ausgegeben werden, so kann dies mit dem folgenden Parameter angegeben werden.

**GTZ** Sortierwerte der Versionen, bei denen trotz Übereinstimmung zwischen dem Text der Version und dem Grundtext der Text ausgegeben werden soll. [ I ]

Auslassungen im Text der Versionen werden durch entsprechend viele Leerzeichen angezeigt. Soll statt dessen ein anderes Zeichen ausgegeben werden, so kann dies mit dem folgenden Parameter angegeben werden.

**LCK** Zeichen, das bei Auslassungen (Lücken) im Text der Version ausgegeben werden soll. [ XI ] < >

An den Stellen, an denen der Text der Version vom Grundtext abweicht, wird der abweichende Text (die Variante) ausgegeben. Sind zur gleichen Stelle des Grundtextes Varianten vorhanden, die untereinander übereinstimmen, so kann beim zweiten und weiteren Vorkommen der Varianten statt des Wortlauts der Variante ein Verweis auf das erste Vorkommen ausgegeben werden. Dies kann mit dem folgenden Parameter verlangt werden.

**GLV** Angabe, was bei gleichen Varianten ausgegeben werden soll. [ I ] <0>

0 = Wortlaut der Variante wird ausgegeben.

1 = Verweis auf das erste Vorkommen der Variante wird ausgegeben. Dabei wird anstelle des sich wiederholenden Wortes, in eckigen Klammern eingeschlossen, das Versionskennzeichen, bzw. der Sortierwert der Version eingesetzt, in der diese Variante zum ersten Mal vorkommt.

2 = Wie 1; falls jedoch die im Protokoll unmittelbar vorhergehende Textversion die gleiche Variante enthält, werden anstelle des Verweises Gänsefüßchen (") ausgegeben.

Enthält die KORREKTUR-Datei auch Angaben zu Vergleichsbereichen (z. B. auf Grund des Parameters VB beim Kommando #VERGLEICHE), so werden Textstellen, die außerhalb des aktuellen Vergleichsbereichs liegen, mit »\« gekennzeichnet.

## Abbruch bei Fehler

**ABB** Programm abbrechen, falls in den Korrekturanweisungen mehr als die angegebene Anzahl Fehler auftreten. [ I ] <0>

## Alphabetisches Verzeichnis der Parameter

<b>ABB</b>	Abbruch bei Fehler . . . . .	1115
<b>BER</b>	Auswählen eines Bereichs . . . . .	1110
<b>DR</b>	Druckausgabesteuerung . . . . .	1111
<b>DRT</b>	Druckertyp . . . . .	1113
<b>DRZ</b>	Druckausgabesteuerung – zusätzliche Angaben . . . . .	1111
<b>FT</b>	Fußtext . . . . .	1112
<b>KT</b>	Kopftext . . . . .	1111
<b>GLT</b>	Protokollzeichen bei Gleichheit des Textes . . . . .	1114
<b>GLV</b>	Ausgabe bei gleichen Varianten . . . . .	1114
<b>GTZ</b>	Zusatzangaben zu GLT . . . . .	1114
<b>LCK</b>	Protokollzeichen bei Lücken . . . . .	1114
<b>MAX</b>	Maximum für Testzwecke . . . . .	1110
<b>NSB</b>	Neue Seite beginnen . . . . .	1112
<b>PAR</b>	Parameter-Konvention . . . . .	1110
<b>PR</b>	Protokollierung des Grundtextes . . . . .	1112
<b>SW</b>	Sortierwerte . . . . .	1113
<b>PRE</b>	Protokoll-Ausgabe einschränken . . . . .	1114
<b>VKZ</b>	Versionskennzeichen . . . . .	1113

\* \* \* \* \*



**#VERGLEICHE**

---

Kommando . . . . .	1119
Leistung . . . . .	1119
Beispiele . . . . .	1120
Allgemeines . . . . .	1121
Arbeitsweise . . . . .	1121
Vergleichsprotokoll . . . . .	1122
Korrekturanweisungen . . . . .	1123
Parameter . . . . .	1125
Einstellen der Parameter-Konvention . . . . .	1125
Silbentrennung . . . . .	1125
Auswählen der Daten . . . . .	1125
Angaben zum Protokoll . . . . .	1129
Angaben zu den Korrekturanweisungen . . . . .	1132
Angaben zum Vergleich . . . . .	1127
Angaben zum Zuordnen . . . . .	1128
Alphabetisches Verzeichnis der Parameter . . . . .	1135
Kompatibilitätsmodi mit speziellen Parametern . . . . .	1136
Angaben zur Silbentrennung . . . . .	1137
Auswählen der Daten . . . . .	1137
Angaben zum Vergleich . . . . .	1139
Aufbau eines Datensatzes in der KORREKTUR-Datei . . . . .	1141
Aufbau des Korrekturschlüssels . . . . .	1142

## Kommando

### #VERGLEICHE

VERSIONA	=	datei	Eingabedatei mit Textversion A.
VERSIONB	=	datei	Eingabedatei mit Textversion B.
MODUS	=	WORT	Wortweise vergleichen.
	=	ZEILE	Zeilenweise vergleichen.
	=	<>	Wortweise vergleichen, wobei jedes Tag auch als Wort gilt, wenn es nicht zwischen Leerstellen steht.
	=	-STD-	(noch nicht definiert)
	=	...	Weitere (veraltete) Modi siehe »Kompatibilitätsmodi mit speziellen Parametern«
LOESCHEN	=	-	* Daten in der KORREKTUR- und in der PROTOKOLL-Datei nicht löschen.
	=	+	Daten in der KORREKTUR- und in der PROTOKOLL-Datei zuerst löschen.
PARAMETER	=	-	* Keine Parameter.
	=	datei	Name der Datei mit den Parametern.
	=	*	Die Parameter folgen auf das Kommando und sind mit *EOF abgeschlossen.
KORREKTUR	=	-	* Keine Korrekturanweisungen ausgeben.
	=	datei	Name der Datei für die Korrekturanweisungen.
PROTOKOLL	=	-	* Kein Protokoll erstellen.
	=	+	Protokoll ins Ablaufprotokoll ausgeben.
	=	-STD-	Protokoll in die Standard-Protokoll-Datei ausgeben.
	=	datei	Name der Datei für das Protokoll.

## Leistung

Mit diesem Programm können zwei Textversionen (A und B) miteinander verglichen werden. Die festgestellten Unterschiede werden in der zur Spezifikation PROTOKOLL angegebenen Datei protokolliert. Außerdem können die Unterschiede in Form von Korrekturanweisungen in die zur Spezifikation KORREKTUR angegebene Datei ausgegeben werden. Sie entsprechen den Konventionen für die Korrekturanweisungen des Kommandos #KAUSFUEHRE; würde mit den Korrekturanweisungen die Version A korrigiert, entstünde (u. U. bis auf die Zeileneinteilung und die Satznummern) die Version B.

Die Zeileneinteilung der beiden Versionen kann völlig verschieden sein. Auslassungen bzw. Einfügungen werden (zeitaufwändig) bis etwa zur Länge einer DIN-A4-Seite erkannt. Auslassungen bzw. Einfügungen beliebiger Länge können durch entsprechende Angaben berücksichtigt werden. Es ist auch möglich, nur bestimmte Dateibereiche zu vergleichen.

### Einschränkung

Bei `MODUS=<>` können keine Korrekturanweisungen ausgegeben werden. Die Angabe eines Dateinamens zur Spezifikation `KORREKTUR` ist in diesem Fall nicht erlaubt.

### Beispiele

Vergleich zweier Dateien und Anzeige der Unterschiede auf dem Bildschirm. Dabei darf das Zweitprotokoll nicht eingeschaltet sein (vgl. Kommando `#PROTOKOLL`), sonst werden die Unterschiede ins Zweitprotokoll ausgegeben.

```
#VE,dat1,dat2,W,PR=+
```

Vergleich zweier Dateien und Ausdrucken der Unterschiede im Querformat auf einem HP-LaserJet, der den vom System vorgegebenen Druckernamen `pr1` hat:

```
#VE,dat1,dat2,W,+,PR=-STD-
```

```
#DR,,HP-LP,pr1
```



## Allgemeines

Zweck des Vergleichs von zwei oder mehr Textversionen kann u. a. sein:

- die Kontrolle über Änderungen im Text (z. B. die Protokollierung von Korrekturen, die mit dem Kommando #EDIERE ausgeführt wurden); das Ergebnis des Vergleichs ist in diesem Fall eine gedruckte Liste, in der die Unterschiede nachgewiesen sind;
- das Auffinden von Erfassungsfehlern bei Texten, die zum Zweck der teilautomatischen Korrektur doppelt erfasst wurden; um diese teilautomatische Korrektur zu ermöglichen, werden die Unterschiede beider Fassungen in Form von Korrekturanweisungen in eine Datei ausgegeben;
- das Auffinden und der Nachweis von Überlieferungsvarianten bei Texten, die kritisch ediert werden sollen; in diesem Fall werden die Vergleichsergebnisse in einer erweiterten Form ausgegeben, so dass die Ergebnisse aufeinander folgender einzelner Vergleiche des Grundtextes mit den übrigen Textversionen ineinander gemischt und weiterverarbeitet werden können.

## Arbeitsweise

Mit einem Aufruf können jeweils zwei Textversionen miteinander verglichen werden. Sollen die Unterschiede von mehr als zwei Textversionen festgestellt werden, so müssen die übrigen Textversionen jeweils einzeln mit dem selben Grundtext (Version A) verglichen werden; die beim Vergleich von je zwei Versionen gefundenen Unterschiede können mit anderen TUSTEP-Programmen weiterverarbeitet werden.

Die Zeileneinteilung der verglichenen Textversionen darf verschieden sein; ebenso dürfen beide jeweils zu vergleichenden Textversionen Einfügungen oder Auslassungen enthalten; diese werden vom Programm automatisch (mit entsprechendem Zeitaufwand bis zur Länge von jeweils ca. einer DIN-A4-Seite) erkannt oder können vom Benutzer als solche angegeben und dann vom Programm (ohne zusätzlichen Zeitaufwand) berücksichtigt werden.

Die Unterschiede der jeweils verglichenen Textversionen können in Form von Korrekturanweisungen in eine Text-Datei ausgegeben werden; die Form der Korrekturanweisungen kann über Parameter den verschiedenen Verwendungszwecken angepasst werden.

## Vergleichsprotokoll

Für jede Zeile aus Version A, für die ein Unterschied festgestellt wurde, wird der Text der Version A und darunter der entsprechende Text der Version B ausgegeben. Zwischen diesen beiden Zeilen werden die festgestellten Unterschiede wie folgt markiert:

- eine Auslassung wird durch »-« unter den Zeichen der oberen Zeile markiert, die in der unteren Zeile fehlen;
- eine Einfügung wird durch »+« über den Zeichen der unteren Zeile markiert, die in dieser eingefügt sind;
- eine Ersetzung wird wie eine Auslassung in der oberen und eine Einfügung in der unteren Zeile markiert. Die Stellen, an denen »+« und »-« zusammentreffen würden, werden mit einem »\*« markiert.
- eine Ersetzung, die nur Groß- und Kleinschreibung betrifft, wird in der oberen und in der unteren Zeile mit »=« markiert.

Beispiel:

```

1.1      Beispiel für      2 Fassungen      eines kurzen Textes
-->                +++      *****      -----
1.1      Beispiel für die 2 Abschriften eines      Textes

```

Falls eine Auslassung genau eine oder mehrere ganze Zeilen betrifft, werden nur diese Zeilen ausgegeben. Zwischen Satznummer und Text werden solche Zeilen mit »-« gekennzeichnet.

Beispiel:

```

2.3      -Erste Zeile, die in Version B fehlt
2.4      -Zweite Zeile, die ebenfalls fehlt

```

Falls eine Einfügung genau eine oder mehrere ganze Zeilen betrifft, wird die Zeile ausgegeben, nach der die Zeilen eingefügt wurden. Sie wird mit »+« gekennzeichnet. Danach werden die eingefügten Zeilen ausgegeben. Sie werden mit »+« gekennzeichnet.

Beispiel:

```

2.3      =Zeile aus Version A, nach der Zeilen eingefügt wurden
2.3/1    +Erste Zeile, die in Version B eingefügt wurde
2.3/2    +Zweite Zeile, die ebenfalls eingefügt wurde

```

Falls eine Ersetzung genau eine oder mehrere ganze Zeilen betrifft, werden nur diese Zeilen ausgegeben. Sie werden mit »-« bzw. mit »+« gekennzeichnet.

Beispiel:

```

2.3   -Erste Zeile aus Version A, die ersetzt wurde
2.4   -Zweite Zeile, die ebenfalls ersetzt wurde
2.3   +Erste Zeile aus Version B, die eingefügt wurde
2.3/1 +Zweite Zeile, die ebenfalls eingefügt wurde
2.3/2 +Weitere Zeile, die auch noch eingefügt wurde

```

Bei `MODUS=ZEILE` gilt folgendes, wenn zu Parameter `PR` nichts anderes angegeben ist: Unterscheiden sich mehrere unmittelbar aufeinander folgende Zeilen aus Version A und Version B nur dadurch, dass sie um die gleiche Anzahl Leerstellen weiter bzw. weniger weit eingerückt sind, wird von diesen Zeilen nur die erste und letzte aus Version A ausgegeben. Dazwischen wird angegeben, wieviele Zeilen um wieviele Leerstellen in Version B weiter bzw. weniger weit eingerückt sind.

Beispiele:

```

2.3     Erste von 12 weiter eingerückten Zeilen
      >>> 12 * 3
2.14    Letzte von 12 weiter eingerückten Zeilen

2.3     Erste von 6 weniger weit eingerückten Zeilen
      <<< 6 * 3
2.8     Letzte von 6 weniger weit eingerückten Zeilen

```

## Korrekturanweisungen

Falls zur Spezifikation `KORREKTUR` eine Datei angegeben ist, werden in diese Datei Korrekturanweisungen ausgegeben. Wenn über Parameter nichts anderes verlangt wird, enthalten die mit dem Kommando `#VERGLEICHE` erzeugten Korrekturanweisungen die Bestandteile, die vom Kommando `#KAUSFUEHRE` erwartet werden: Angabe des zu korrigierenden Textbereichs in Version A, Angabe der Korrekturart (`»-«`, `»+«`, `»=«`, `»*«` für `»Löschen«`, `»Einfügen«`, `»Ersetzen«`, Kommentar), Angabe des von der Korrekturanweisung betroffenen Textes in der Version B (`»Korrekturtext«`).

Wird die Textversion A mit dem Kommando `#KAUSFUEHRE` unter Verwendung dieser Korrekturanweisungen korrigiert, so erhält man (ggf. bis auf die Zeileneinteilung und die Satznummern) den Text der Version B. Benutzt man das Kommando `#VERGLEICHE` zum Auffinden von Erfassungsfehlern bei Texten, die zum Zweck der teilautomatischen Korrektur doppelt erfasst wurden, so wird man diese Korrekturanweisungen zuvor (in der Regel vor allem durch Löschen der Korrekturanweisungen für die Stellen, an denen die Textversion A schon richtig war) entsprechend verändern, damit die erforderlichen Korrekturen durch einen nachfolgenden Aufruf des Kommandos `#KAUSFUEHRE` ausgeführt werden.

Für weitergehende Aufgabenstellungen, z. B. aus dem Bereich der Textkritik, ist über die oben beschriebenen Bestandteile hinaus die Erweiterung der Korrekturanweisungen um den Korrekturschlüssel, um den von der Korrekturanweisung betroffenen Text (`»Originalwortlaut«`) und seine Umgebung (`»Kontext«`) in Version A, um die Stellenangabe in Version B, sowie um ein Kennzeichen für die Version B vorgesehen.

Der Korrekturschlüssel wird benötigt, wenn die Unterschiede von mehr als zwei Textversionen in einer gemeinsamen Übersicht zusammengestellt werden sollen. Er kann über Parameter verlangt werden.

Will man mit dem Kommando #VAUFBEREITE die Unterschiede von mehr als zwei Textfassungen in zeilensynoptischer Anordnung zum Ausdrucken aufbereiten, so muss eine (beliebige) Textversion A nacheinander mit den übrigen, jeweils als Version B angegebenen Textversionen verglichen werden; die Unterschiede müssen als Korrekturanweisungen mit Korrekturschlüssel in KORREKTUR-Dateien abgespeichert werden.

Originalwortlaut und Kontext der Varianten in Textversion A werden zusätzlich u. a. dann benötigt, wenn die Varianten in der von einem kritischen Apparat gewohnten Form zusammengestellt werden sollen.

Die beim Vergleich der verschiedenen Textversionen mit dem gleichen Grundtext gewonnenen, mit Korrekturschlüssel und den übrigen Ergänzungen versehenen Korrekturanweisungen können zu diesem Zweck mit den Kommandos #SVORBEREITE und einem anschließenden #SORTIERE in die dazu notwendige Reihenfolge (nämlich: Stellenangabe, Wortlaut der Abweichung, Sortierwert für die diese Abweichung enthaltende Quelle) gebracht und mit dem Kommando #RAUFBEREITE zusammen mit dem Wortlaut des Grundtextes (»Lemma«) ausgegeben werden.

## Parameter

Jeder Parameter ist einer Parameterart zugeordnet. Sie ist jeweils als römische Zahl in [ ] angegeben. Damit ist festgelegt, in welcher Form die Angaben vom Programm erwartet werden. Die einzelnen Parameterarten sind in der Beschreibung »TUSTEP-Grundlagen« im Kapitel »{}-Parameter« bzw. »<>-Parameter« definiert.

Für manche Parameter gibt es voreingestellte Werte, die angenommen werden, falls die entsprechenden Werte nicht angegeben sind. Diese Voreinstellungen sind in < > angegeben.

Außer den im folgenden beschriebenen Parametern sind auch Parameter zur Definition von Zeichen- und Stringgruppen möglich. [ v ]

### Einstellen der Parameter-Konvention

Ob die Parameter nach der »{}-Parameter-Konvention« oder nach der »<>-Parameter-Konvention« interpretiert werden, kann mit dem Kommando #PARAMETER, {} bzw. #PARAMETER, <> eingestellt werden. Darüber hinaus kann die Konvention auch mit folgendem Parameter eingestellt werden.

**PAR**            Parameter-Konvention einstellen.

    {}    Nachfolgende Parameter nach der »{}-Parameter-Konvention« interpretieren.

    <>    Nachfolgende Parameter nach der »<>-Parameter-Konvention« interpretieren.

Die Einstellung mit dem Parameter **PAR** hat Vorrang vor der mit dem Kommando #PARAMETER gewählten Einstellung; sie gilt jeweils nur für die nachfolgenden Parameter bis zum nächsten Parameter **PAR** bzw. bis zum Ende der Parameter dieses Programms.

### Silbentrennung

Silbentrennungen bleiben unberücksichtigt. Jeder Bestandteil eines getrennten Wortes wird als ein eigenständiges Wort behandelt.

Angaben zur Silbentrennung sind jedoch bei den Kompatibilitätsmodi möglich (siehe Parameter **STR** Seite 1137).

### Auswählen der Daten

Soll der gesamte Inhalt beider Dateien in der bestehenden Reihenfolge verglichen werden, braucht keiner der folgenden Parameter angegeben zu werden.

Soll in beiden Versionen nur ein bestimmter Bereich bzw. sollen beide Versionen nur ab einer bestimmten Stelle verglichen werden, und sind die Bereichs- bzw. Stellenangaben in beiden Versionen identisch, kann der Bereich bzw. die Stelle mit dem

Parameter `BER` angegeben werden. Sind die Bereichs- bzw. Stellenangaben für den zu vergleichenden Text in den beiden Versionen nicht identisch, muss die Auswahl mit dem Parameter `VB` angegeben werden.

**BER** Angabe eines Bereichs (»Seite.Zeile–Seite.Zeile«, falls die Datei im Textmodus nummeriert ist; Zeile–Zeile, falls die Datei im Programmmodus nummeriert ist) oder einer Anfangsstelle (»Seite.Zeile« bzw. »Zeile«), falls nicht jeweils die gesamte Datei (von Version A und B) verglichen werden soll. [ XI ]

Sind beide Eingabedateien Segment-Dateien und soll aus beiden ein Segment mit dem gleichen Namen verglichen werden, kann anstelle des Bereichs der Name des Segments angegeben werden.

Ist eine der Eingabedateien eine Segment-Datei und die andere Eingabedatei im Programmmodus nummeriert, kann anstelle des Bereichs ebenfalls der Name des Segments angegeben werden. In diesem Fall werden die Daten des angegebenen Segments aus der Segment-Datei mit den Daten der anderen Datei verglichen.

Dieser Parameter ist nur zugelassen, wenn die Satznummern in beiden Dateien aufsteigend sind; außerdem schließen sich die Parameter `VB` und `BER` gegenseitig aus.

Sollen aus beiden Dateien nur bestimmte Bereiche verglichen werden, kann das mit dem Parameter `VB` angegeben werden. Bei diesem Parameter können die Bereichsangaben in den beiden Versionen unterschiedlich sein.

Die Angabe von Bereichen kann auch sinnvoll sein, wenn zwar der gesamte Inhalt beider Dateien verglichen werden soll, aber wegen großer Unterschiede, insbesondere wegen größerer Auslassungen oder Einfügungen, ein großer Zeitaufwand zum Auffinden paralleler Stellen oder eine nur ungenaue Zuordnung der gefundenen Unterschiede zu erwarten ist.

**VB** Angabe von zu vergleichenden Bereichen. Mehrere Angaben sind möglich und müssen durch Apostroph getrennt werden. [ XI ]

Eine vollständige Angabe für einen in beiden Versionen zu vergleichenden Bereich besteht aus der Bereichsangabe für Version A und der Bereichsangabe des entsprechenden Bereichs in Version B, die durch »=« verbunden sind.

Eine Bereichsangabe besteht aus einer Anfangsposition und, mit einem Minuszeichen damit verbunden, einer Endposition in der Form  $s.z[/u][,w]-s.z[/u][,w]$ , wobei  $s$  für die (bis zu 6-stellige) Seitennummer,  $z$  für die (bis zu 3-stellige) Zeilennummer,  $u$  für die (bis zu 3-stellige) Unterscheidungsnummer und  $w$  für die (bis zu 2-stellige) Wortnummer steht; die in [ ] eingeschlossenen Teile können fehlen. Ist keine Wortnummer angegeben, wird die Wortnummer 1 angenommen.

Die Anfangsposition ist jeweils das erste Wort, das zum Bereich gehört; die Endposition ist jeweils das erste Wort, das nicht mehr zum Bereich gehört.

Sind die Bereichsangaben für die Version A und für die Version B identisch, so kann die Angabe für die Version B einschließlich des davor stehenden »=« entfallen.

Schließt bei zwei aufeinander folgenden Bereichsangaben zur gleichen Version der zweite Bereich unmittelbar an den ersten an, so kann die Angabe der Endposition in der ersten Bereichsangabe einschließlich des davor stehenden »-« entfallen.

Beginnt der zu vergleichende Bereich am Dateianfang, kann als Anfangsposition 0 . 0 angegeben werden; endet der zu vergleichende Bereich am Dateiende, kann als Endposition 0 . 0 angegeben werden.

Die Bereichsangaben werden in Form einer Korrekturanweisung (Korrekturart = Kommentar, Korrekturzeichen = »\*«) vor den diesen Bereich betreffenden Korrekturanweisungen in die KORREKTUR-Datei ausgegeben.

Dieser Parameter ist nur zugelassen, wenn die Satznummern in beiden Dateien aufsteigend sind; außerdem schließen sich die Parameter BER und VB gegenseitig aus.

## Angaben zum Vergleich

Sollen beim Vergleich zweier Textversionen bestimmte Unterschiede (z. B. verschiedene Schreibweisen eines Wortes) unberücksichtigt bleiben, kann dies durch Angaben zu den folgenden Parametern ermöglicht werden.

Achtung:

Bei MODUS=WORT können Zeichenfolgen jeweils nur innerhalb einzelner Wörter ausgetauscht werden; auszutauschende Textteile können nur einzelne Wörter sein. Als Trennzeichen zwischen den Wörtern gelten nur ein oder mehrere Leerzeichen. Sonderzeichen gelten nicht als Trennzeichen und gehören jeweils zum Wort.

Bei MODUS=ZEILE können Zeichenfolgen jeweils nur innerhalb einzelner Zeilen ausgetauscht werden; auszutauschende Textteile können nur einzelne Zeilen sein.

Falls mehr als einer der Parameter XV, XVX und WTV angegeben ist, werden diese Parameter in der hier beschriebenen Reihenfolge auf den ggf. schon modifizierten Text angewandt.

**XV** Zeichenfolgenpaare (und Ausnahmezeichenfolgen), deren jeweils erste Zeichenfolge durch die jeweils zweite Zeichenfolge ersetzt werden soll. [ X ]

**XVX** Zeichenfolgenpaare (und Ausnahmezeichenfolgen), deren jeweils erste Zeichenfolge durch die jeweils zweite Zeichenfolge ersetzt werden soll. [ X ]

**WTV** Paare von Textteilen. [ IV ]

Der jeweils erste Textteil wird auf Übereinstimmung mit einzelnen Wörtern bzw. Zeilen geprüft. Dabei werden Groß- und Kleinbuchsta-

ben nicht unterschieden. Bei Übereinstimmung wird das Wort bzw. die Zeile durch den jeweils zweiten Textteil ersetzt.

**GKU** Angabe, ob Groß- und Kleinbuchstaben unterschieden werden sollen.  
[ 1 ] <1>

- 0 = Groß- und Kleinbuchstaben nicht unterscheiden
- 1 = Groß- und Kleinbuchstaben beim Vergleich auf Übereinstimmung unterscheiden
- 2 = Groß- und Kleinbuchstaben beim Vergleich und beim Zuordnen unterscheiden

## Angaben zum Zuordnen

Treten beim Vergleich zweier Textversionen Unterschiede auf, so muss das Programm einerseits die Abgrenzung größerer Einfügungen bzw. Auslassungen vornehmen, andererseits die einander entsprechenden Textteile aus beiden Versionen, trotz der darin enthaltenen Unterschiede, einander möglichst wortgenau zuordnen.

Sind die Unterschiede durch orthographische Eigenheiten der zu vergleichenden Versionen bedingt, so kann dem Programm eine bessere Zuordnung des Textes der beiden zu vergleichenden Versionen durch Angaben zu den folgenden Parametern ermöglicht werden.

Achtung:

Bei `MODUS=WORT` können Zeichenfolgen jeweils nur innerhalb einzelner Wörter ausgetauscht werden; auszutauschende Textteile können nur einzelne Wörter sein. Als Trennzeichen zwischen den Wörtern gelten nur ein oder mehrere Leerzeichen. Sonderzeichen gelten nicht als Trennzeichen und gehören jeweils zum Wort.

Bei `MODUS=ZEILE` können Zeichenfolgen jeweils nur innerhalb einzelner Zeilen ausgetauscht werden; auszutauschende Textteile können nur einzelne Zeilen sein.

Falls mehr als einer der Parameter `XZ`, `XVZ` und/oder `WTZ` angegeben ist, werden diese Parameter in der hier beschriebenen Reihenfolge auf den ggf. schon modifizierten Text angewandt.

Falls auch die Parameter `XV`, `XVX` und/oder `WTV` angegeben sind, werden die Parameter `XZ`, `XZX` und/oder `WTZ` auf den ggf. schon damit modifizierten Text angewandt.

**XZ** Zeichenfolgenpaare (und Ausnahmezeichenfolgen), deren jeweils erste Zeichenfolge durch die jeweils zweite Zeichenfolge ersetzt werden soll. [ X ]

Voreinstellung:

```
>SZ          , . ! ? ; : " [ ] ( ) { } - \
XZ           : % { 1-2 } { % } : : # . : : # { ! } [ + - ] : : { C : SZ } : : { 2-0 } : :
```

**XZX** Zeichenfolgenpaare (und Ausnahmezeichenfolgen), deren jeweils erste Zeichenfolge durch die jeweils zweite Zeichenfolge ersetzt werden soll. [ X ]



**WTZ** Paare von Textteilen. [ IV ]

Der jeweils erste Textteil wird auf Übereinstimmung mit einzelnen Wörtern bzw. Zeilen geprüft. Dabei werden Groß- und Kleinbuchstaben nicht unterschieden. Bei Übereinstimmung wird das Wort bzw. die Zeile durch den jeweils zweiten Textteil ersetzt.

Das Programm versucht die Textteile, die in beiden Versionen Unterschiede aufweisen, einander möglichst wortgenau zuzuordnen, auch wenn sie sich in Einzelheiten (z. B. in der Orthographie) unterscheiden. Das kann dazu führen, dass sich unterscheidende Wörter einander zugeordnet werden, obwohl eine wortweise Zuordnung nicht sinnvoll ist. Mit dem folgenden Parameter kann als zusätzliche Bedingungen für die Zuordnung angegeben werden, dass die umgebenden Wörter mit berücksichtigt werden sollen.

**NGZ** Nur gleiche Wörter zuordnen. [ I ] <0>

Angabe, wieviele aufeinander folgende Wörter vor und nach den sich unterscheidenden Wörtern mindestens übereinstimmen müssen, damit die dazwischen liegenden, sich unterscheidenden Wörter nicht wortweise, sondern als Wortfolge einander zugeordnet werden. In der Regel reicht der Wert 1 für eine sinnvolle Zuordnung aus.

## Angaben zum Protokoll

Soll kein Protokoll ausgegeben werden (`PROTOKOLL=-`), braucht keiner der folgenden Parameter angegeben zu werden.

Soll ein Protokoll in eine `PROTOKOLL`-Datei ausgegeben werden, muss mindestens der Parameter `DRT` angegeben werden, falls zur Spezifikation `MODUS` kein Druckertyp angegeben wurde.

**DR** Angaben zur Druckausgabesteuerung. [ I ]

Es können vier Zahlenwerte angegeben werden:

1. Zahl: Spalten <1>

Anzahl der Spalten, die auf jeder Seite nebeneinander gedruckt werden sollen

2. Zahl: Rand <0>

Anzahl der Leerstellen links der ersten Spalte

3. Zahl: Breite <132>

Anzahl der Zeichen je Spalte

4. Zahl: Zwischenraum <0>

Anzahl der Leerstellen zwischen den Spalten

**DRZ** Zusätzliche Angaben zur Druckausgabesteuerung. [ I ]

Es können drei Zahlenwerte angegeben werden:

1. Zahl: Kopftext <3>

Anzahl der Zeilen für den Kopftext, einschließlich der Leerzeilen

2. Zahl: Höhe <60>

Anzahl der Zeilen je Reihe, ohne die Zeilen für den Kopf- und Fußtext

3. Zahl: Fußtext <0>

Anzahl der Zeilen für den Fußtext, einschließlich der Leerzeilen

**KT**

Textteile, die als Kopftext oben auf jeder Seite gedruckt werden sollen.

[ II ] <: &Q3 @/ &D2 &U2 &#6::&Q0:>

In den Textteilen werden folgende Steueranweisungen durch die entsprechenden aktuellen Werte ersetzt:

&Q0 Segmentname (von Parameter BER)

&Q1 Projektname der QUELL-Datei (ohne Dateiname)

&Q2 Dateiname der QUELL-Datei (ohne Projektname)

&Q3 Projekt- und Dateiname der QUELL-Datei

&D0 Wochentag (z. B. Sonntag)

&D1 Datum xx.xx.xx (z. B. 02.04.96)

&D2 Datum xx. xxx. xxxx (z. B. 2. Apr. 2008)

&D3 Datum xx. xxxxxxxxxxxx xxxx (z. B. 2. April 2008)

&U1 Uhrzeit xx.xx (z. B. 12.00)

&U2 Uhrzeit xx:xx (z. B. 12:00)

&#n Seitennummer mit maximal n (n=1 bis 6) Stellen

Die Seitennummer kann nur einmal eingesetzt werden. Wird für die Seitennummer »- &#n -« (n=1 bis 6) angegeben, so wird die Seitennummer in die Mitte zwischen die Minuszeichen eingesetzt; die Minuszeichen werden bis auf ein Leerzeichen als Zwischenraum nach rechts bzw. links zur Seitennummer hin verschoben.

Jeder der Textteile kann durch die Formatieranweisungen »@z« und »@/« in drei Teile gegliedert sein:

linksbündig @z auf Mitte zentriert @/ rechtsbündig

Diese einzelnen Teile werden linksbündig, auf Mitte zentriert und rechtsbündig eingesetzt. Jeder einzelne Teil kann (bei Teil zwei und drei einschließlich der davor stehenden Formatieranweisung) fehlen.

Jeder Textteil wird in eine neue Zeile des Kopftextes gedruckt. Bei mehrspaltigem Druck können auch Textteile für die einzelnen Spalten angegeben werden. Dazu gibt es folgende Regelung:

Beginnt ein Textteil mit »\* :«, so wird der Rest des Textteils über jede Spalte in den Kopftext eingetragen. Steht anstelle des Sterns eine Zahl, so wird der Rest des Textteils über die durch die Zahl bezeichnete Spalte

te eingetragen. Hat die Zahl den Wert 0, so gilt der Rest des Textteils für die ganze Zeile. Beginnt ein Textteil nicht in der beschriebenen Weise, so wird »0 : « angenommen (Normalfall).

Mit einem Textteil, der für eine ganze Zeile des Kopftextes gilt, wird immer eine neue Zeile begonnen. Ein Textteil, der über einer bestimmten Spalte stehen soll, wird in die gleiche Zeile wie der vorangehende Textteil eingetragen, falls diese Zeile nicht schon einen Text für die ganze Zeile oder für diese oder eine weiter rechts stehende Spalte enthält; andernfalls wird mit diesem Textteil eine neue Zeile begonnen.

**FT** Textteile, die als Fußtext unten auf jeder Seite gedruckt werden sollen. [ II ] <>

Es gelten die gleichen Regelungen wie für den Kopftext (Parameter  $\kappa T$ ). Die Seitennummer kann jedoch nicht in den Fußtext eingesetzt werden, wenn sie schon in den Kopftext eingesetzt worden ist.

**PR** Angabe, welche Zeilen bei der Ausgabe in die PROTOKOLL-Datei protokolliert werden sollen und wie die Satznummern angezeigt werden sollen. [ I ]

Es können insgesamt drei Zahlenwerte angegeben werden.

1. Zahl: Wenn Unterschiede im Text auftreten <0>

0 = Nur die Zeilen aus Version A protokollieren, die Unterschiede zu Version B aufweisen.

1 = Alle Zeilen aus Version A protokollieren.

2. Zahl: Wenn sich mehrere unmittelbar aufeinander folgende Zeilen aus Version A und Version B nur dadurch unterscheiden, dass sie um die gleiche Anzahl Leerstellen weiter bzw. weniger weit eingerückt sind. <0>

0 = Nur erste und letzte betroffene Zeile protokollieren.

1 = Alle betroffenen Zeilen protokollieren.

3. Zahl: Wenn die Eingabedaten im Programmmodus nummeriert sind (genauer: wenn die Seitennummer aller Sätze 0 ist) <0>

0 = Satznummern im Programmmodus anzeigen.

1 = Satznummern im Textmodus anzeigen.

**DRT** Druckertyp, für den die Daten aufbereitet werden. [ XI ]

Dieser Parameter darf nicht angegeben werden, wenn zur Spezifikation MODUS schon ein Druckertyp angegeben wurde. In allen anderen Fällen ist dieser Parameter obligat, wenn das Protokoll in eine PROTOKOLL-Datei (d. h. nicht ins Ablaufprotokoll) ausgegeben werden soll.

Hinweis: Mit dem Kommando #LISTE, DRUCKERTYPEN werden die definierten Druckertypen aufgelistet.

## Angaben zu den Korrekturanweisungen

Werden keine Korrekturanweisungen ausgegeben (KORREKTUREN=-), braucht keiner der folgenden Parameter angegeben zu werden.

Werden Korrekturanweisungen ausgegeben und ist keiner der folgenden Parameter angegeben, werden in einer Korrekturanweisung nur die Stellenangabe des von der Korrekturanweisung betroffenen Textes in Version A, das Korrekturzeichen (-, =, + oder \*) und ggf. der Korrekturtext angegeben.

Eine Stellenangabe besteht aus einer Positionsangabe in der Form  $s.z[/u] [ , w]$  oder aus einer Anfangsposition und, mit einem Minuszeichen damit verbunden, einer Endposition in der Form  $s.z[/u] [ , w] - s.z[/u] [ , w]$ , wobei  $s$  für die (bis zu 6-stellige) Seitennummer,  $z$  für die (bis zu 3-stellige) Zeilennummer,  $u$  für die (bis zu 3-stellige) Unterscheidungsnummer und  $w$  für die (bis zu 2-stellige) Wortnummer steht; die in  $[ ]$  eingeschlossenen Teile können fehlen.

Die Stellenangaben werden so kurz wie möglich ausgegeben. So wird z. B.  $1.2, 3-1.2, 5$  zu  $1.2, 3-5$  verkürzt und  $1.2, 1-1.4, 5$  zu  $1.2-4$ , falls das 5. Wort das letzte in der Zeile 1.4 ist und die Stellenangabe deshalb nur vollständige Zeilen bezeichnet.

Für die weitere Bearbeitung der Korrekturanweisungen kann es zweckmäßig sein, die automatische Verkürzung der Stellenangaben mit dem folgenden Parameter zu unterdrücken.

**SZW** Angabe, ob Stellenangaben verkürzt werden sollen. [ 1 ]

Es können zwei Zahlenwerte angegeben werden:

1. Zahl: Seiten- und Zeilennummer  $\langle 0 \rangle$

0 = verkürzen bei gleichen Seiten-/Zeilennummern

1 = immer ausgeben

2. Zahl: Wortnummer  $\langle 0 \rangle$

0 = weglassen, falls ganze Zeilen betroffen

1 = immer ausgeben

Jede Korrekturanweisung wird als ein Satz in die KORREKTUR-Datei ausgegeben. Beginnt innerhalb des Korrekturtextes eine neue Zeile in Version B und sollen solche Zeilenwechsel in der Korrekturanweisung erkennbar bleiben, so kann mit dem Parameter KFZ verlangt werden, dass die Korrekturanweisung bei jedem Zeilenwechsel der Version B aufgeteilt wird.

**KFZ** Angabe, ob für Korrekturanweisungen, deren Korrekturtext einen Zeilenwechsel umfasst, Fortsetzungszeilen erzeugt werden sollen. [ 1 ]

$\langle 0 \rangle$

0 = keine Erzeugung von Fortsetzungszeilen

1 = Fortsetzungszeilen werden entsprechend der Zeilenaufteilung der Version B erzeugt

Wird eine Korrekturanweisung in mehrere Zeilen aufgeteilt, so enthalten die zusätzlich entstehenden Sätze in der KORREKTUR-Datei den Korrekturschlüssel (nur falls der Parameter SW angegeben ist), das Zeichen »+« (als Kennzeichen für die Fortsetzung einer Korrekturanweisung) und die Fortsetzung des Korrekturtextes.

Sollen die Korrekturanweisungen in erweiterter Form ausgegeben werden, sind zu den folgenden Parametern die entsprechenden Angaben erforderlich. Welche Angaben eine Korrekturanweisung enthalten kann, ist dem Abschnitt »Aufbau eines Datensatzes in der KORREKTUR-Datei« zu entnehmen.

Damit Korrekturanweisungen, die beim Vergleich eines Grundtextes mit verschiedenen Textversionen erzeugt wurden, auch nach dem Zusammenmischen der richtigen Version zugeordnet werden können, können die Korrekturanweisungen mit einem Kennzeichen versehen werden. Dieses Kennzeichen muss mit dem Parameter VKZ angegeben werden.

**VKZ**           Textteil, mit dem die Korrekturanweisungen gekennzeichnet werden. [ II ]

Der hier angegebene Textteil wird, eingeschlossen in runden Klammern, nach der Stellenangabe des von der Korrekturanweisung betroffenen Textes in Version A eingefügt. Der Textteil darf keine runden Klammern enthalten.

Soll der von der Korrekturanweisung betroffene Text aus Version A (»Originalwortlaut«) und evtl. seine Umgebung (»Kontext«) in die Korrekturanweisung eingesetzt werden, so kann das durch entsprechende Angaben zum Parameter UMG erreicht werden.

**UMG**           Angaben zur Ausgabe des Originalwortlautes und des Kontextes. [ I ]

Es können insgesamt neun Zahlenwerte angegeben werden.

Drei Zahlenwerte für Korrekturanweisungen der Korrekturart Löschen:

1. Zahl: Kontext vor dem Originalwortlaut <0>

Anzahl der Wörter, die vor dem Originalwortlaut in die Korrekturanweisung eingesetzt werden sollen.

2. Zahl: Originalwortlaut <0>

Maximale Anzahl der Wörter, die vom Originalwortlaut in die Korrekturanweisung eingesetzt werden sollen. Besteht der Originalwortlaut aus mehr Wörtern, so wird er durch Weglassen von Wörtern aus der Mitte entsprechend gekürzt. Die weggelassenen Wörter werden durch » . . . . « ersetzt, sofern mit dem Parameter UMK nichts anderes angegeben ist.

3. Zahl: Kontext nach dem Originalwortlaut <0>

Anzahl der Wörter, die nach dem Originalwortlaut in die Korrekturanweisung eingesetzt werden sollen.

Drei Zahlenwerte für Korrekturanweisungen der Korrekturart Ersetzen:

4. Zahl: Analog zur 1. Zahl <Wert der 1. Zahl>

5. Zahl: Analog zur 2. Zahl <Wert der 2. Zahl>

6. Zahl: Analog zur 3. Zahl <Wert der 3. Zahl>

Drei Zahlenwerte für Korrekturanweisungen der Korrekturart Einfügen:

7. Zahl: Analog zur 1. Zahl <Wert der 1. Zahl>

8. Zahl: Ohne Bedeutung (Originalwortlaut = 0 Wörter)

9. Zahl: Analog zur 3. Zahl <Wert der 3. Zahl>

Der Originalwortlaut wird vom Kontext, sofern dieser mit dem Parameter UMG verlangt wurde, in der Korrekturanweisung durch » : : « abgegrenzt. Wird der Originalwortlaut in der Korrekturanweisung gekürzt (vgl. Parameter UMG), so wird anstelle der weggelassenen Wörter » . . . . « eingesetzt. Will man diese Voreinstellungen ändern, müssen mit dem Parameter UMK die gewünschten Zeichenfolgen angegeben werden.

**UMK** Textteile, die den Kontext abgrenzen, bzw. die eingesetzt werden sollen, falls der Originalwortlaut gekürzt werden muss. [ II ]

</ : : / . . . . / : : />

Der erste angegebene Textteil trennt den Originalwortlaut nach links, der dritte angegebene Textteil trennt ihn nach rechts vom Kontext ab.

Soll auf Grund der Angaben zum Parameter UMG vor bzw. nach dem Originalwortlaut kein Kontext ausgegeben werden, werden auch der erste bzw. dritte Textteil nicht in die Korrekturanweisung eingesetzt.

Der zweite angegebene Textteil wird dann in die Korrekturanweisung eingesetzt, wenn der Originalwortlaut gekürzt werden muss.

Die Stellenangabe für den von der Korrekturanweisung betroffenen Text aus Version B (Korrekturtext) wird nur in die Korrekturanweisung eingesetzt, wenn dies mit dem folgenden Parameter verlangt wird. Die Ausgabe dieser Stellenangabe erlaubt, sie später (z. B. bei der Erstellung von entsprechenden Registern) als Referenz zu benutzen.

**STB** Angabe, ob die Stellenangabe für den Text aus der Version B (Korrekturtext) in die Korrekturanweisung eingesetzt werden soll. [ I ] <0>

0 = Stellenangabe nicht einsetzen

1 = Stellenangabe einsetzen

Die Stellenangabe für den Korrekturtext wird, in spitzen Klammern eingeschlossen, in der gleichen Form wie die Stellenangabe für den Originalwortlaut vor dem Korrekturzeichen eingesetzt.

Der Korrekturschlüssel wird nur in die Korrekturanweisung eingesetzt, wenn der Parameter **SW** angegeben ist. Er wird benötigt, wenn die Korrekturanweisungen sortiert werden sollen, oder wenn sie mit dem Kommando **#VAUFBEREITE** zum Drucken aufbereitet werden sollen.

**SW** Sortierwert. [I] <0>

Als Sortierwert wird eine Zahl von 0 bis 99 erwartet. Er ist Bestandteil des Korrekturschlüssels und kann neben anderen Sortierkriterien dazu benutzt werden, die Korrekturanweisungen beim Sortieren in die gewünschte Reihenfolge zu bringen.

Die Angabe des Parameters bewirkt in jedem Fall das Einsetzen des Korrekturschlüssels in die Korrekturanweisung; der Aufbau des Korrekturschlüssels ist unter »Aufbau des Korrekturschlüssels« auf Seite 1142 beschrieben.

## Alphabetisches Verzeichnis der Parameter

<b>ABK</b>	Definition von Abkürzungszeichen . . . . .	1140
<b>ASP</b>	Angabe von Aufsatzpunkten bzw. Vergleichsbereichen . . . . .	1137
<b>BER</b>	Auswahl eines Bereichs aus Version A und B . . . . .	1126
<b>DR</b>	Druckausgabesteuerung . . . . .	1129
<b>DRT</b>	Druckertyp . . . . .	1131
<b>DRZ</b>	Druckausgabesteuerung – zusätzliche Angaben . . . . .	1129
<b>FT</b>	Fußtext . . . . .	1131
<b>GLZ</b>	Zeichen-Tabelle für den Vergleich . . . . .	1139
<b>GKU</b>	Groß- und Kleinbuchstaben unterscheiden . . . . .	1128
<b>IGN</b>	Definition von zu ignorierenden Zeichen . . . . .	1139
<b>KBA</b>	Kennung von Bereichsangaben . . . . .	1138
<b>KFZ</b>	Korrekturfolgezeilen . . . . .	1132
<b>KT</b>	Kopfertext . . . . .	1130
<b>NGZ</b>	Nur gleiche Wörter zuordnen. . . . .	1129
<b>PAR</b>	Parameter-Konvention . . . . .	1125
<b>PR</b>	Angabe zum Umfang des Protokolls . . . . .	1131
<b>STB</b>	Stellenangabe des Korrekturtextes . . . . .	1134
<b>STR</b>	Angaben zur Silbentrennung . . . . .	1137
<b>SW</b>	Sortierwert . . . . .	1135
<b>SZW</b>	Verkürzung der Stellenangaben . . . . .	1132
<b>UMG</b>	Umgebung des Originalwortlautes (Kontext) . . . . .	1133
<b>UMK</b>	Kennzeichen zur Abgrenzung der Umgebung . . . . .	1134
<b>VB</b>	Angabe von Vergleichsbereichen . . . . .	1126
<b>VKZ</b>	Versionskennzeichen . . . . .	1133
<b>WTV</b>	Wörter tauschen zum Vergleichen . . . . .	1127
<b>WTZ</b>	Wörter tauschen zum Zuordnen . . . . .	1129
<b>XV</b>	Ersetzen von Zeichenfolgen zum Vergleichen . . . . .	1127
<b>XVX</b>	Ersetzen von Zeichenfolgen zum Vergleichen . . . . .	1127
<b>XZ</b>	Ersetzen von Zeichenfolgen zum Zuordnen . . . . .	1128
<b>XZX</b>	Ersetzen von Zeichenfolgen zum Zuordnen . . . . .	1128

## Kompatibilitätsmodi mit speziellen Parametern

Um Aufwärtskompatibilität zu gewährleisten, stehen auch noch die Modi der Vorgängerversion zur Verfügung. Diese Modi und die dazugehörigen speziellen Parameter sind nachfolgend beschrieben. Diese Parameter sind bei den Modi WORT und ZEILE nicht erlaubt.

MODUS	= T	Textunterschiede protokollieren; gleichwertig mit MODUS=+; -.
	= K	Korrekturanweisungen protokollieren, sonst wie bei MODUS=-; -.
	= x; y	Für x ist Minus oder Plus anzugeben: <ul style="list-style-type: none"> <li>- Nur Zeilen protokollieren, in denen Unterschiede festgestellt werden.</li> <li>+ Wie »-«; falls sich jedoch das erste bzw. letzte Wort einer Zeile der Version A von dem entsprechenden Wort der Version B unterscheidet, auch die vorangehende bzw. nachfolgende Zeile protokollieren.</li> </ul> <p>Für y ist Minus, Plus oder ! anzugeben:</p> <ul style="list-style-type: none"> <li>- Zeilenwechsel unberücksichtigt lassen.</li> <li>+ Zeilenwechsel als Orientierungshilfe verwenden.</li> <li>! Zeilenwechsel als Orientierungshilfe verwenden und zur weiteren Orientierung nur gleichlautende Wörter berücksichtigen.</li> </ul> <p>Wird die Angabe ;y weggelassen, so wird zunächst versucht, die Zeilenwechsel in den Eingabedateien zu berücksichtigen. Können damit keine Übereinstimmungen erzielt werden, bleiben die Zeilenwechsel unberücksichtigt.</p>
	= . . .	Druckertyp, für den das Protokoll mit den Textunterschieden (wie bei MODUS=T) aufbereitet werden soll. Er kann auch über Parameter angegeben werden. <p>Mit dem Kommando #LISTE, DRUCKERTYPEN können die möglichen Druckertypen aufgelistet werden.</p>



## Angaben zur Silbentrennung

**STR** Angabe zur Silbentrennung. [ I ] <1>

Es kann ein Zahlenwert angegeben werden:

0 = Eingabedaten enthalten keine Silbentrennung

1 = Silbentrennung bei der Eingabe aufheben

In den Eingabedaten gilt als Silbentrennzeichen ein »-«, das (nachdem abschließende Leerzeichen entfernt wurden) als letztes Zeichen in einem Eingabesatz steht, wenn das zweitletzte Zeichen ebenfalls ein »-« ist, oder wenn das zweitletzte Zeichen ein Buchstabe und das drittletzte Zeichen kein Steuerzeichen (\$, &, @, \, \_, #, %) ist.

Beim Aufheben der Silbentrennung wird ein getrenntes »ck«, das als »k-« und »k« geschrieben ist, nicht wieder zu »ck«.

## Auswählen der Daten

Soll der gesamte Inhalt beider Dateien in der bestehenden Reihenfolge verglichen werden, braucht keiner der folgenden Parameter angegeben zu werden.

Soll in beiden Versionen nur ein bestimmter Bereich bzw. sollen beide Versionen nur ab einer bestimmten Stelle verglichen werden, und sind die Bereichs- bzw. Stellenangaben in beiden Versionen identisch, kann der Bereich bzw. die Stelle mit dem Parameter **BER** (siehe Seite 1126) angegeben werden. Sind die Bereichs- bzw. Stellenangaben für den zu vergleichenden Text in den beiden Versionen nicht identisch, muss die Auswahl mit dem Parameter **ASP** angegeben werden.

Die Angabe von bestimmten Stellen (Aufsatzpunkten) kann auch sinnvoll sein, wenn zwar der gesamte Inhalt beider Dateien verglichen werden soll, aber wegen großer Unterschiede, insbesondere wegen größerer Auslassungen oder Einfügungen, ein großer Zeitaufwand zum Auffinden paralleler Stellen oder eine nur ungenaue Zuordnung der gefundenen Unterschiede zu erwarten ist.

**ASP** Angabe von Aufsatzpunkten bzw. von zu vergleichenden Bereichen. Mehrere Angaben sind möglich und müssen durch Apostroph getrennt werden. [ XI ]

Eine vollständige Angabe für einen in beiden Versionen zu vergleichenden Bereich besteht aus der Bereichsangabe für Version A und der Bereichsangabe des entsprechenden Bereichs in Version B, die durch »=« verbunden sind.

Eine Bereichsangabe besteht aus einer Anfangsposition und, mit einem Minuszeichen damit verbunden, einer Endposition in der Form  $s.z[/u] [ , w ] - s.z[/u] [ , w ]$ , wobei  $s$  für die (bis zu 6-stellige) Seitennummer,  $z$  für die (bis zu 3-stellige) Zeilennummer,  $u$  für die (bis zu 3-stellige) Unterscheidungsnummer und  $w$  für die (bis zu 2-stellige) Wortnummer steht; die in [ ] eingeschlossenen Teile können fehlen.

Die Anfangsposition ist jeweils das erste Wort, das zum Bereich gehört; die Endposition ist jeweils das letzte Wort, das noch zum Bereich gehört.

Sind die Bereichsangaben für die Version A und die Version B identisch, so kann die Angabe für die Version B einschließlich des davor stehenden »=« entfallen.

Schließt bei zwei aufeinander folgenden Bereichsangaben zur gleichen Version der zweite Bereich unmittelbar an den ersten an, so kann die Angabe der Endposition in der ersten Bereichsangabe einschließlich des davor stehenden »-« entfallen. Ebenso kann die Endposition der letzten Bereichsangabe weggelassen werden, wenn sich der Bereich bis zum Dateiende erstreckt. Solche Angaben ohne Endposition heißen Aufsatzpunkte.

Sind nur Aufsatzpunkte bzw. durch »=« verbundene Paare von Aufsatzpunkten angegeben, so werden die beiden Versionen nicht vom ersten Aufsatzpunkt an, sondern vom Dateianfang an verglichen. Dabei werden die beiden Dateien in unmittelbar aneinander anschließende, jeweils nacheinander zu vergleichende Bereiche unterteilt. An jedem angegebenen Aufsatzpunkt endet bzw. beginnt ein solcher Bereich. Dies kann aus den oben genannten Gründen sinnvoll sein, obwohl beide Versionen von Anfang bis Ende miteinander verglichen werden sollen.

Die Bereichsangaben werden in Form einer Korrekturanweisung (Korrekturart = Kommentar, Korrekturzeichen = »\*«) vor den diesen Bereich betreffenden Korrekturanweisungen in die KORREKTUR-Datei ausgegeben.

Dieser Parameter ist nur zugelassen, wenn die Satznummern in beiden Dateien aufsteigend sind; außerdem schließen sich die Parameter BER und ASP gegenseitig aus.

Besteht der Text von Version B nur aus Fragmenten, die mit den entsprechenden Bereichen der Version A verglichen werden sollen, kann es zweckmäßig sein, diese Bereiche nicht über Parameter, sondern direkt im Text von Version B anzugeben.

**KBA** Zeichenfolgen, die in Version B die Bereichsangaben kennzeichnen.  
[ IX ]

Die Bereichsangaben müssen unmittelbar hinter einer der mit dem Parameter KBA angegebenen Zeichenfolge stehen; sie haben die gleiche Form wie beim Parameter ASP; folgt in der gleichen Zeile noch Text, so muss hinter der Bereichsangabe noch ein Leerzeichen stehen. Mit den auf diese Weise angegebenen Bereichen der Version A wird der der Bereichsangabe folgende Text der Version B bis unmittelbar vor die nächste Bereichsangabe verglichen; explizite Bereichsangaben für den Text von Version B entfallen.

Es ist darauf zu achten, dass der gesamte Text von Version B anhand solcher Bereichsangaben entsprechenden Bereichen der Version A zugeordnet werden muss. Unmittelbar am Anfang von Version B muss also bereits eine solche Bereichsangabe stehen.

Die Bereichsangaben werden in Form einer Korrekturanweisung (Korrekturart = Kommentar, Korrekturzeichen = »\*«) vor den diesen Bereich betreffenden Korrekturanweisungen in die KORREKTUR-Datei ausgegeben. Hinter dem Korrekturzeichen »\*« wird angegeben, als wievielte Zeichenfolge das Kennzeichen für diese Bereichsangabe mit dem Parameter `KBA` angegeben ist.

## Angaben zum Vergleich

Treten beim Vergleich zweier Textversionen Unterschiede auf, so muss das Programm einerseits die Abgrenzung größerer Einfügungen bzw. Auslassungen vornehmen, andererseits die einander entsprechenden Textteile aus beiden Versionen, trotz der darin enthaltenen Unterschiede, einander möglichst wortgenau zuordnen.

Sind die Unterschiede durch orthographische Eigenheiten der zu vergleichenden Versionen oder durch Verwendung von abkürzenden Schreibweisen bedingt, so kann dem Programm eine bessere Zuordnung des Textes der beiden zu vergleichenden Versionen durch Angaben zu den folgenden Parametern ermöglicht werden.

Dabei können Zeichen angegeben werden, die als gleichwertig gelten sollen, solche, die ignoriert werden sollen und solche, die als Abkürzungszeichen gelten sollen. Auch die nur auf diesen Zeichen beruhenden Unterschiede werden jedoch ausgegeben.

**GLZ** Zeichen bzw. Zeichengruppen. [ VI ]

Die Zeichen, die als gleichwertig behandelt werden sollen, müssen zuvor jeweils als eine Zeichengruppe (mit Parametern der Art [ V ] ) definiert werden. Die Kennungen der einzelnen Zeichengruppen müssen mit dem Parameter `GLZ` angegeben werden.

Sollen z. B. einerseits i und y, andererseits c, g, k und q als gleichwertig gelten, so muss eine Zeichengruppe, die i und y enthält, und eine andere Zeichengruppe, die c, g, k und q enthält, definiert werden. Dadurch, dass die Kennungen dieser Zeichengruppen mit dem Parameter `GLZ` angegeben werden, gelten jeweils einerseits die Zeichen i und y und andererseits die Zeichen c, g, k und q als gleichwertig.

Darüber hinaus können mit diesem Parameter Zeichen angegeben werden, die ignoriert werden sollen und solche, die als Abkürzungszeichen gelten sollen. In diesem Fall ist zusätzlich eine entsprechende Angabe mit dem Parameter `IGN` bzw. `ABK` erforderlich.

**IGN** Zahlenwert, der angibt, als wievieltens Zeichen mit dem Parameter `GLZ` das Zeichen (oder die Kennung der Zeichengruppe) angegeben ist, das beim Vergleichen ignoriert werden soll. Kennungen von Zeichengruppen werden dabei als ein Zeichen gezählt. [ I ]

Beim Parameter `IGN` wird nur eine Zahl erwartet. Deshalb müssen, wenn mehr als ein Zeichen ignoriert werden soll, alle zu ignorierenden Zeichen zuvor als eine Zeichengruppe definiert werden und die Kennung dieser Zeichengruppe dann mit dem Parameter `GLZ` angegeben werden.

**ABK**

Zahlenwert, der angibt, als wievielttes Zeichen mit dem Parameter `GLZ` das Zeichen (oder die Kennung der Zeichengruppe) angegeben ist, das beim Vergleichen als Abkürzungszeichen gelten soll. Kennungen von Zeichengruppen werden dabei als ein Zeichen gezählt. [ 1 ]

Beim Parameter `ABK` wird nur eine Zahl erwartet. Deshalb müssen, wenn mehr als ein Zeichen als Abkürzungszeichen gelten soll, alle als Abkürzungszeichen verwendeten Zeichen zuvor als eine Zeichengruppe definiert werden und die Kennung dieser Zeichengruppe dann mit dem Parameter `GLZ` angegeben werden.

## Aufbau eines Datensatzes in der KORREKTUR-Datei

```
KS StA (VKZ) n [ KL :: OWL :: KR ] <StB> KZ KTxt
```

```

          ***                               ** ****
SW      ++
VKZ          +++++
UMG          ++++++
STB                               +++++

```

\*\*\* Daten, die auf jeden Fall erzeugt werden  
 +++ Daten, die bei Angabe des links angegebenen  
 Parameters erzeugt werden

Abk.      Bedeutung

KS	Korrekturschlüssel
StA	Stellenangabe in Version A
VKZ	Kennzeichnung der Korrekturanweisung
n	Anzahl der "]" im Kontext u. Originalwortlaut
KL	Kontext links vom Originalwortlaut
OWL	Originalwortlaut (aus Version A)
KR	Kontext rechts vom Originalwortlaut
StB	Stellenangabe in Version B
KZ	Korrekturzeichen
KTxt	Korrekturtext (aus Version B)

Der Originalwortlaut samt Kontext wird in eckige Klammern eingeschlossen. Enthält dieser Text eine schließende eckige Klammer, gibt die Zahl *n* vor der öffnenden eckigen Klammer, die den Kontext einleitet, die Anzahl der in Kontext und Originalwortlaut enthaltenen schließenden eckigen Klammern an, um die eckige Klammer, die den Kontext abschließt, eindeutig erkennen zu können. Enthält der Kontext und der Originalwortlaut keine schließende eckige Klammer, entfällt die Angabe *n*.

Folgt die Zahl *n* unmittelbar auf die Stellenangabe *StA* (das ist dann der Fall, wenn keine in runde Klammern eingeschlossene Kennzeichnung ergänzt wird; vgl. Parameter *VKZ* Seite 1133), wird vor die Angabe *n* die Zeichenfolge » ( ) « eingefügt, um die davor stehende Stellenangabe von der Angabe *n* abzugrenzen.

Statt der Zeichenfolge » : : « zwischen dem Kontext und dem Originalwortlaut kann jede andere Zeichenfolge gewählt werden (vgl. Parameter *UMK*, Seite 1134).

## Aufbau des Korrekturschlüssels

Der Korrekturschlüssel ist insgesamt 44 Zeichen lang und ist wie folgt aufgebaut:

APO	EPO	KA	SW	FNR	POS
-----	-----	----	----	-----	-----

APO	1	17	Anfangsposition des Korrekturtextes
	1	6	Seitennummer
	7	3	Zeilennummer
	10	3	Unterscheidungsnummer
	13	2	Wortnummer
	15	3	Zeichennummer
EPO	18	17	Endposition des Korrekturtextes
	18	6	Seitennummer
	24	3	Zeilennummer
	27	3	Unterscheidungsnummer
	30	2	Wortnummer
	32	3	Zeichennummer
KA	35	2	Korrekturart:
	35	1	0 = Fehler 1 = Seite-Zeile-Wort-Zeichen 2 = Seite-Zeile-Wort 3 = Seite-Zeile
	36	1	0 = Fehler 1 = Kommentar (*) 2 = Löschen (-) 3 = Ersetzen (=) 4 = Einfügen (+)
SW	37	2	Sortierwert (Versionsnummer)
FNR	39	3	Folgenummer (für Fortsetzungszeilen)
POS	42	3	Position des Korrekturzeichens (*, -, =, +) in der Korrekturanweisung

Die erste Zahl gibt jeweils die Zeichenposition innerhalb des Korrekturschlüssels an, die zweite Zahl die Anzahl der Zeichen.

Die Zeichennummer wird vom Kommando #VERGLEICHE nicht benutzt; sie ist für den Aufbau des Korrekturschlüssels durch das Kommando #SVORBEREITE vorgesehen.

\* \* \* \* \*

**#SATZ**

Kommando . . . . .	1147
Leistung . . . . .	1148
Anmerkung zu den Dateien . . . . .	1149
Zur Steuerung des Satzprogramms . . . . .	1150
Parameter . . . . .	1152
Identifikation des Satzauftrags . . . . .	1152
Angaben zum Protokoll . . . . .	1153
Auswahl der Daten . . . . .	1153
Schriften . . . . .	1154
Schriftgrade für die einzelnen Textteile . . . . .	1161
Satzspiegel . . . . .	1163
Seiten- und Spaltenmontage . . . . .	1167
Zeilenumbruch und Silbentrennung . . . . .	1170
Marginalien . . . . .	1173
Lebende Kolumnentitel . . . . .	1176
Abschnittsgrenzen . . . . .	1178
Einschaltungen . . . . .	1179
Zwischenüberschriften (Titelzeilen) . . . . .	1180
Fußnoten . . . . .	1183
Apparate . . . . .	1188
Makros . . . . .	1189
Umdefinition von Zeichen und Dickten . . . . .	1193
Übersicht über die Voreinstellungen . . . . .	1197
Alphabetisches Verzeichnis der Parameter . . . . .	1198
Steueranweisungen . . . . .	1200
I. Unterteilung des Textes	
1.    Seitenumbruch . . . . .	1201
2.    Kolumnentitel; Spaltenkopftext . . . . .	1205
3.    Titelzeilen (Rubriken, Zwischenüberschriften) . . . . .	1210
4.    Einschaltungen . . . . .	1211
4.1.    Einspaltige Einschaltungen (»Petit-Satz«) . . . . .	1211
4.2.    Mischen von ein- und mehrspaltigem Blocksatz . . . . .	1211
4.3.    Verändern der Satzbreite . . . . .	1212
5.    Kritische Apparate . . . . .	1214
6.    Fußnoten . . . . .	1215
7.    Absätze, Blindzeilen, Freiraum; Einbinden von Grafiken . . . . .	1217
7.1.    Absätze . . . . .	1217
7.2.    Blindzeilen . . . . .	1218
7.3.    Freiraum; Einbinden von Grafiken . . . . .	1219
7.4.    Vertikaler Tabulator; Ändern von Spaltenhöhe und Durchschuss . . . . .	1226



## II. Gestaltung der Zeilen

8.	Zeilenumbruch; Spatien . . . . .	1231
9.	Silbentrennung . . . . .	1235
10.	Einrücken; Zentrieren; Tabellen; Marginalien . . . . .	1238
10.1.	Einzüge, Einrückungen . . . . .	1238
10.2.	Zentrieren . . . . .	1241
10.3.	Tabellen- und Spaltensatz . . . . .	1243
10.4.	Merkstellen; Positionieren; Textfelder . . . . .	1244
10.5.	Zeilenzähler; Marginalien . . . . .	1248

## III. Die darzustellenden Zeichen

11.	Auszeichnungen . . . . .	1251
11.1.	Auszeichnungsschriften; Sperrung . . . . .	1251
11.1.1.	Auszeichnung über mnemonische Anweisungen . . . . .	1252
11.1.2.	Auszeichnung über Anwahl des Umschaltbereichs . . . . .	1255
11.2.	Über-, Durch-, Unterstreichung, Unterpunktierung . . . . .	1257
11.3.	Änderung von Schriftgröße und Zeilenabstand . . . . .	1259
11.4.	Wahl der Druckfarbe . . . . .	1261
12.	Akzente, diakritische Zeichen, Sonderzeichen . . . . .	1263
12.1.	Akzente . . . . .	1263
12.1.1.	Akzente über den Buchstaben . . . . .	1263
12.1.2.	Akzente unter den Buchstaben . . . . .	1264
12.2.	Umlaute, Sonderbuchstaben, Ziffern, Ligaturen . . . . .	1265
12.3.	Hochgestellte Buchstaben und Ziffern . . . . .	1268
12.3.1.	Hochgestellte Einzelzeichen . . . . .	1268
12.3.2.	Hochgestellte Bereiche mit Größenänderung . . . . .	1268
12.3.3.	Hochgestellte Bereiche ohne Größenänderung . . . . .	1269
12.4.	Übersetzte Zeichen . . . . .	1269
12.5.	Tiefgestellte Zeichen . . . . .	1270
12.5.1.	Tiefgestellte Einzelzeichen . . . . .	1270
12.5.2.	Tiefgestellte Bereiche mit Größenänderung . . . . .	1270
12.5.3.	Tiefgestellte Bereiche ohne Größenänderung . . . . .	1271
12.6.	Untergesetzte Zeichen . . . . .	1271
12.7.	Satzzeichen, Sonderzeichen . . . . .	1271
12.8.	Sonstige Zeichen und Symbole . . . . .	1275
12.8.1.	Mit #(name) codierte Sonderzeichen . . . . .	1275
12.8.2.	Römische Zahlen aus arabischen Zahlen . . . . .	1276
12.8.3.	Zeichenadresse aus dem Encoding von Type-1-Fonts . . . . .	1276
12.8.4.	Zeichenadressierung über Zeichennamen . . . . .	1277
12.8.5.	Unicode-Zeichen 2000 bis 202F (»General Punctuation«) . . . . .	1277
12.9.	PostScript-Grafiken . . . . .	1278
12.10.	Spiegeln und Drehen von Zeichen . . . . .	1279
12.11.	Linien, Punktreihen . . . . .	1279
13.	Griechische Schrift . . . . .	1284

---

14.	Koptische Schrift . . . . .	1286
15.	Hebräische Schrift . . . . .	1287
16.	Arabische Schrift . . . . .	1289
17.	Russische Schrift . . . . .	1293
18.	Phonetische Zeichen . . . . .	1294
IV. Generelle Anweisungen		
19.	Makros . . . . .	1295
20.	Kommentar . . . . .	1296
21.	Verknüpfungen für PDF-Dateien . . . . .	1297
21.1.	Verknüpfung mit Seiten . . . . .	1297
21.2.	Verknüpfung mit benannten Zielen (named destinations) . . . . .	1298
21.2.1.	Definition eines benannten Zieles . . . . .	1298
21.2.2.	Verknüpfung mit einem benannten Ziel . . . . .	1299
21.3.	Verknüpfung zu einem Dokument im WWW . . . . .	1299
21.4.	Aufruf eines anderen Dokuments oder Programms . . . . .	1300
21.5.	Einfügen von Notizen . . . . .	1300
22.	Hardware-nahe Anweisungen (Interncode) . . . . .	1302
	Zeichencodierung in der Ausgabedatei . . . . .	1308
Anhang: Verzeichnis der Steueranweisungen . . . . .		1309

## Kommando

#SATZ

QUELLE	= <code>datei</code>	Name der Datei mit den Daten (mit Steueranweisungen), die gesetzt werden sollen.
	= <code>-STD-</code>	Die Daten (mit Steueranweisungen), die gesetzt werden sollen, stehen in der Standard-Text-Datei.
ZIEL	= <code>-</code>	* Keine Ausgabe der Daten mit den Steueranweisungen.
	= <code>datei</code>	Name der Datei, in die die Daten (mit den Steueranweisungen) in der neuen Seiten-Zeilen-Einteilung ausgegeben werden sollen.  Wird bei <code>MODUS=T</code> oder <code>MODUS=A</code> , durch Apostroph getrennt, eine zweite Datei angegeben, so wird in diese Datei der Text der Fußnoten ausgegeben.
	= <code>-STD-</code>	Die Daten (mit den Steueranweisungen) sollen in der neuen Seiten-Zeilen-Einteilung in die Standard-Text-Datei ausgegeben werden.
MODUS	= <code>T</code>	Die QUELL-Datei enthält die Daten für den Textteil des zu setzenden Werkes.
	= <code>F</code>	Die QUELL-Datei enthält nur die Daten für die Fußnoten des zu setzenden Werkes.
	= <code>A</code>	Die QUELL-Datei enthält Daten, deren Seiten- und Zeileneinteilung bereits feststeht und die (z. B. über das Standard-Makro <code>*AUMBRUCH</code> bzw. <code>*SUMBRUCH</code> erzeugte) entsprechende Steueranweisungen für Zwangs-Umbruch enthalten.
LOESCHEN	= <code>-</code>	* Enthält die ZIEL-, die AUSGABE- oder die PROTOKOLL-Datei schon Daten, so wird das Programm abgebrochen.
	= <code>+</code>	Daten in der ZIEL-, in der AUSGABE- und in der PROTOKOLL-Datei zuerst löschen.
PARAMETER	= <code>-</code>	* Keine Parameterangaben.
	= <code>datei</code>	Name der Datei mit den Parametern. Es können, durch Apostroph getrennt, zwei Dateien angegeben werden. Der Inhalt der zweiten Parameterdatei wird nach dem Inhalt der ersten Parameterdatei ausgewertet. Für Reihenfolge und Anzahl der Parameter gelten die gleichen Regeln wie wenn nur eine Parameterdatei angegeben ist.

	= *	Die Parameter folgen auf das Kommando und sind mit *EOF abgeschlossen.
AUSGABE	= -	* Keine Ausgabe von Steuercodes für die Satzausgabe.
	= <code>datei</code>	Name der Datei, in die die Steuercodes für die Satzausgabe ausgegeben werden sollen.
PROTOKOLL	= <code>-STD-</code>	* Das Protokoll mit den gesetzten Daten, der Umbruch-Information und den Fehlermeldungen soll in die Standard-Protokoll-Datei ausgegeben werden.
	= <code>datei</code>	Name der Datei, in die das Protokoll mit den gesetzten Daten, der Umbruch-Information und den Fehlermeldungen ausgegeben werden soll.  Wird bei <code>MODUS=T</code> oder <code>MODUS=A</code> , durch Apostroph vom ersten Dateinamen getrennt, eine zweite Datei angegeben, so werden in diese Datei die vorkommenden Zeilennummern und die Position der Schriftgrundlinie jeder Zeile (genauer: die am Zeilenende geltende Position der Schriftgrundlinie; Abstand vom oberen Satzspiegelrand; siehe auch die Erläuterung zur Anweisung <code>&amp;le</code> ) sowie (mit »Zeilennummer« 999) die Gesamthöhe jeder Spalte vor einem evtl. notwendigen Spaltenhöhenausgleich ausgegeben.
	= +	Das Protokoll mit den gesetzten Daten, der Umbruch-Information und den Fehlermeldungen soll ins Ablaufprotokoll ausgegeben werden.
	= -	Nur die Fehlermeldungen sollen ins Ablaufprotokoll ausgegeben werden.
FUSSNOTEN	= <code>datei</code>	Bei <code>MODUS=T</code> : Name der Datei mit den bereits gesetzten Fußnoten (= <code>AUSGABE</code> -Datei eines vorhergehenden Satzprogrammlaufs mit <code>MODUS=F</code> ).
	= -	* Keine Eingabe von bereits gesetzten Fußnoten.
SCHRIFTEN	= <code>datei</code>	Name einer (mit dem Standard-Makro <code>*PSFONT</code> erzeugten) Datei, in der die Dicktentabellen für benutzereigene Schriften stehen.
	= -	* Keine benutzereigenen Schriften.

## Leistung

Mit diesem Programm können Texte typographisch aufbereitet werden. Die Aufteilung des Textes auf Zeilen und Seiten (einschl. Silbentrennung, Randausgleich, Marginalien, Fußnoten, lebenden Kolumnentiteln usw.) erfolgt automatisch; sie kann über Steueranweisungen, die im Text enthalten sind, gesteuert werden.

Zur Kontrolle und Weiterverarbeitung der Ergebnisse des Satzprogramms dienen die weiter unten beschriebenen Standard-Makros für die Satzumgebung.

Die Satzausgabe kann mit Hilfe dieser Makros auf PostScript-Druckern oder (für professionellen Satz) auf PostScript-Belichtern erfolgen. Bezüglich der Verfügbarkeit von Fonts bzw. entsprechender Lizenzen für diese Geräte ist ggf. eine Abstimmung mit deren Betreibern erforderlich.

## Anmerkung zu den Dateien

QUELLE: Text-Datei, maximale Satzlänge (einschl. aufgelöster Makros der Gruppe A, vgl. unten S. 1189): 8000 Zeichen. Maximale Länge eines einzelnen Wortes (= Zeichenfolge zwischen Leerzeichen): 480 Zeichen einschließlich der darin enthaltenen aufgelösten Makros. Sind in einem Wort mit »<name>« codierte Makros der Gruppe A enthalten, die in der ZIEL-Datei und in der PROTOKOLL-Datei nicht aufgelöst werden, so darf die – nur programmintern vorgenommene – Auflösung dieser Makros nicht zu einer Gesamtwortlänge von mehr als 480 Zeichen führen.

Die QUELL-Datei enthält den Text, der gesetzt werden soll, einschließlich der zum Satz erforderlichen Steueranweisungen.

ZIEL: Text-Datei; Zeilen, die länger als 480 Zeichen sind, werden an einem Spatium unterteilt. Wird kein Spatium gefunden, wird dennoch unterteilt und die Warnung »Wort aufgespalten« ins Protokoll ausgegeben. Dateigröße wie QUELL-Datei, zuzüglich Platz für darin enthaltene, in der ZIEL-Datei aufgelöste Makros der Gruppe A.

In die ZIEL-Datei werden die Daten so ausgegeben, wie sie in der QUELL-Datei stehen, also einschließlich der in der QUELL-Datei enthaltenen Steueranweisungen (bezüglich der Makros vgl. jedoch die Beschreibung der Steueranweisungen ab S. 1200). Die Zeileneinteilung (einschließlich der Silbentrennungen) und die Seiten-Zeilen-Nummer richtet sich in der ZIEL-Datei jedoch nach dem vom Satzprogramm errechneten Umbruch.

In der (ggf. als zweite angegebenen) ZIEL-Datei, in die die Fußnoten ausgegeben werden, entspricht die Seitennummer der einzelnen Sätze der Nummer der Seiten bzw. Spalten, auf der die jeweiligen Fußnoten(zeilen) nach dem Umbruch stehen; die Zeilennummern werden, mit 501 beginnend, seiten- bzw. spaltenweise neu vergeben. Der Inhalt der Datei ist (mit Ausnahme der Satznummern) identisch mit dem Inhalt der ZIEL-Datei des Satzprogrammablaufs, mit dem die Fußnoten gesetzt wurden.

Die ZIEL-Datei kann z. B. als Korrekturgrundlage nach einem Probeausdruck dienen und hat dabei gegenüber der QUELL-Datei den Vorteil, dass im Editor die im ausgedruckten Ergebnis stehenden Seitennummern zum Auffinden der zu korrigierenden Textstellen verwendet werden können. Sie ist außerdem Grundlage für Register- und ähnliche Arbeiten, bei denen die endgültigen Seiten- und ggf. Zeilennummern des gesetzten und gedruckten Buches als Referenzen benötigt werden.

AUSGABE: Die AUSGABE-Datei (SEQ-Datei, keine RAN-Datei) enthält die vom Satzprogramm erzeugten anlagenunabhängigen Steuercodes für die Belichtung. Diese Datei kann mit dem Standard-Makro \*PSAUS für die Ausgabe auf PostScript-Dru-

ckern bzw. -Belichtern aufbereitet werden. Die AUSGABE-Datei sollte nur mit den von TUSTEP dafür vorgesehenen Standard-Makros weiterverarbeitet werden. Insbesondere dürfen die Satznummern der AUSGABE-Datei nicht verändert werden.

PROTOKOLL: In der PROTOKOLL-Datei stehen die gleichen Daten wie in der ZIEL-Datei (bei MODUS=T auch die Fußnoten, falls solche vorhanden sind) in der gleichen Seiten- und Zeileneinteilung, aufbereitet für die Ausgabe auf einem Zeilendrucker. Zusätzlich enthält die PROTOKOLL-Datei Angaben über die Höhen der einzelnen Seiten bzw. Spalten (genauer: die am Spaltenende erreichte Höhe), über Umfang und Art des Höhenausgleichs sowie Fehlermeldungen und weitere Informationen.

Wird eine zweite Datei zu PROTOKOLL angegeben, so werden in diese Datei mit der gleichen Satznummer wie in der ZIEL-Datei die vorkommenden Zeilennummern und die – nach einem gegebenenfalls durchgeführtem Spaltenhöhenausgleich – am Zeilenende geltende Position der Schriftgrundlinie (Abstand vom oberen Satzspiegelrand) jeder Zeile ausgegeben. Wird innerhalb einer Zeile oder am Zeilenende die Schriftgrundlinie verschoben (z. B. mit &!P(Vn) ), so wird in die zweite Protokoll-Datei die durch die Verschiebung erreichte vertikale Position der Schriftgrundlinie ausgegeben. Wird innerhalb der Zeile die Anweisung &!e ausgeführt, wird die beim letzten Aufruf von &!e in dieser Zeile geltende Position ausgegeben. Am Ende jeder Seite wird, mit Zeilennummer 999, die Gesamthöhe der Spalte vor einem gegebenenfalls durchgeführten Spaltenhöhenausgleich ausgegeben.

## Zur Steuerung des Satzprogramms

Das Satzprogramm wird durch zwei unterschiedliche Arten von Angaben gesteuert:

### 1. Auftragsbezogene Angaben

Satzspiegel, verwendete Schriften, Schriftgröße und Durchschuss für die einzelnen Textteile (Grundtext, Titelzeilen, Einschaltungen, Fußnoten, Kolummentitel), Art und Stellung der Kolummentitel und der Seitenzahlen etc. sind Angaben, die für einen ganzen (Teil-)Auftrag konstant bleiben. Sie werden dem Satzprogramm direkt über Parameter mitgeteilt und sind, da nicht im Text enthalten, auf einfache Weise jeweils veränderbar, um z. B. einen bestimmten Umfang eines Werkes zu erreichen oder die Satzeinrichtung veränderten Wünschen von Autoren/Herausgebern/Verlegern anzupassen.

### 2. Textbezogene Angaben

Die Gestaltung und Untergliederung des Textes geschieht ausschließlich anhand von Steueranweisungen, die in den Eingabedaten (maschinenlesbares »Manuskript«) selbst enthalten sein müssen. Die Zeileneinteilung der QUELL-Datei bleibt (außer nach der Steueranweisung &!U) ebenso unberücksichtigt wie in den Eingabedaten mit Hilfe von Leerzeichen schon durchgeführte Einrückungen, Tabellierungen usw.; ein oder mehrere aufeinander folgende Leerzeichen gelten als ein einziges Leerzeichen.

Die Steueranweisungen gliedern sich in

- I. Anweisungen zur Unterteilung des Textes in Abschnitte und deren Stellung innerhalb der Seite (Kapitel 1–7),

- II. Anweisungen zur Gestaltung der Zeilen (z. B.: Einrücken, Zentrieren) (Kapitel 8–10),
- III. Anweisungen für die Darstellung der Zeichen (z. B. Auszeichnungen, Sonderzeichen, nicht-lateinische Schriften) (Kapitel 11–17),
- IV. Generelle Anweisungen (Kapitel 19–22).

Anhand der auftragsbezogenen Parameter und der im Text enthaltenen Anweisungen umbricht das Programm den (ohne Rücksicht auf die spätere Zeilen- und Seiten-Einteilung) endlos geschriebenen Text selbständig in Zeilen gleicher Länge (Block-satz) und Seiten gleicher Höhe. Dabei wird darauf geachtet, dass die letzte Zeile eines Abschnitts (»Ausgangszeile«) nicht als erste in einer neuen Spalte steht, und dass die erste Zeile eines neuen Abschnitts oder eine Zwischenüberschrift nicht als letzte unten in einer Spalte steht; in der Setzersprache: Hurenkinder und Schusterjungen werden vermieden.

Es ist auch möglich, die automatischen Umbruch-Entscheidungen des Programms generell oder gezielt an einzelnen Stellen abzuschalten und den Zeilen- und Seiten-umbruch durch die im Text enthaltenen Anweisungen ausschließlich selbst zu bestimmen.

Ebenso ist es möglich, den automatischen Spaltenhöhenausgleich sowohl durch auftragsbezogene Parameter als auch durch im Text enthaltene Anweisungen zu beeinflussen.

Für den Höhenausgleich für zu kurz (z. B. vor Überschriften) oder zu lang (z. B. durch eingebrachte Hurenkinder) gewordene Spalten sind folgende Maßnahmen vorgesehen:

- Bei nicht ganz gefüllten Spalten kann verlangt werden, dass der gesamte zum Austreiben der Spalte fehlende Raum an einer besonders markierten Stelle eingefügt wird.
- Es wird versucht, bei bestimmten Vorschubanweisungen, die einen Anteil von als »variabel« bezeichnetem Vorschub enthalten (das sind die Vorschübe vor und nach Titelzeilen und Einschaltungen, vor und nach der Fußnotenlinie, und die mit  $\$ \$ \$ + n \$ \$ \$$  codierten Vorschübe), durch Veränderung dieses variablen Vorschubanteils die Spalte auf die verlangte Höhe zu bringen. Dabei werden die variablen Vorschübe um maximal 50% erweitert und um maximal 33% verringert. Die Freiräume vor und nach der Fußnotenlinie werden dabei als erste berücksichtigt.
- Wenn die zuvor genannte Maßnahme nicht ausreicht, wird durch Veränderung des Durchschusses zwischen den Textzeilen in 1/2-Punkt-Schritten, mit der letzten Textzeile beginnend, der restliche Raum ausgetrieben oder eingebracht.

## Parameter

Jeder Parameter ist einer Parameterart zugeordnet. Sie ist jeweils als römische Zahl in [ ] angegeben. Damit ist festgelegt, in welcher Form die Angaben vom Programm erwartet werden. Die einzelnen Parameterarten sind in der Beschreibung »TUSTEP-Grundlagen« im Kapitel »Parameter« definiert.

Für manche Parameter gibt es voreingestellte Werte, die eingesetzt werden, falls keine anderen Werte angegeben sind. Diese Voreinstellungen sind in < > angegeben.

Bei den meisten Parametern sind Zahlenwerte anzugeben. Jeder dieser Zahlenwerte ist in der Beschreibung der leichten Verständigung halber mit einem Namen bezeichnet. Die Zahlenwerte sind in der beschriebenen Reihenfolge anzugeben; die Namen der Zahlenwerte dürfen nicht mit angegeben werden.

Die Parameter können in beliebiger Reihenfolge angegeben werden. Falls jedoch einer der Parameter BRE, SEI, SPA, ZLN, MAL, MAR angegeben ist, müssen diese vor dem Parameter MON angegeben werden.

Eine Parameterzeile darf maximal 600 Zeichen lang sein. Fortsetzungszeilen sind nur bei den Parametern MAC, MAH und MAA vorgesehen.

Die Beschreibung berücksichtigt, dass das Satzprogramm auch mehrspaltigen Satz erzeugen kann. Deswegen ist dort, wo nicht die (bei mehrspaltigem Satz) bereits aus mehreren Spalten zusammenmontierte Seite gemeint ist, von »Spalten« die Rede. Bei einspaltigem Satz kann an diesen Stellen statt »Spalte« jeweils »Seite« gelesen werden.

## Identifikation des Satzauftrags

**PRO** Projekt-Identifikation: erscheint in der Spaltenüberschrift des Protokolls und der Belichtung. [ XI ] <Projekt>

Ist dieser Parameter nicht angegeben, so wird der voreingestellte oder der zuletzt mit dem Kommando #DEFINIERE, PROJEKT=name eingestellte Projektname eingesetzt.

**LAU** Behandlung des Satzprogrammablaufs. [ I ] <1 1 0>

ILLAUF 1 = »1. Durchlauf«  
(bei MODUS=T wie »2«, bei MODUS=A wie »3«).

2 = »2. Durchlauf«: lässt (bei MODUS=T), falls Apparate angegeben sind und die Spalte um mehr als den im 7. Wert des Parameters HOE angegebenen Wert (Voreinstellung: 30 Punkt) zu lang wird, in der nächsten Spalte entsprechend viel Platz frei; die zu lang gewordene Spalte wird nicht ausgeglichen. Über Parameter HOE kann ein anderer Wert als 30 Punkt angegeben werden. Vgl. außerdem den ersten Wert beim Parameter SIL.



- 3 = Wie 2, aber Platz in Folgespalten nicht frei lassen (bei MODUS=A kann gleichbedeutend 1 oder 2 angegeben werden).
- 4 = zu lang gewordene Spalten werden auch dann durch negativen Durchschuss verkürzt, wenn sie um mehr als den im 7. Wert des Parameters HOE angegebenen Wert zu lang geworden sind.
- ILIG 0 = Satz ohne f-Ligaturen, ch- und ck-Ligaturen und Unterschneidungen.
- 1 = Sofern in der QUELL-Datei codiert, werden f-Ligaturen (ff fi fl ft ffi ffl; diese Ligaturen sind nicht in allen Fonts verfügbar) und Ligaturen ch und ck benutzt sowie Unterschneidungen (Kleinbuchstaben ohne Oberlänge und A nach T V W Y und umgekehrt; Anführungszeichen vor T V W Y) vorgenommen.
- 2 = es werden nur Unterschneidungen vorgenommen.
- IFENT 0 = Keine Berücksichtigung von Character Entities
- 1 = Die Character-Entities `&amp;`, `&lt;`, `&gt;`, `&apos;`, `&quot;` werden als `&`, `<`, `>`, `'`, `"` gesetzt. Die in Kapitel 12.8.5. aufgeführten, mit `#[XXXX]` codierten Sonderzeichen können auch als hexadezimale Entities `&^#x2000;` bis `&^#x202F;` codiert sein; `&^#x017F;` wird wie `#.s (f)` und `&^#x0292;` wie `#.z (3)` interpretiert, `&^#x00A0;` (non-breaking space) wird wie `#._` behandelt.

## Angaben zum Protokoll

- DRT** Druckertyp, für den die PROTOKOLL-Datei aufbereitet werden soll.  
[ XI ] <PS-10>
- Mit dem Kommando #LISTE, DRUCKERTYPEN können die definierten Druckertypen aufgelistet werden.

## Auswahl der Daten

Soll die gesamte Datei verarbeitet werden, braucht keiner der folgenden Parameter angegeben zu werden.

- BER** Angabe eines Bereichs (»seite.zeile-seite.zeile«) oder einer Anfangsstelle (»seite.zeile«), falls nicht die ganze Datei verarbeitet werden soll. [ XI ] <0.0-999999.999/999>

Dieser Parameter ist nur zugelassen, wenn die Satznummern in der QUELL-Datei alle aufsteigend sind.

**FAN** Fußnotenanzug: Stelle (»seite.zeile«), ab der die (bereits gesetzten) Fußnoten berücksichtigt werden sollen. [ XI ] <0.0>

Die Angabe des Parameters FAN ist nur sinnvoll bei MODUS=T, wenn gleichzeitig eine Angabe zum Parameter BER gemacht wurde). Es ist die Seiten- und Zeilennummer der entsprechenden Fußnotenzeile aus dem Protokoll des Satzprogrammlaufs anzugeben, mit dem die Fußnoten gesetzt wurden.

**MAX** Angaben für Testzwecke, wie viele Sätze maximal eingelesen bzw. wie viele Zeilen bzw. Spalten maximal aufbereitet werden sollen. [ 1 ]

Es können drei Zahlenwerte angegeben werden:

NEIN maximal einzulesende Sätze <99999999>  
 NAUS maximal zu setzende Zeilen <99999999>  
 NSAUS maximal zu setzende Spalten <99999999>

**ABB** Zahl der erlaubten Fehlerkommentare. [ 1 ]

NBFZ Basis-Fehlerzahl <20>  
 NSFZ zusätzlich pro Spalte erlaubte Fehlerzahl <2>  
 NKOMZ maximale Zeichenzahl für Kommentare <1000>  
 NAPPZ maximale Zeichenzahl für Apparateinträge <2400>

Nach NBFZ + (NSFZ \* lfd. Nummer der Spalte) Fehlern wird der Lauf abgebrochen.

Wird die maximale Zeichenzahl für einen Kommentar oder einen Apparateintrag überschritten, wird der Kommentar bzw. Apparateintrag mit einer entsprechenden Fehlermeldung beendet.

**TXB** Textbeginn [ XI ] <fehlt>

Mit dem Parameter TXB kann ab Spalte 11 ein Tag angegeben werden, das den Textbeginn bezeichnet. Das Tag muss mit einem der Parameter MAC oder MAH definiert sein. Alles, was vor diesem Tag steht, gilt als Vorspann (Metadaten) und wird auf Seite 0 (möglichst mit gleicher Zeileneinteilung wie in der Quelldatei) ausgegeben.

## Schriften

**SCH** Zuordnung von Schriften zu sechzehn Umschaltbereichen [ XI ]

NRSCH1 Schriftnummer für Bereich 1 (Grundschrift) <31801>  
 NRSCH2 Schriftnummer für Bereich 2 (Anweisung }) <31802>  
 NRSCH3 Schriftnummer für Bereich 3 (Anweisung ++)<31804>  
 NRSCH4 Schriftnummer für Bereich 4 (Anweisung @@)<31803>  
 NRSCH5 Schriftnummer für Bereich 5 (Anweisung ##)<31901>  
 NRSCH6 Schriftnummer für Bereich 6 (Anweisung "" )<>  
 NRSCH7 Schriftnummer für Bereich 7 (Anweisung >>)<>  
 NRSCH8 Schriftnummer für Bereich 8 (Anweisung <<)<>

NRSCH9    Schriftnummer für Bereich 9 (Anweisung !! (09)) <>  
 NRSCH10   Schriftnummer für Bereich 10 (Anweisung !! (10)) <>  
 NRSCH11   Schriftnummer für Bereich 11 (Anweisung !! (11)) <>  
 NRSCH12   Schriftnummer für Bereich 12 (Anweisung !! (12)) <>  
 NRSCH13   Schriftnummer für Bereich 13 (Anweisung !! (13)) <>  
 NRSCH14   Schriftnummer für Bereich 14 (Anweisung !! (14)) <>  
 NRSCH15   Schriftnummer für Bereich 15 (Anweisung !! (15)) <>  
 NRSCH16   Schriftnummer für Bereich 16 (Anweisung !! (16)) <>

Negative Angabe der Schriftnummer schaltet die Silbentrennung für den entsprechenden Umschaltbereich aus.

Die für Bereich 1 angegebene Schrift gilt als Grundschrift. Sie gilt so lange als eingestellt, bis mit einer der in Kapitel 11.1 beschriebenen Steueranweisungen auf eine andere Schrift umgeschaltet wird. Die Grundschrift muss eine lateinische Schrift sein.

Werden Kapitälchen benutzt, so müssen diese in Bereich 3 liegen.

Ein »-« hinter der Schriftnummer verhindert Unterschneidungen (und Ligaturen) bei dieser Schrift.

Hinter der Schriftnummer kann in runden Klammern eines der Zeichen K / F A G H R T P D angegeben werden, um zu erreichen, dass im Zweifelsfall (z. B. wenn zu mehr als einem Umschaltbereich eine griechische Schrift angegeben ist) mit den Steueranweisungen #K+ #/+ #F+ #A+ #G+ #H+ #R+ #T+ #P+ #D+ auf die so gekennzeichneten Schriften umgeschaltet werden soll, falls die betreffende Umschaltung aus der Grundschrift heraus erfolgt. Steht hinter mehr als einer Schriftnummer das gleiche Zeichen, so gilt es nur für die letzte der so gekennzeichneten Schriftnummern. Ist hinter einer Schriftnummer (K) angegeben, so werden nicht nur die Kapitälchen, sondern alle zwischen #K+ und #K- stehenden Zeichen aus dieser Schrift genommen (vgl. auch unten zu Parameter KAP).

Bei PostScript-Schriften, bei deren Bereitstellung mit dem Makro \*PSFONT zur Spezifikation MEDLIG eine Datei mit Alternativziffern (normalerweise Mediaevalziffern) angegeben wurde, kann »+« hinter der Schriftnummer bzw. hinter der Klammer angegeben werden, um zu erreichen, dass als Standard-Ziffern für diese Schrift die Ziffern aus dem zu MEDLIG angegebenen Font verwendet werden. Die in der im Parameter SCH angegebenen Schrift enthaltenen Ziffern (normalerweise Tabellenziffern) werden damit zu Alternativziffern, die mit #.1 usw. zu codieren sind (vgl. Kapitel 12.2).

Schriften mit sehr großer Mittelhöhe (»x-Höhe«) können durch % hinter der Schriftnummer bzw. der Klammer markiert werden, um zu verhindern, dass übergesetzte Buchstaben zu niedrig und Akzente unter den Buchstaben zu hoch sitzen. Umgekehrt können Schriften mit sehr kleiner Mittelhöhe durch ! oder !! hinter der Schriftnummer markiert werden, um zu verhindern, dass Akzente unter dem Buchstaben zu tief sitzen; dabei verschiebt !! die Akzente weniger weit nach oben

als !. Schriften mit ungewöhnlich kleiner Versalhöhe können durch ; hinter der Schriftnummer markiert werden. Akzente über Versalien werden dann um ca 20% weniger weit nach oben verschoben.

Relativ steil stehende Kursivschriften können durch \ statt / in der Klammer hinter der Schriftnummer markiert werden, um zu verhindern, dass übergesetzte Buchstaben zu weit rechts und und Akzente unter den Buchstaben zu weit links stehen. Reicht dies nicht aus, weil die betreffende Kursivschrift ungewöhnlich steil steht, so können diese Schriften mit \\ in der Klammer markiert werden.

Nummern von Schriften, aus denen nur einzelne Zeichen mit dem Parameter BIL bzw. mit der Steueranweisung &! (##mmmm/nnn) oder &! (#Ummmm\zeichename\dicke) angesprochen werden, müssen nicht mit dem Parameter SCH angegeben werden.

#### Verfügbare Schriften

Alle im Folgenden aufgeführten lateinischen Schriften enthalten die Ligaturen fi und fl; die Ligaturen ff ffi ffl sind nur bei den Schriften vorhanden, die mit »=« markiert sind. Die Ligatur ft fehlt immer. Statt der (nicht vorhandenen) echten Kapitälchen werden außer bei den mit »\*« markierten Schriften Versalien in kleinerem Schriftgrad benutzt. Mediaevalziffern stehen bei den mit »+« markierten Schriften zur Verfügung.

30001	HEBKRUPP (= Hebräisch)
30101	Helvetica
30102	Helvetica-Oblique
30103	Helvetica-Bold
30104	Helvetica Kapitälchen
30105	Helvetica-BoldOblique
30151	Helvetica-Narrow
30152	Helvetica-Narrow-Oblique
30153	Helvetica-Narrow-Bold
30154	Helvetica-Narrow Kapitälchen
30155	Helvetica-Narrow-BoldOblique
30251	Courier
30252	Courier-Oblique
30253	Courier-Bold
30254	Courier Kapitälchen
30255	Courier-BoldOblique
30901	Bookman-Light
30902	Bookman-LightItalic
30903	Bookman-Demi
30904	Bookman Kapitälchen

30905	Bookman-DemiItalic
	Arabisch:
31001	NaskhOneQubic (= Arabisch)
31003	NaskhBoldOneQubic
31201	NewCenturySchlbk-Roman
31202	NewCenturySchlbk-Italic
31203	NewCenturySchlbk-Bold
31204	NewCenturySchlbk Kapitälchen
31205	NewCenturySchlbk-BoldItalic
31401	Garamond-Light
31402	Garamond-LightItalic
31403	Garamond-Bold
31404	Garamond Kapitälchen
31405	Garamond-BoldItalic
31451	AGaramond-Regular = +
31452	AGaramond-Italic = +
31453	AGaramond-Semibold = +
31454	AGaramondExp-Regular * +
31455	AGaramond-SemiboldItalic
31456	AGaramond-Bold
31459	AGaramond-BoldItalic
31601	Palatino-Roman
31602	Palatino-Italic
31603	Palatino-Bold
31604	Palatino Kapitälchen
31605	Palatino-BoldItalic
	Newton:
31701	NewtTU
31702	NewtTU-Italic
31703	NewtTU-Bold
31704	NewtTU Kapitälchen
31705	NewtTU-BoldItalic
	Newton Cyrillic:
31751	NewtTUC
31752	NewtTUC-Italic
31753	NewtTUC-Bold
	Newton Armenian:
31761	NewtAmTU (z. Zt. nur für &! (#mmmm/nnn) und &! (#Ummmm\zeichename\dickte))
31762	NewtAmTU-Italic (z. Zt. nur für &! (#mmmm/nnn) und &! (#Ummmm\zeichename\dickte))
31763	NewtAmTU-Bold (z. Zt. nur für &! (#mmmm/nnn) und &! (#Ummmm\zeichename\dickte))

	Newton Georgian:
31771	NewtGeTU (z. Zt. nur für &! (##mmmm/nnn) und &! (#Ummmm\zeichename\dickte))
31772	NewtGeTU-Italic (z. Zt. nur für &! (##mmmm/nnn) und &! (#Ummmm\zeichename\dickte))
	Newton Greek:
31781	NewtGrTU
31782	NewtGrTU-Italic
31783	NewtGrTU-Bold
31785	NewtGrTU-BoldItalic
31801	Times-Roman
31802	Times-Italic
31803	Times-Bold
31804	Times Kapitälchen
31805	Times-BoldItalic
31851	TimesTenCyr-Upright (= Russisch)
31852	TimesTenCyr-Inclined
31853	TimesTenCyr-Bold
31855	TimesTenCyr-BoldInclined
31901	Griechisch
31902	ZapfDingbats (Sonderzeichen für BIL und &! (##mmmm/nnn) und &! (#Ummmm\zeichename\dickte))
31903	Symbol (Sonderzeichen für BIL und &! (##mmmm/nnn) und &! (#Ummmm\zeichename\dickte))
31920	NewtonPhoneticTu
31921	Times-PhoneticIPA
31922	Times-PhoneticAlternate (Sonderzeichen für BIL und &! (##mmmm/nnn) und &! (#Ummmm\zeichename\dickte))
31931	LAstrologyPi-One (Sonderzeichen für BIL und &! (##mmmm/nnn) und &! (#Ummmm\zeichename\dickte))
31932	LAstrologyPi-Two (Sonderzeichen für BIL und &! (##mmmm/nnn) und &! (#Ummmm\zeichename\dickte))
31951	Koptisch
32001	TusLibertineO-Heb
32101	TusLibertineO = +
32102	TusLibertineOI (kursiv) = +
32103	TusLibertineOB (fett) = +
32104	TusLibertineO (Kapitälchen) * +

32105	TusLibertineOBI (fett-kursiv)
32111	TusBiolinumO = +
32112	TusBiolinumOI (kursiv) = +
32113	TusBiolinumOB (fett) = +
32114	TusBiolinumO (Kapitälchen) * +
32501	TusLibertineO-Greek
32502	TusLibertineO-Greek-Italic
32503	TusLibertineO-Greek-Bold
32511	TusBiolinumO-Greek
32512	TusBiolinumO-Greek-Italic
32513	TusBiolinumO-Greek-Bold
32601	TusLibertineO-Cyrillic
32602	TusLibertineO-Cyrillic-Italic
32603	TusLibertineO-Cyrillic-Bold
32611	TusBiolinumO-Cyrillic
32612	TusBiolinumO-Cyrillic-Italic
32613	TusBiolinumO-Cyrillic-Bold

Von diesen Schriften gehören die Schriftfamilien Helvetica (Nummern 30101, 30102, 30103, 30105), Courier (Nummern 30251, 30252, 30253, 30255), Times (31801, 31802, 31804, 31803) und Symbol (31903) zu den Standard-Fonts, die auf den meisten PostScript-Ausgabegeräten installiert sind. Zumindest die Dickenwerte dieser Schriften stimmen in der Regel mit den von TUSTEP benutzten Tabellen überein; die Formen der einzelnen Zeichen können von Gerät zu Gerät variieren. Zusätzlich können auf den meisten PostScript-Ausgabegeräten Koptisch (31951) und Griechisch (31901) benutzt werden; bei der Schrift 31901 handelt es sich in Wirklichkeit um den Font »Symbol«, der um Akzente, Spiritus und iota subscriptum erweitert wurde; die Schrift 31951 ist z. Zt. eine Pixel-Schrift (Type-3-Font), die von TUSTEP in die PostScript-Datei eingebunden und damit automatisch auf den Drucker bzw. Belichter geladen wird.

Der Font 30001 (Hebräisch, M. Krupp) wird von TUSTEP automatisch in die PostScript-Datei mit eingebunden.

Bei den Schriften mit den Nummern 31701 bis 31785 handelt es sich um Fonts der Newton Font-Familie der Firma Paratype, Moskau, die für TUSTEP überarbeitet wurden und nur zusammen mit TUSTEP und zur Nutzung mit TUSTEP ausgeliefert werden dürfen. Dasselbe gilt für die Schrift 31920, einen phonetischen Zeichensatz aus der Newton-Familie. Wird einer dieser Fonts im Satzprogramm benutzt, so wird er von \*PSAUS automatisch in die PostScript-Datei mit eingebunden.

Bei der Font-Familie TusLibertine und TusBiolinum handelt es sich um eine Umwandlung von OTF nach Type1 (und vor allem bei Font Nummer 32512 um eine Überarbeitung) der Open-Type-Fonts Linux Libertine bzw. Biolinum, die als Open Source Fonts unter der »GNU General Public License + font exception« und der »SIL Open Font License«

zur Verfügung stehen. Diese Fonts werden nicht in die von #\*PSAUS erzeugten .ps-Dateien eingebunden; sie stehen (einschließlich der griechischen und kyrillischen Zeichen) unter den Namen TusLibertineO, TusLibertineOI, TusLibertineOB, TusLibertineOBI bzw. TusBiolinumO, TusBiolinumOI und TusBiolinumOB in den Dateien TusLibertine\_R.pfa, TusLibertine\_RI.pfa, TusLibertine\_RB.pfa, TusLibertine\_RBI.pfa bzw. TusBiolinum\_R.pfa, TusBiolinum\_RI.pfa und TusBiolinum\_RB.pfa im Verzeichnis tustep auf dem Träger TUSTEP\_LIB bereit. Damit sie für GhostScript bzw. für Acrobat Distiller zugänglich sind, muss die Konfiguration von Ghostview und der Inhalt der dort benutzten fontmap bzw. (bei Acrobat Distiller) der Inhalts von »Voreinstellungen« / »Schriftordner« entsprechend ergänzt werden.

Bei allen übrigen Schriften ist es vom Ausgabegerät (Drucker, Belichter) abhängig, ob sie vorhanden sind oder nicht. Identität des Schriftnamens ist nicht immer Gewähr für Identität der Schrift. Im Zweifelsfall empfiehlt es sich, sich durch einen Druck- bzw. Belichtungstest zu vergewissern, ob die erwarteten Schriften vorhanden sind.

Außer den hier aufgeführten Schriften können benutzereigene Schriften benutzt werden, die dem Satzprogramm über die Spezifikation SCHRIFTEN bekannt gemacht werden können. Vgl. dazu das Standard-Makro \*PSFONT.

**KAP**

Sonderbehandlung der Kapitälchenschrift. <fehlt>

Dieser Parameter wird angegeben, wenn alle zwischen ++ und { / ++ { bzw. #K+ und #K- / #?- stehenden Zeichen aus Schrift-Umschaltbereich 3 genommen werden sollen, obwohl dort eine Kapitälchenschrift steht. Fehlt dieser Parameter, so nimmt das Programm nur für die Kleinbuchstaben, Akzente und Ziffern die entsprechenden Zeichen aus der Kapitälchenschrift in Bereich 3; die übrigen Zeichen werden aus Bereich 1 genommen; dies ist der Normalfall und ermöglicht, z. B. bei Namen, die mit großen Anfangsbuchstaben in Kapitälchen gesetzt werden sollen, die Umschaltung auf Kapitälchen schon vor dem Anfangsbuchstaben anzugeben.

**VSS**

Umsetzung von Versal-ß. <->

- = - Versal-ß bzw. Kapitälchen-ß wird als SS bzw. ss ausgegeben (das gleiche gilt, wenn der Parameter VSS fehlt).
- = + Versal-ß bzw. Kapitälchen-ß wird als ß bzw. als ſ ausgegeben, falls der verwendete Font ein solches Zeichen enthält und in der Dicktentabelle ein von 0 verschiedener Wert dafür ausgewiesen ist; andernfalls wird das Versal-ß bzw. Kapitälchen-ß wie bisher als SS bzw. ss ausgegeben; zusätzlich wird eine entsprechende Fehlermeldung in die PROTOKOLL-Datei ausgegeben.



- HBU** Hebräische und arabische Textteile umdrehen. <1>  
 IFHBU = 1 Hebräische und arabische Textteile sind bereits umgedreht.  
 = 0 Hebräische und arabische Textteile werden automatisch umgedreht.
- SLW** Verändern der Schriftlaufweite. <fehlt>  
 Mit diesem Parameter kann die Laufweite der mit Parameter SCH angegebenen Schriften durch positives oder negatives Spationieren in Vielfachen von 1/1000 Geviert (»Bildlinien«) verändert werden.  
 +m fügt hinter jedem Zeichen einen m Bildlinien breiten Freiraum ein.  
 -m nimmt am rechten Rand jedes Zeichens m Bildlinien Freiraum weg (das Zeichen wird nicht beschnitten, ragt aber ggf. in das nachfolgende Zeichen hinein / über den rechten Rand hinaus).  
 Es können so viele Werte angegeben werden wie Schriftnummern mit dem Parameter SCH angegeben sind. Der n-te Zahlenwert gilt für die an n-ter Stelle angegebene Schrift.
- SGM** Schriftgrößen-Modifikation. <fehlt>  
 Mit diesem Parameter kann die Höhe und die Breite der mit dem Parameter SCH angegebenen Schriften relativ zu den Originalwerten modifiziert werden. Angaben absolut (z. B. 1100) oder relativ (z. B. +100) in Promille. Sollen Höhe und Breite unterschiedlich modifiziert werden, so sind die Angaben für Höhe und Breite durch »:« zu trennen. Es können so viele Werte(paare) angegeben werden wie Schriftnummern mit dem Parameter SCH angegeben sind. Die n-te Angabe gilt für die an n-ter Stelle angegebene Schrift. Soll eine Schrift nicht modifiziert werden, so kann statt 1000 auch 0 angegeben werden.

## Schriftgrade für die einzelnen Textteile

- GRO** Schriftgrad und Durchschuss für die einzelnen Textteile (alle Angaben in typographischen Punkt). [ XI ]  
 <9+2 8+1 8+1 10+2 12+2 9+2 8+2 8+2>
- NGRADT+NDURCHT Schriftgrad+Durchschuss für Grundtext und Titelzeilen der Stufe 1 (»&«) <9+2>
- NGRADE+NDURCHE Schriftgrad+Durchschuss für Einschaltungen (»\$\$\$«) <8+1>
- NGRADF+NDURCHF Schriftgrad+Durchschuss für Fußnoten  
 <8+1>  
 (bei FUSNOTEN=- und MODUS ungleich F:  
 <0+0> )

---

NGRADU2+NDURCHU2	Schriftgrad+Durchschuss für Titelzeilen der Stufe 2 ( $\gg&&\ll$ ) $\langle 10+2 \rangle$
NGRADU3+NDURCHU3	Schriftgrad+Durchschuss für Titelzeilen der Stufe 3 ( $\gg&&\ll$ ) $\langle 12+2 \rangle$ (für Stufe 1.n siehe unter Parameter $\&n\&$ )
NGRAD+NDURCHS	<p>Schriftgrad+Durchschuss für die Seitennummer <math>\langle \text{NGRADT}+\text{NDURCHT} \rangle</math></p> <p>Die Summe aus Schriftgrad und Durchschuss ergibt den Abstand zwischen Text und Seitennummer.</p> <p>Ein negativer Wert für den Durchschuss der Seitennummer gibt an, um wieviel die (oben stehende) Seitennummer nach unten verschoben werden soll. Sie wird um den angegebenen Wert unterhalb der Stelle ausgegeben, an der sie bei positiver Angabe stünde. Für die Berechnung des Abstandes zur Satzspiegel-Oberkante wird dann – falls angegeben – die Durchschuss- bzw. Abstandsangabe für den lebenden Kolumnentitel herangezogen. Ist dafür keine Angabe gemacht, wird der entsprechende Wert für den Grundtext benutzt.</p> <p>Negativer Durchschuss, der größer ist als der Schriftgrad, kann angegeben werden, um die (unten stehende) Seitennummer oberhalb der Satzspiegelunterkante zu setzen. Für die Ausgabe der Seitennummer wird in diesem Fall vertikal auf die Stelle positioniert, die um den negativ angegebenen Durchschuss oberhalb der Satzspiegelunterkante liegt, und dann ein Vorschub um den für die Seitennummer angegebenen Schriftgrad ausgeführt.</p>
NGRADK+NDURCHK	<p>Schriftgrad+Durchschuss für lebende Kolumnentitel oben <math>\langle 8+\text{NDURCHS} \rangle</math></p> <p>Der Durchschuss wird, falls die Seitennummer unten steht, für die Berechnung des Abstandes (= Summe aus Schriftgrad und Durchschuss) zwischen Kolumnentitel und Text berücksichtigt.</p> <p>Sollen innerhalb von Kolumnentiteln mit Hilfe von Steueranweisungen Schriftgrößenwechsel oder Positionierungen vorgenommen werden, so muss als Größe für den Ko-</p>

lumentitel die Grundschrift-Größe angegeben werden. Alle Schriftgrößenwechsel sind dann im Kolummentitel selbst anzugeben.

NGRADKU+NDURCHKU Schriftgrad+Durchschuss für lebende Kolummentitel unten <8+NDURCHS>

Untere Kolummentitel stehen in der selben Zeile wie die unten stehende Seitennummer.

### Verfügbare Schriftgrade

TUSTEP arbeitet mit dem auf dem europäischen Kontinent üblichen Didot-Maßsystem mit 1 Punkt = 0,375 mm; es unterstützt die Schriftgrade von 4 bis 144 Punkt. Andere Schriftgrade können durch entsprechende Maßstabsangaben bei der Ausgabe mit dem Standard-Makro \*PSAUS (Spezifikation FAKTOR; siehe dort) erreicht werden.

Die angegebenen Schriftgrade können in der Breite modifiziert werden. In diesem Fall besteht die Größenangabe aus zwei durch »:« getrennten Zahlen, von denen die erste die Kegelhöhe, die zweite die veränderte Geviertbreite angibt (z. B. 10:9+2 für eine 10-Punkt-Schrift, modifiziert auf 9-Punkt-Breite, mit zusätzlich 2 Punkt Durchschuss).

Die Schriftgrade zwischen 4 und 24 Punkt können in Viertelpunkt-Schritten angegeben werden. Diese Schriftgrade können außerdem in der Breite in den gleichen Abstufungen modifiziert werden. Die Angaben erfolgen als Dezimalbruch (z. B. 9,5:9,25+2).

Die Zeilenabstände können derzeit nur ganzzahlige Vielfache von Punktschritten sein; die Durchschussangabe erfolgt derzeit dennoch ganzzahlig; für die Berechnung des Zeilenabstands werden derzeit nach der Addition die Stellen hinter dem Komma ignoriert.

Hinter dem Schriftgrad kann anstelle des Durchschusses auch der gewünschte Zeilenabstand angegeben werden. Statt durch »+« ist der Zeilenabstand durch »//« vom Schriftgrad zu trennen. Die Angaben 9+2 und 9//11 sind also gleichwertig.

Negativer Durchschuss muss durch ein »-« hinter dem »+« angegeben werden. Die Angaben 9//8 und 9+-1 sind also gleichwertig.

## Satzspiegel

**BRE** Breite des Satzspiegels (in Punkt) für die einzelnen Textteile. [ I ]

<288 288 288 288 288 288 288 288 288>

NBRT Satzbreite für Grundtext <288>

NBRE Satzbreite für Einschaltungen (« \$ \$«) <NBRT>

NBRF Satzbreite für Fußnoten <NBRT>

NBRU2	Satzbreite für Titelzeilen der Stufe 2 (« &&«) <NBRT>
NBRU3	Satzbreite für Titelzeilen der Stufe 3 (« &&&«) <NBRT> (für Titelzeilen der Stufe 1.n siehe unter Parameter &n&)
NBRSZ	Satzbreite für Seitennummer <NBRT> Bei mehrspaltigem Satz ist, wenn nicht die Spalten, sondern die Seiten gezählt werden sollen, die Breite der ganzen Seite anzugeben.
NBRKT	Satzbreite für lebenden Kolumnentitel oben <NBRT> (vgl. die Anmerkung zum Parameter KOL)
NBRST	Satzbreite für Strich unter Kolumnentitel <NBRT>
NBRKU	Satzbreite für lebenden Kolumnentitel unten <NBRT> (vgl. die Anmerkung zum Parameter KOL)

Die Mindest-Satzbreite ist 50 Punkt; sie kann durch Angabe von »!« unmittelbar hinter der Zahl unterschritten werden. Die maximale Satzbreite ist 2400 Punkt.

**HOE**

Höhe des Satzspiegels. [ I ] <438 2 0 0 1 0 30>

NHOCH Spaltenhöhe (ausschließlich Kolumnentitel und Seitennummer) in Punkt. <438>

Die Mindest-Spaltenhöhe ist 48 Punkt = 18 mm; die maximale Spaltenhöhe (einschließlich Seitennummer und Kolumnentitel) ist 3600 Punkt = 135 cm.

Ist zur Spezifikation AUSGABE keine Datei angegeben, so kann eine größere Spaltenhöhe verlangt werden; diese muss dann negativ angegeben werden, um zu signalisieren, dass dies Absicht und kein Schreibfehler ist. In diesem Fall wird außerdem der Raum, der von Fußnoten eingenommen wird, bei der Steueranweisung \$\$\$-n\$\$\$ (siehe unten Kapitel 7.1) unberücksichtigt gelassen.

NAUSGL Angaben zum Spaltenhöhenausgleich <2>

<0 = nur Austreiben zu kurzer Spalten auf Spaltenhöhe, kein Zusammenschieben zu lang gewordener Spalten; Fußnotenzeilen, die nur um Bruchteile der Zeilenhöhe nach unten über die Spaltengrenze hinausragen würden, auf die nächste Spalte überlaufen lassen.

0 = nur Zusammenschieben zu lang gewordener Spalten, Austreiben auf volle Spaltenhöhe nur an den durch &!A gekennzeichneten Stellen.

>0 = zu lange Spalten auf Spaltenhöhe zusammenschieben; zu kurze Spalten austreiben, wenn weniger als NAUSGL Zeilen bis zur vollen Spaltenhöhe fehlen.

Wenn zum Spaltenhöhenausgleich der Durchschuss zwischen den Zeilen nicht verändert werden darf, ist auf den Betrag von NAUSGL der Wert 100 aufzuaddieren. Es werden dann nur die als variabel bezeichneten Vorschübe zum Spaltenhöhenausgleich herangezogen.

Wird auf den Betrag von NAUSGL der Wert 200 statt 100 aufaddiert, so werden auch die »variablen« Vorschübe beim Ausgleich der Spaltenhöhe nicht verändert. Zum Spaltenhöhenausgleich stehen dann nur der Freiraum zwischen Text und Fußnoten (falls vorhanden) und die mit &!A markierten Stellen zur Verfügung. Der Satzspiegel wird für solche Seiten so vergrößert, dass die vorletzte Zeile auf der Satzspiegelunterkante steht. Wird 300 statt 200 aufaddiert, so unterbleibt die Vergrößerung des Satzspiegels; die letzte Zeile ragt dann u. U. um Bruchteile der Zeilenhöhe über den Satzspiegel hinaus.

-900 = Spaltenhöhe nicht auffüllen; wenn die Spalte zu lang geworden ist (z. B. wegen eines eingebrachten Hurenkinds), wird der Abstand zur unten stehenden Seitennummer um den Betrag verringert, um den die Spalte zu lang geworden ist, maximal jedoch um NGRADS+NDURCHS (vgl. Parameter GRO).

-999 = Spaltenhöhe nicht auffüllen; Seitennummern, die unten auf der Seite stehen, werden im angegebenen Abstand (siehe Parameter GRO ) unmittelbar hinter der letzten Textzeile gesetzt.

nn; 2 = gleichbedeutend mit nn (für nn stehen die beschriebenen Werte): Falls die variablen Vorschübe für den Spaltenhöhenausgleich nicht ausreichen, wird der Durchschuss zwischen den Textzeilen in Vielfachen von 1/2 Punkt verändert.

nn; 8 = (für nn stehen die beschriebenen Werte): Falls die variablen Vorschübe für den Spaltenhöhenausgleich nicht ausreichen, wird der Durchschuss zwischen den Textzeilen in Vielfachen von 1/8 Punkt um max. 1 Punkt verändert. Reicht das nicht aus, so kann hinter einem weiteren Strichpunkt angegeben werden, um wieviele Achtelpunkt der Durchschuss maximal verändert werden darf.

NSCHUS 0 = Schusterjungen nicht dulden. <0>  
1 = Schusterjungen dulden.

NHUR 0 = Hurenkinder vermeiden (werden in die vorhergehende Spalte eingebracht). <0>

- 1 = Hurenkinder zulassen, wenn sie einen Fußnotenverweis enthalten.  
Hurenkinder bleiben in jedem Fall (auch ohne Angabe von 1 für NHUR) oben stehen, wenn sie einen Fußnotenverweis enthalten und die Fußnoten spaltenweise nummeriert werden (d. h. wenn der erste Wert des Parameters FNN 1, -1 oder \* ist).
- 2 = Hurenkinder in jedem Fall zulassen.
- MINTZL Mindestzahl von Textzeilen, die auch bei übergelaufenen Fußnoten oben auf der Seite stehen sollen <1>
- IFSUNT Angabe zur Behandlung von Fußnotenzeilen, die über den Satzspiegelrand hinausragen würden <0>
- 0 = Bei NAUSGL = 0 und bei positivem NAUSGL werden Fußnotenzeilen, die um weniger als 1/2 (Fußnoten-) Zeilenabstand über den Satzspiegel hinausragen würden, noch auf der aktuellen Seite untergebracht. Bei NAUSGL < 0 werden Fußnotenzeilen, die über den unteren Satzspiegelrand hinausragen würden, auf die nächste Seite übertragen.
- Wenn es sich um die erste Fußnotenzeile einer Seite handelt, führt dies in der Regel dazu, dass nach der Zeile mit dem Fußnotenaufruf noch eine weitere Textzeile auf der Seite untergebracht wird, obwohl die erste Fußnotenzeile (wegen des Abstandes zwischen Text und Fußnote) nicht mehr auf dieser Seite begonnen wird.
- n = Fußnotenzeilen, die um weniger als  $n/2$  (Fußnoten-) Zeilenabstände über den unteren Satzspiegelrand hinausragen würden, werden noch auf dieser Seite untergebracht.
- n kann statt in Vielfachen von halben Zeilenabständen auch in Punkt angegeben werden, indem ein ».« hinter die Zahl geschrieben wird. Ist kein Punkt hinter n angegeben, so darf n eine maximal zweistellige Zahl sein.
- n = Wie n, jedoch nur für die erste Zeile einer (in der letzten Textzeile aufgerufenen) Fußnote; für Folgezeilen wird so verfahren wie bei IFSUNT = 0. Damit kann der Forderung, die zuletzt aufgerufene Fußnote noch auf der Seite beginnen zu lassen, gezielter nachgekommen werden.
- n; n2 = Wie -n; der hinter dem ».« angegebene Wert gilt für die letzte Zeile einer Fußnote.

(Die bis einschl. Version 2004 geltenden Regeln entsprechen angenähert der Angabe »-4«).

IAPPU	Falls Apparate um mehr als IAPPU Punkt unten über den Satzspiegel hinausragen, soll auf der Folgeseite entsprechend viel Platz freigehalten werden. <30>
-------	--

## Seiten- und Spaltenmontage

<b>SEI</b>	Seitennummerierung. [I] <1 1 1 0 0 0 0 1 0>
NRSEIT	erste Seitennummer <1> Negative Angabe: Seitennummer um jeweils 2 weiter-schalten.  Ist der Parameter SEI nicht angegeben, so wird der ent-sprechende Wert aus dem Parameter MON eingesetzt.
NRPOS	Position der Seitennummer <1> -99 = keine (Zeile mit einer) Seitennummer ausgeben 0 = unten außen,                    2 = oben außen, 1 = unten Mitte,                    3 = oben Mitte, 7 = unten links,                    4 = oben innen, 8 = unten rechts,                    5 = oben links, 9 = unten innen,                    6 = oben rechts, 10, 11, . . . , 19: Wie 0, 1, . . . , 9, aber Seitennummer in römischen Ziffern (Großbuchstaben) 20, 21, . . . , 29: Wie 10, 11, . . . , 19, aber Kleinbuchsta-ben.  Bei den Positionen »außen« und »innen« kann auf die ent-sprechenden Angaben 100 aufaddiert werden, um anzu-geben, dass trotz spaltenweiser Nummerierung die Posi-tionierung seitenweise vorgenommen werden soll.  Negative Angabe: Die Ausgabe der Seitennummer wird unterdrückt (unsichtbar ausgegeben) bis sie durch die Steueranweisung &!N+ bzw. &!N. explizit eingeschaltet wird.
NRSUB	Schrift-Umschaltbereich, der für die Seitennummer gelten soll <1> 0 = Seitennummer wird unsichtbar ausgegeben; die Zei-le, in der diese (unsichtbare) Seitennummer steht, wird ausgegeben (kann statt NRPOS = -1 notwendig sein, wenn untere Kolumnentitel gesetzt werden sol-len, obwohl keine Seitennummer ausgegeben wer-den soll).
NRKUR	0 = Seitennummer soll gerade gesetzt werden. <0>

	1 = Seitennummer soll schräg gesetzt werden (muss auch angegeben werden, wenn in <code>NRSUB</code> eine Kursivschrift steht).
NRZUS	0 = kein Zusatz zur Seitennummer <0> 1 = Stern hinter der Seitennummer ergänzen 2 = Seitennummer zwischen Geviertstriche setzen 3 = Seitennummer in runde Klammern setzen 4 = Seitennummer in eckige Klammern setzen 5 = Wie 1 und 2 gleichzeitig 6 = Wie 1 und 3 gleichzeitig 7 = Wie 1 und 4 gleichzeitig 8 = Seitennummer zwischen Halbgeviertstriche setzen 9 = Wie 1 und 8 gleichzeitig (Angaben 2–9 sind nur bei <code>NRPOS</code> = 1 oder 3 erlaubt)
NREINZ	Einzug (in Punkt) der Seitennummer vom linken bzw. rechten Rand. Ein negativer Wert gibt den Abstand der ersten (linke Seiten) bzw. letzten (rechte Seiten) Ziffer vom Satzspiegelrand an. Wird auf diesen Wert -1000 aufaddiert, so gibt er (modulo 1000) den Abstand der letzten (linke Seiten) bzw. ersten (rechte Seiten) Ziffer vom Satzspiegelrand an. Nicht erlaubt bei <code>NRPOS</code> = 1 und 3 <0>
NZAB	Zähler und
NNAB	Nenner (Geviert-Bruchteile) für vergrößerten Abstand zwischen Klammern bzw. Strichen und Seitennummer <0 1>
NSFRB	Farbe und alternativer Schriftgrad für die Seitenzahl. <0>  Sollen Grauwerte zwischen schwarz (0) und weiß (100) angegeben werden, so ist nur ein Zahlenwert anzugeben. Soll die Seitenzahl farbig gesetzt werden, so sind insgesamt 4 Zahlenwerte für die Druckfarben Cyan, Magenta, Yellow, Black (in dieser Reihenfolge) im CMYK-System anzugeben. Jede Zahl steht für den (ganzzahligen) Prozentwert des Anteils jeder dieser vier Druckfarben an der gewünschten Farbe.  Kommen in einem Satzlauf sowohl arabische als auch römische Seitenzahlen vor, so kann hinter den Angaben für die Farbe, durch »:« davon getrennt, angegeben werden, in welchem Schriftgrad die römischen Seitennummern zu setzen sind. Die Angabe <code>NGRADS</code> zum Parameter <code>GRO</code> gilt dann nur für die arabischen Seitenzahlen; zu <code>NRPOS</code> muss eine einstellige Zahl (für »Seitenzahlen in arabischen Ziffern«) angegeben sein.

Bei mehrspaltigem Satz wird statt der Seitennummer die Spaltennummer über bzw. unter den einzelnen Spalten ausgegeben, falls die Breite `NBRSZ` für die Seitennummer im Parameter `BRE` nicht mindestens



um die Hälfte größer ist als die Breite NBRT für den Grundtext. In diesem Fall gilt das für »Seiten« Gesagte für die einzelnen Spalten; der »linken Seite« entspricht: »Spalte mit gerader Spaltennummer«, der »rechten Seite«: »Spalte mit ungerader Spaltennummer«. Auch die Positionsangaben »links«, »mitte«, »rechts«, »innen«, »außen« beziehen sich dann auf die einzelnen Spalten.

**SPA** Spaltenzahl. [ I ] <1 1 0>

Das Programm setzt einspaltig; die Spalten werden erst bei der Ausgabe mit dem Standard-Makro \*PSAUS zu Seiten zusammenmontiert. Zur Berechnung und zur Ausgabe der Seitennummern und der lebenden Kolummentitel wird die folgende Angabe benötigt:

NSPALT Zahl der Spalten (1 bis 6) <1>

Ist der Parameter SPA nicht angegeben, so wird der entsprechende Wert aus dem Parameter MON eingesetzt.

NSPALTf Zahl der Spalten für die Fußnoten (1 oder 2) <1>

Sollen bei zweispaltigem Satz alle Fußnoten einer Seite unter der jeweils rechten Spalte gesetzt werden, so ist zu NSPALTf der Wert -1 anzugeben.

Sollen die Fußnoten zweispaltig unter einspaltigem Text gesetzt werden, so muss beim Satz der Fußnoten die Breite der einzelnen Spalten angegeben werden.

NEINRF Angabe (in Punkt), wie weit die zweite Spalte der Fußnoten vom linken Rand eingerückt werden soll.

Anmerkung: Mischung von ein- und mehrspaltigem Satz auf einer Seite ist über das Standard-Makro \*SUMBRUCH möglich.

**MON** Seitenmontage. [ I ] <1 0 288 12 12>

Dieser Parameter muss in jedem Fall angegeben werden, wenn einer der Parameter BRE, SEI, SPA, ZLN, MAL, MAR angegeben wird. Die Angaben werden für die Weiterverarbeitung nach dem Satz benötigt.

NSPALT Zahl der Spalten (maximal 6), die zu einer Seite zusammenmontiert werden sollen. <1>

Soll hier ein anderer Wert angegeben werden als mit dem Parameter SPA, so muss NSPALT negativ angegeben werden.

Soll beim Zusammenmontieren der Spalten zu Seiten nicht links, sondern rechts begonnen werden (z. B. bei hebräischen Texten, bei denen die erste Spalte einer Seite rechts neben der zweiten Spalte stehen muss), so muss auf die Spaltenzahl der Wert 10 aufaddiert werden.

N1SPAL	<p>Nummer der ersten Spalte (muss bei mehrspaltigem Satz mit der Nummer der ersten Seite übereinstimmen). &lt;0&gt;</p> <p>Soll die mit dem Parameter SEI angegebene erste Seitennummer NRSEIT oder die Voreinstellung übernommen werden, so kann für N1SPAL der Wert 0 angegeben werden.</p> <p>Soll hier ein anderer Wert angegeben werden als mit dem Parameter SEI, so muss N1SPAL negativ angegeben werden.</p>
NBREIT	Breite der (gleichbreiten) Spalten ohne Marginalien, in Punkt <288>
NFREIO	Freiraum vor der ersten Spalte in Punkt (für den Abstand zum linken Materialrand, für Marginalien und Zeilenzähler links). Der Mindestabstand zum Materialrand beträgt 6 Punkt. <12>
NFREIn	Freiraum nach der n-ten Spalte in Punkt (für den Abstand zwischen den Spalten, für Marginalien und Zeilenzähler rechts, für den Abstand zum rechten Materialrand). Der Mindestabstand zum Materialrand beträgt 6 Punkt. Hier werden genau NSPALT Zahlen erwartet. <12>

## Zeilenumbruch und Silbentrennung

Die kleinste Einheit, die in den beiden folgenden Parametern benutzt wird, ist die »Bildlinie«. Diese Bezeichnung stammt noch vom Digiset 50T1, einem Kathodenstrahlbelichter, für den das TUSTEP-Satzprogramm Ende der 60er Jahre zunächst entwickelt worden war. Eine Bildlinie betrug dort 1/50 einer Geviertbreite (bei einer 9-Punkt-Schrift also 9/50 Punkt). Seit 1992 arbeitet das Satzprogramm mit PostScript-Fonts; dort wird die Zeichen-Dicke in Vielfachen von 1/1000 Geviert angegeben; eine »Bildlinie« entspricht damit 1/1000 Geviert.

<b>AUS</b>	Austreiben der Zeilen. [ XI ] <0 0 0 1/10 1/3 1 0 0>
IFLATT	<p>Angaben zum Austreiben der Zeilen &lt;0&gt;</p> <p>0 = Blocksatz (Zeilen austreiben)</p> <p>&lt;0 = Fehlen in einer Zeile zur vollen Satzbreite noch mehr als  IFLATT  Bildlinien, so wird die Zeile nicht ausgetrieben.</p> <p>&gt;0 = Müssten in einer Zeile die Spatien beim Austreiben um mehr als IFLATT Bildlinien vergrößert werden, so wird die Zeile nicht ausgetrieben.</p>
IEINZR	0 = kein rechter Einzug in Ausgangszeilen (= letzte Zeile eines Abschnitts) <0>

	1 = rechter Einzug von mind. 10 Punkt in Ausgangszeilen
	n = rechter Einzug von mind. n Punkt in Ausgangszeilen
	-n = Ausgangszeilen, bei denen weniger als n Punkt zur vollen Satzbreite fehlen, sollen ausgetrieben werden.
IZENTE	Mindest-Einzug für Zeilen, die auf Mitte zentriert werden sollen; wenn nicht 0, wird in dem zu zentrierenden Bereich keine Silbentrennung durchgeführt. <0>
ISPMIN	Toleranzgrenze für Spatien-Verringerung ohne Fehlerkommentar. <1/10>
	Der Betrag, um den eine Verringerung ohne Fehlerkommentar maximal vorgenommen werden soll, kann in Bildlinien oder in Geviert-Bruchteilen angegeben werden.
	Wird »-1« angegeben, so wird ein Kommentar mit der Spatienbreite auch dann ins Protokoll ausgegeben, wenn sie nicht vom Soll abweicht, aber wenigstens 1 Spatium in der Zeile ist und die Zeile weder auf Mitte zentriert noch nach rechts geschoben ist.
ISPPL	Toleranzgrenze für Spatien-Vergrößerung ohne Fehlerkommentar. <1/3>
	Der Betrag, um den eine Vergrößerung ohne Fehlerkommentar maximal vorgenommen werden soll, kann in Bildlinien oder in Geviert-Bruchteilen angegeben werden.
ISPUNT	Untergrenze für Spatienbreite <1>
	Zu lange Zeilen werden nur so weit zusammengeschoben, dass die in Bildlinien angegebene Spatienbreite nicht unterschritten wird.
IRNDAU	Angabe, um wieviele Bildlinien die Zeichen Divis, Punkt, Komma und Apostroph am rechten Rand von ausgetriebenen oder rechts zentrierten Zeilen über den Satzspiegelrand hinausragen sollen, um einen optischen Randausgleich zu erreichen. Sollen Divis, Punkt, Komma und Apostroph ganz über den Satzspiegelrand hinausragen, so ist 999 anzugeben. <0>
LAUSGZ	Mindestlänge (in % der augenblicklich geltenden Zeilenlänge), die eine Ausgangszeile von Abschnitten haben muss. Wird diese Länge unterschritten, so wird ins Protokoll die Fehlermeldung »Kurze Ausgangszeile« ausgegeben. <0>
<b>SIL</b>	Silbentrennung. [ XI ] <0 1/6 1/12 5 2 2 4>
ITRENN	0 = Silbentrennung wird durchgeführt <0>

- 1 = auch fester Ausschluss wird (ohne Divis) abtrennbar  
 2 = Silbentrennungen der QUELL-Datei bleiben unberücksichtigt.

Falls ein Wort getrennt werden muss, das in der QUELL-Datei getrennt ist, wird die Trennung der QUELL-Datei nach Möglichkeit beibehalten, es sei denn, dass dies durch die Angabe 2 zu ITRENN verhindert wird. Silbentrennungen aus der QUELL-Datei ignoriert das Programm für neue Trennversuche jedoch – trotz Angabe von 2 zu ITRENN – nur im »1. Durchlauf« (vgl. Parameter LAU); in allen anderen Fällen werden sie bei neuen Trennversuchen berücksichtigt.

- 3 = Wie 1 + 2  
 10, 11, 12, 13 = Wie 0, 1, 2, 3, jedoch darf »-« auch nach Ziffer oder nach \ als Trennstelle benutzt werden.  
 -1 = keine Silbentrennung außer an den durch »\« bezeichneten Stellen; zusammengesetzte Wörter, die Bindestrich enthalten, dürfen nach diesem getrennt werden.

ISPPOS Silbentrennung nur durchführen, wenn durch das Austreiben der Zeile die Spatien um mehr als ISPPOS Bildlinien vergrößert würden. <1/6>

Die Angabe kann in Bildlinien oder in Geviert-Bruchteilen erfolgen. Sie wird auch zur Bestimmung der frühestmöglichen Trennstelle eines zu trennenden Wortes herangezogen.

ISPNEG Silbentrennung nur durchführen, wenn durch das Aufnehmen des ganzen Wortes in die Zeile die Spatien um mindestens den angegebenen Wert verringert würden. <1/12>

Die Angabe kann in Bildlinien oder in Geviert-Bruchteilen erfolgen. Sie wird auch zur Bestimmung der spätestmöglichen Trennstelle eines zu trennenden Wortes herangezogen.

ITRENNW Nur Wörter mit mindestens ITRENNW Buchstaben trennen <5>

ITRNV Mindestzahl von Buchstaben, die bei Silbentrennung vor dem Divis stehen müssen <2>

ITRNH Mindestzahl von Buchstaben, die bei Silbentrennung hinter dem Divis stehen müssen <2>

**NTRNS** Zahl von aufeinander folgenden Zeilen einer Spalte, an deren Ende ein Wort getrennt werden darf. Wird dieser Wert überschritten, so wird ein Fehlerkommentar in die PROTOKOLL-Datei ausgegeben; ist NTRNS negativ angegeben, so unterbleibt die Silbentrennung in der betreffenden Zeile. <4>

**RSR** Rechtschreibreform von 1996: Trennregeln. [I] <1>

**NEUREG** Angabe, ob bei der Silbentrennung die Regeln von 1901/1902 oder die von 1996 angewandt werden sollen (betrifft insbesondere die Trennung von ck und st):

0 = Silbentrennung nach den alten Regeln von 1901/1902; ck wird als  $\backslash\backslash^{\wedge}k-k$  getrennt; in den Eingabedaten vorhandene Silbentrennung k-k und  $\backslash\backslash^{\wedge}k-k$  wird, wenn ein Vokal vor k-k steht, immer zu ck, es sei denn, an der selben Stelle würde wieder getrennt.

-1 = Wie 0, jedoch wird in den Eingabedaten vorhandene Silbentrennung k-k nur dann zu ck, wenn sie mit  $\backslash\backslash^{\wedge}$  vor dem »k-« markiert ist.

Hinweis: Wenn ein Text zunächst nach alten Regeln gesetzt wurde und die ZIEL-Datei als QUELL-Datei eines weiteren Satzlaufes mit neuen Regeln genutzt werden soll, so müssen erst die Trennungen k-k in ck (soweit markiert oder – bei älteren Dateien – ohne Markierung zutreffend) zurückverwandelt werden und die Markierungen (sie bedeuten Trennverbot) entfernt werden.

1 = Silbentrennung nach den neuen Regeln von 1996

## Marginalien

**ZLN** Zeilennummerierung. [I] <0 5 6 1 0 0 0>

**NRSW** Schrittweite der Zeilennummer <0>  
 0 = keine Zeilennummer  
 2 = Zeilennummerierung im 2er-Schritt  
 5 = Zeilennummerierung im 5er-Schritt  
 Andere Schrittweiten sind möglich.

**NRABST** Abstand der Zeilennummer vom Satzspiegelrand in Punkt. <5>

Negative Angabe: am rechten Satzspiegelrand einstellige Zeilennummern linksbündig (d. h. über der Zehnerstelle von zweistelligen Zeilennummern) setzen.

**NRGRO** Schriftgröße der Zeilennummer in Punkt <6>

NRSUB	Schrift-Umschaltbereich, aus dem die Zeilennummer gesetzt werden soll <1>
NRKUR	0 = Zeilennummer soll gerade gesetzt werden (muss auch angegeben werden, wenn in NRSUB eine Kursivschrift steht) <0> 1 = Zeilennummer soll schräg gesetzt werden
IZNLKS	Angabe, wo die Zeilennummer stehen soll: <0> 0 = am rechten Rand, 10 = am linken Rand, 1 = innen (ungerade Spalten: links, gerade: rechts) 2 = außen (gerade Spalten: links, ungerade: rechts) 11 = innen (ungerade Seiten: links, gerade: rechts) 12 = außen (gerade Seiten: links, ungerade: rechts)  Positionierung der Zeilennummern bei mehrspaltigem Satz: Links ausgegebene Zeilennummern stehen am linken Seitenrand, rechts ausgegebene Zeilennummern stehen am rechten Rand der jeweiligen Spalte.
IZNPR	Ausgabe von Meldung über Zeilennummer ins Protokoll: <0>  0 = (gleichwertig mit »keine Angabe«): keine Meldung erzeugen. 1 = Meldungen über die am Satzspiegelrand ausgegebenen Zeilennummern erzeugen. 2 = Meldungen über Zeilennummern erzeugen, auch wenn die Zeilennummern nicht am Satzspiegelrand ausgegeben werden. 3 = wie 2, jedoch auch für die Zeilen, für die die Ausgabe der Zeilennummern durch &!r- bzw. &!r. unterdrückt wird. 4 = Eine Kommentarzeile mit der Seitennummer in der Form <!-- pn8888 --> vor der jeweils ersten Zeile einer Seite und mit der Zeilennummer in der Form <!-- ln8888 --> hinter jeder Zeile in die Ziel-Datei ausgeben.

Ist zu IZNPR einer der Werte 1–3 angegeben, und ist die Zeilennummer eine mit &!R(nn) erzeugte Nummer, so wird in der Zieldatei hinter dem Text der betreffenden Zeilen eine eigene Fortsetzungszeile mit der Unterscheidungsnummer 990 und dem Text <!-- ln8888 --> ausgegeben, wobei für 8888 die aktuelle Nummer eingesetzt wird. Für IZNPR=4 wird kein Kommentar im Protokoll erzeugt. Die Zeilennummer ist die mit &!R(nn) erzeugte Nummer oder die laufende Nummer der Zeile auf der Seite = die Zeilennummer aus der Satznummer der Zieldatei.

Sind in der Quelldatei solche Kommentarzeilen enthalten, so werden sie bei der Eingabe übergangen.

**MAL**

Marginalien am linken Rand. [ I ] <>

NGRADM Schriftgrad für Marginalien.

Soll innerhalb von Marginalien Schriftgrößenwechsel vorgenommen werden, so muss als Größe für die Marginalien die Grundschrift-Größe angegeben werden. Alle Schriftgrößenwechsel sind dann in der Marginalie selbst anzugeben.

NABSTM Abstand des Marginalien-Feldes vom Satzspiegelrand; negative Angabe möglich für Marginalien, die in den Satzspiegel hineinragen.

NBRM Breite des Marginalien-Feldes

NPOSM Zentrierung der Marginalien im Marginalien-Feld:

0 = auf Mitte zentriert

1 = links zentriert

2 = rechts zentriert

Positionierung der Marginalien bei mehrspaltigem Satz: Alle linken Marginalien (auch die durch @I und @A entstandenen) stehen am linken Seitenrand, alle rechten Marginalien (auch die durch @I und @A entstandenen) stehen am rechten Rand der jeweiligen Spalte.

MSUB Schrift-Umschaltbereich, der für die Marginalien gelten soll (vgl. Parameter SCH)

MKUR 0 = Marginalien sollen gerade gesetzt werden.

1 = Marginalien sollen schräg gesetzt werden (muss auch angegeben werden, wenn in MSUB eine Kursivschrift steht).

**MAR**

Marginalien am rechten Rand. [ I ] <>

(Angaben wie bei Parameter MAL)

**MLS**

Marginalien innerhalb des Satzspiegels auf linken Seiten (bei 1-spaltigem Satz) bzw. in Spalten mit gerader Nummer. [ I ] <>

(Angaben wie bei Parameter MAL)

**MRS**

Marginalien innerhalb des Satzspiegels auf rechten Seiten (bei 1-spaltigem Satz) bzw. in Spalten mit ungerader Nummer. [ I ] <>

(Angaben wie bei Parameter MAL)

## Lebende Kolummentitel

Lebende Kolummentitel können seiten- oder spaltenbezogen gesetzt werden. Für die entsprechenden Angaben im Parameter `KOL` gilt:

- »linke Seiten« = Seiten mit gerader Nummer,
- »rechte Seiten« = Seiten mit ungerader Nummer;
- »innen« = auf linken Seiten rechts, auf rechten Seiten links,
- »außen« = auf linken Seiten links, auf rechten Seiten rechts.

Bei mehrspaltigem Satz werden die lebenden Kolummentitel über die einzelnen Spalten gesetzt, falls die Breite `NBRKT` (siehe Parameter `BRE`) für den lebenden Kolummentitel nicht mindestens um die Hälfte größer ist als die Breite `NBRT` für den Grundtext.

In diesem Fall gilt das unten für »Seiten« Gesagte für die einzelnen Spalten; der »linken Seite« entspricht: »Spalte mit gerader Spaltennummer«, der »rechten Seite«: »Spalte mit ungerader Spaltennummer«.

Bei jedem Wechsel des Kolummentitels im Text ist in der `PROTOKOLL`-Datei ein Hinweis in der Form eines Fehlerkommentars vorgesehen. Dieser Hinweis kann durch negative Angabe von `KOLTL` beim folgenden Parameter unterdrückt werden.

<b>KOL</b>	Lebende Kolummentitel. [ I ] <0 0 0 0 0>
<b>KOLTL</b>	Lage und Aufteilung der lebenden Kolummentitel oben auf der Seite <0>
	0 = kein lebender Kolummentitel
	1, 2, . . . , 9 = Kolummentitel auf Mitte zentrieren, wobei
	1 = Kolummentitel identisch auf linken und rechten Seiten; der zuletzt vor dem Seitenwechsel angegebene Kolummentitel wird eingesetzt.
	2 = Kolummentitel geteilt (links und rechts verschieden); der zuletzt vor dem Seitenwechsel angegebene Kolummentitel wird eingesetzt.
	3 = Kolummentitel geteilt; auf linken Seiten wird der zuletzt vor dem Seitenwechsel angegebene, auf rechten Seiten der auf der jeweiligen Seite als letzter angegebene Kolummentitel eingesetzt.
	4 = Kolummentitel geteilt; auf rechten Seiten wird der zuletzt vor dem Seitenwechsel angegebene, auf linken Seiten der auf der jeweiligen Seite als letzter angegebene Kolummentitel eingesetzt.
	5 = Kolummentitel geteilt; der auf der jeweiligen Seite als letzter angegebene Kolummentitel wird eingesetzt.
	6 = Kolummentitel identisch auf linken und rechten Seiten; der als erster nach dem Seitenwechsel angegebene Kolummentitel wird eingesetzt.



- 7 = Kolummentitel geteilt (links und rechts verschieden); der als erster nach dem Seitenwechsel angegebene Kolummentitel wird eingesetzt.
- 8 = Kolummentitel geteilt; auf linken Seiten wird der als erster nach dem Seitenwechsel angegebene, auf rechten Seiten der als letzter auf der jeweiligen Seite angegebene Kolummentitel eingesetzt.
- 9 = Kolummentitel geteilt; auf rechten Seiten wird der als erster nach dem Seitenwechsel angegebene, auf linken Seiten der als letzter auf der jeweiligen Seite angegebene Kolummentitel eingesetzt.

11, 12, . . . , 19 = Wie 1, 2, . . . , 9, Lage: innen

21, 22, . . . , 29 = Wie 1, 2, . . . , 9, Lage: außen

31, 32, . . . , 39 = Wie 1, 2, . . . , 9, Lage: links

41, 42, . . . , 49 = Wie 1, 2, . . . , 9, Lage: rechts

71, 72, . . . , 79 = Wie 1, 2, . . . , 9, Lage: an Seitennummer herangerückt.

Der Kolummentitel selbst muss dazu linksbündig (mit Zentrieranweisungen für nach rechts geschobene Kolummentitel) und mit der gleichen Breite gesetzt werden wie die Seitennummer (gegebenenfalls einschließlich deren Verschiebung).

**LINAB** Abstand einer waagerechten Linie unter dem Kolummentitel (die Länge der Linie wird mit dem Parameter **BRE** festgelegt); bei **LINAB** = 0 wird keine Linie erzeugt. <0>

Negative Angabe: Die Ausgabe von Kolummentitel und Linie unter dem Kolummentitel wird unterdrückt, bis sie durch die Steueranweisung &!Q+ oder &!Q. bzw. &!L+ oder &!L. explizit eingeschaltet wird.

**KOLTLU** Lage und Aufteilung der lebenden Kolummentitel unten auf der Seite <0>

0 = kein Kolummentitel am Fuß der Seite

1, 2 = Kolummentitel auf Mitte zentrieren:

1 = Kolummentitel identisch auf linken und rechten Seiten; der zuletzt auf der Seite angegebene Kolummentitel wird eingesetzt

2 = Kolummentitel geteilt (links und rechts verschieden); der zuletzt auf der Seite angegebene Kolummentitel wird eingesetzt

11, 12 = Wie 1, 2, Lage: innen

21, 22 = Wie 1, 2, Lage: außen

31, 32 = Wie 1, 2, Lage: links

41, 42 = Wie 1, 2, Lage: rechts

71, 72 = Wie 1, 2, Lage: an Seitennummer herangerückt.

Der Kolumnentitel selbst muss dazu linksbündig (mit Zentrieranweisungen für nach rechts geschobene Kolumnentitel) und mit der gleichen Breite gesetzt werden wie die Seitennummer (gegebenenfalls einschließlich deren Verschiebung).

- KLEINR Ausrücken (in Punkt) des lebenden Kolumnentitels <0>  
Ein linksbündig gesetzter Kolumnentitel (und ggf. die zugehörige Linie unter dem Kolumnentitel) wird um die (negativ anzugebende) Punktzahl nach links über den Satzspiegelrand ausgerückt. Linksbündig gesetzt werden Kolumnentitel bei KOLTL = 31 bis 39, auf Seiten mit gerader Nummer bei KOLTL = 21 bis 29, auf Seiten mit ungerader Nummer bei KOLTL = 11 bis 19.
- KLEINRU Ausrücken (in Punkt) des lebenden Kolumnentitels unten auf der Seite <0>  
Angaben analog zu KLEINR

## Abschnittsgrenzen

- § Absatz. [ XI ] <0 3/2 10.>
- LEER1 Leerzeilen bzw. Freiraum vor dem Abschnitt. <0>  
Negative Angabe: der Vorschub fällt weg, wenn unmittelbar vorher eine Titelzeile bzw. Einschaltung steht, für welche der Freiraum LEER2 nicht negativ angegeben ist.
- NOCH Anzahl der Zeilen, die in einer Spalte jeweils noch frei sein müssen, damit ein durch § codierter Abschnitt (außer nach &!W und nach Titelzeilen) noch in dieser Spalte begonnen wird. <LEER1 + 3/2>
- NEINR Einrückung (in Punkt) der ersten Zeile des Abschnitts <10.>  
Negative Angabe: Ein unmittelbar nach einer Überschrift oder einer Einschaltung oder nach einer Blindzeile (siehe unter 7.2) beginnender Abschnitt wird ohne Einzug gesetzt.

### Anmerkung:

Bei den Parametern \$ \$\$ & && &&& &n& können die Werte in Vielfachen des Zeilenabstands der Grundschrift oder in typographischen Punkt angegeben werden. Im letzteren Fall steht ein » . « hinter der entsprechenden Zahl. Vielfache von Zeilenabständen können ganzzahlig oder als Bruch (z. B. »3/2«) angegeben werden.

Der über die Angabe NOCH verlangte Freiraum wird so weit wie möglich für Text reserviert. Falls in einer der nachfolgenden Zeilen eine mehrzeilige Fußnote beginnt, wird – außer der ersten Zeile – nur so viel von der Fußnote noch unten in die Spalte gebracht, dass nach Möglichkeit der verlangte Freiraum für Textzeilen verfügbar

bleibt. Durch negative Angabe von NOCH kann verlangt werden, dass das Unterbringen von Fußnotenzeilen vorrangig vor der Platzreservierung für weitere Textzeilen erfolgen soll.

Kommen in einer Spalte mehrere Steueranweisungen vor, für die eine Angabe zu NOCH vorgesehen ist (einschließlich der Steueranweisung \$\$\$-n\$\$\$, vgl. in der Beschreibung unter 7.1.), und ist an der Stelle, an der eine solche Steueranweisung vorkommt, das Ende des in der vorangehenden Steueranweisung geforderten Raums noch nicht erreicht, so wird der größere der beiden Werte »jetzt verlangter Freiraum« und »Rest vom zuletzt verlangten Freiraum« für Text reserviert.

Vgl. auch die Anmerkung am Ende der Beschreibung des Parameters &n&.

## Einschaltungen

\$\$	Einschaltungen. [ XI ] <7. 5. 23. 10.>
LEER1	<p>Leerzeilen bzw. Freiraum (variabel) vor der Einschaltung. &lt;7.&gt;</p> <p>Statt des variablen Freiraums kann fester Freiraum verlangt werden, indem unmittelbar hinter LEER1 ein »!« angegeben wird.</p> <p>Negative Angabe: der Vorschub fällt weg, wenn unmittelbar vorher eine Titelzeile bzw. Einschaltung steht, für welche der Freiraum LEER2 nicht negativ angegeben ist.</p>
LEER2	<p>Leerzeilen bzw. Freiraum (variabel) nach der Einschaltung &lt;5.&gt;</p> <p>Statt des variablen Freiraums kann fester Freiraum verlangt werden, indem unmittelbar hinter LEER2 ein »!« angegeben wird.</p> <p>Negative Angabe: Alle Vorschübe nach der Einschaltung werden durch unmittelbar nachfolgende Vorschubanweisungen angegeben (»unmittelbar« heisst: nur Steueranweisungen für Kolummentitel dürfen davor stehen). Folgt unmittelbar nach »Einschaltung-Ende« keine Vorschubanweisung, so wird der hier negativ angegebene Vorschub ausgeführt.</p> <p>\$\$\$+4\$\$\$ nach Einschaltung-Ende bedeutet bei negativem LEER2 einen Vorschub um insgesamt nur 4 Punkt.</p>
NOCH	<p>Anzahl der Zeilen, die in einer Spalte jeweils noch frei sein müssen, damit eine Einschaltung (außer nach &amp;!w und nach Titelzeilen) noch in dieser Spalte begonnen wird &lt;LEER1 + 3/2&gt;</p>
NEINR	Einrückung der ganzen Einschaltung <10.>

Vgl. auch die Anmerkung am Ende der Beschreibung der Parameter § und &n&.

## Zwischenüberschriften (Titelzeilen)

- & Titelzeilen Stufe 1. [ XI ] <3/2 1/2 0 0 5>
- LEER1 Leerzeilen bzw. Freiraum (variabel) vor der Titelzeile der Stufe 1. <3/2>
- Negative Angabe: der Vorschub fällt weg, wenn unmittelbar vorher eine Titelzeile steht, für welche der Freiraum LEER2 nicht negativ angegeben ist.
- Statt des variablen Freiraums kann fester Freiraum verlangt werden, indem unmittelbar hinter LEER1 ein »!« angegeben wird.
- Wird zu LEER1 ein Wert angegeben, der größer als die Satzspiegelhöhe ist, so wird ein Seitenwechsel nur ausgeführt, wenn diese Überschrift nicht auf eine andere Überschrift folgt.
- LEER2 Leerzeilen bzw. Freiraum (variabel) nach der Titelzeile der Stufe 1. <1/2>
- Negative Angabe analog zu negativem LEER2 bei §§.
- Statt des variablen Freiraums kann fester Freiraum verlangt werden, indem unmittelbar hinter LEER2 ein »!« angegeben wird.
- LEER3 Leerzeilen bzw. Freiraum vor der Titelzeile der Stufe 1, wenn sie oben in eine neue Spalte kommt <0>
- LEER4 Leerzeilen bzw. Freiraum (variabel) vor der Titelzeile der Stufe 1, wenn sie unmittelbar auf eine Titelzeile der gleichen oder einer anderen Stufe oder auf eine Einschaltung folgt. <0>
- Die Voreinstellung 0 bedeutet, dass der Gesamtvorschub zwischen den beiden Überschriften bzw. zwischen Einschaltung und Überschrift dem größeren der beiden Vorschübe vor dieser Überschrift bzw. nach der vorhergehenden Überschrift entspricht, falls LEER2 für die vorhergehende Überschrift nicht negativ angegeben ist.
- Bei positivem LEER4 wird der vor dieser Titelzeile verlangte Freiraum um den nach der vorhergehenden Titelzeile verlangten Freiraum verringert, falls dieser nicht negativ angegeben ist und LEER1 positiv ist (d. h.: der Gesamtfreiraum wird nach Möglichkeit auf den Wert reduziert, der dem größeren der beiden hier zusammentreffenden

Freiräume entspricht). Bei negativem LEER4 wird der Freiraum zwischen den beiden Titelzeilen auf den Betrag des angegebenen Wertes reduziert; dies gilt auch für die Angabe  $-0$  (vgl. die Anmerkung am Ende der Beschreibung von  $\&n\&$ ).

Statt des variablen Freiraums kann fester Freiraum verlangt werden, indem unmittelbar hinter LEER4 ein  $\rangle! \langle$  angegeben wird.

NOCH Anzahl der Zeilen, die in einer Spalte jeweils noch frei sein müssen, damit eine Titelzeile der Stufe 1 noch in dieser Spalte begonnen wird.  $\langle \text{LEER1} + \text{LEER2} + 3 \rangle$

Vgl. auch die Anmerkung am Ende der Beschreibung der Parameter  $\$$  und  $\&n\&$  sowie die Anmerkung zu  $\rangle\$\$\$\$-n\$\$\$\$\langle$  in der Beschreibung der Steueranweisungen (siehe unten Kapitel 7.1).

$\&\&$  Titelzeilen Stufe 2. [ XI ]  $\langle 2 \ 1 \ 0 \ 0 \ 8 \rangle$

LEER1 (wie bei  $\&$ , für Stufe 2)  $\langle 2 \rangle$

LEER2 (wie bei  $\&$ , für Stufe 2)  $\langle 1 \rangle$

LEER3 (wie bei  $\&$ , für Stufe 2)  $\langle 0 \rangle$

LEER4 (wie bei  $\&$ , für Stufe 2)  $\langle 0 \rangle$

NOCH (wie bei  $\&$ , für Stufe 2)  $\langle \text{LEER1} + \text{LEER2} + 5 \rangle$

Vgl. auch die Anmerkung am Ende der Beschreibung der Parameter  $\$$  und  $\&n\&$ .

$\&\&\&$  Titelzeilen Stufe 3. [ XI ]  $\langle 0 \ 1 \ 0 \ 0 \ 999 \rangle$

LEER1 (wie bei  $\&$ , für Stufe 3)  $\langle 0 \rangle$

LEER2 (wie bei  $\&$ , für Stufe 3)  $\langle 1 \rangle$

LEER3 (wie bei  $\&$ , für Stufe 3)  $\langle 0 \rangle$

LEER4 (wie bei  $\&$ , für Stufe 3)  $\langle 0 \rangle$

NOCH (wie bei  $\&$ , für Stufe 3)  $\langle 999 \rangle$

Vgl. auch die Anmerkung am Ende der Beschreibung der Parameter  $\$$  und  $\&n\&$ .

$\&n\&$  Titelzeilen Stufe 1.n (für n können die Ziffern 1 bis 9 und die Buchstaben von a bis w stehen). [ XI ]

$\langle 3/2 \ 1/2 \ 0 \ 0 \ 5 \ 9+2 \ 288 \ 0 \rangle$

LEER1 (wie bei  $\&$ , für Stufe 1.n)  $\langle 3/2 \rangle$   
( $-999$ : kein Vorschub vor der ersten Zeile)

LEER2 (wie bei  $\&$ , für Stufe 1.n)  $\langle 1/2 \rangle$

LEER3	(wie bei &, für Stufe 1.n) <0>
LEER4	(wie bei &, für Stufe 1.n) <0>
NOCH	(wie bei &, für Stufe 1.n) <LEER1+LEER2+3>
NGRAD <sub>n</sub> +NDURCH <sub>n</sub>	Schriftgrad und Durchschuss für Titelzeilen der Stufe 1.n <9+2>
	Hinter dem Schriftgrad kann anstelle des Durchschusses auch der gewünschte Zeilenabstand angegeben werden. Statt durch »+« ist der Zeilenabstand durch »//« vom Schriftgrad zu trennen. Die Angaben 9+2 und 9//11 sind also gleichwertig.
NBR <sub>Tn</sub>	Satzbreite für Titelzeilen der Stufe 1.n <NBR <sub>T</sub> >
NICHTU	0 = Normalfall: wird wie Titelzeile behandelt 1 = folgt auf diese Titelzeile eine andere, so wird dort LEER4 nicht ausgewertet (d. h.: eine mit &n&{ abgeschlossene Titelzeile, für die NICHTU = 1 angegeben ist, zählt für diesen Zweck nicht als Titelzeile).

**Anmerkung:**

Steht im Text unmittelbar vor einer der Steueranweisungen \$ \$\$ & && &&& &n& eine Steueranweisung für Zeilenvorschub (außer \$\$\$; vor \$ und \$\$ auch eine Anweisung für Titelzeilen-Ende oder Einschaltung-Ende), so wird der Vorschub vor der neuen Titelzeile bzw. Einschaltung bzw. dem neuen Abschnitt um einen (dort schon ausgeführten) Zeilenvorschub verringert, es sei denn, dass der Freiraum LEER2 nach der vorhergehenden Titelzeile bzw. Einschaltung negativ angegeben ist.

Folgen Einschaltungen unmittelbar aufeinander, oder folgt unmittelbar auf eine Titelzeile eine Einschaltung oder umgekehrt, oder folgt unmittelbar auf eine Titelzeile oder eine Einschaltung ein durch »\$« codierter Abschnitt, so wird der Vorschub vor der neuen Titelzeile bzw. Einschaltung bzw. dem neuen Abschnitt um den nach der davor stehenden Titelzeile bzw. Einschaltung verlangten Freiraum LEER2 verringert, falls dort nicht ein negativer Wert angegeben ist und LEER1 positiv ist (d. h.: der Gesamtfreiraum wird nach Möglichkeit auf den Wert reduziert, der dem größeren der beiden hier zusammentreffenden Freiräume entspricht).

Folgen Titelzeilen unmittelbar aufeinander, so gilt die gleiche Regel, wenn LEER4 für die neue Titelzeile 0 ist. Ist LEER4 nicht 0, so wird dies als Freiraum zwischen den beiden Titelzeilen eingesetzt (es sei denn, dass der durch Vorschübe hinter der ersten der beiden Titelzeilen entstandene Freiraum durch Unterdrücken weiterer Vorschübe nicht mehr auf LEER4 reduziert werden kann). Ist LEER4 negativ oder -0, so wird der Gesamtfreiraum zwischen den beiden Titelzeilen auf den Betrag von LEER4 reduziert.

Vgl. die Anmerkung am Ende der Beschreibung des Parameters §.

## Fußnoten

<b>FN</b>	Fußnoten-Gestaltung. [ XI ] <1 0 9 10 10 1 0 0 36 1/2 0 0>
IFNLIN	<p>0 oder negativ: keine Fußnotenlinie          positiv: Linie (36 bzw. IFNLNG Punkt lang) zwischen Fußnoten und Text. Ist IFNLAB (8. Zahlenwert zu Parameter FN) größer als IFNLIN, so wird der Wert von IFNLAB auch für IFNLIN eingesetzt.</p> <p>Der (positiv oder negativ angegebene) Betrag von IFNLIN gibt den Abstand zwischen Text und Fußnoten in Vielfachen des Grundschrift-Zeilenabstands bzw. in Punkt an. Dieser Wert wird intern auf die nächste um 3 Punkt größere ungerade Zahl erhöht. Sind die Angaben kleiner als ein halber Grundschrift-Zeilenabstand, so wird dafür ein Grundschrift-Zeilenabstand (eine Leerzeile) eingesetzt. Ist ein fester Vorschub IFNLAB zwischen Fußnotenlinie und erster Fußnotenzeile angegeben, so wird der dort angegebene Wert eingesetzt, falls er größer ist als der zu IFNLIN angegebene Wert.</p> <p>Der Abstand zwischen Text und Fußnoten wird bevorzugt zum Spaltenhöhenausgleich herangezogen. Ist die Spalte zu hoch geworden, wird er jedoch nicht unter den zu IFNMAB (siehe unten) angegebenen Mindestabstand verringert. Wird zu IFNLAB ein von 0 verschiedener Wert angegeben, so wird nur der Freiraum vor der Fußnotenlinie zum Spaltenhöhenausgleich herangezogen.</p>
IFNZLI	<p>0 = Fußnotennummer eingerückt          1 = Fußnotennummer linksbündig          n = Fußnotennummer linksbündig mit festem Abstand von n Bildlinien zum Text der ersten Zeile          -n = Fußnotennummer eingerückt mit festem Abstand von n Bildlinien zum Text der ersten Zeile</p>
IFNZNO	<p>9 = Fußnotennummer aus hochgestellten Standard-Ziffern (das sind die Ziffern, die mit einfachen Ziffern codiert werden)          -9 = Fußnotennummer aus hochgestellten Alternativziffern (das sind die Ziffern, die mit #.1 usw. codiert werden)          1 = Fußnotennummer aus Standard-Ziffern          -1 = Fußnotennummer aus Alternativziffern</p>
IFNEN1	n = Text der ersten Fußnotenzeile soll um n Punkt vom linken Satzspiegelrand eingerückt werden.

- $n+m$  = wie  $n$ , aber zuvor ganze erste Zeile um  $m$  Punkt vom linken Satzspiegelrand einrücken.
- Ist für  $n$  der Wert 0 angegeben, so beginnt der Text  $1/4$  Geviert hinter der letzten Ziffer der (linksbündig gesetzten) Fußnotennummer.
- Ist bei  $n > 0$  die verlangte Einrückung zu klein für die Fußnotennummer, so wird der Text ggf. über die Fußnotennummer belichtet, falls kein fester Abstand zum Text verlangt wird.
- IFNENF  $n$  = Folgezeilen der Fußnoten sollen um  $n$  Punkt eingerückt werden.
- Ist für  $n$  der Wert 999 angegeben, so werden die Folgezeilen so weit eingerückt wie die erste Zeile.
- IFNSUB  $n$  = Schrift-Umschaltbereich, aus dem die Fußnotenziffern gesetzt werden sollen.
- IFNKUR 0 = Fußnotenziffern sollen gerade gesetzt werden.  
1 = Fußnotenziffern sollen schräg gesetzt werden (muss auch angegeben werden, wenn in IFNSUB eine Kursivschrift steht).
- IFNLAB 0 = Abstand zwischen Fußnotenlinie und erster Fußnotenzeile wird vom Programm festgelegt und wird ggf. mit zum Spaltenhöhenausgleich herangezogen.  
 $n$  = fester Vorschub ( $n$ . Punkt bzw.  $n$  Vielfache des Grundschrift-Zeilenabstands; Gesamtvorschub, nicht: Freiraum) zwischen Fußnotenlinie und erster Fußnotenzeile.  
Ist IFNLAB größer als IFNLIN (1. Wert zu Parameter FN), so wird der Wert von IFNLAB auch für IFNLIN eingesetzt.
- IFNLNG  $n$  = Länge der Fußnotenlinie in Punkt (muss geradzahlig sein).
- $n+m$  = Die  $n$  Punkt lange Fußnotenlinie wird um  $m$  Punkt vom linken Satzspiegelrand eingerückt.
- $n;nf$  = Auf Seiten, auf denen der Fußnotenblock nicht mit einer neuen Fußnote, sondern mit einer von der vorhergehenden Seite überlaufenden Fußnotenzeile beginnt, soll die Linie  $nf$  statt  $n$  Punkt lang sein.
- $n:l$  = Die Fußnotenlinie wird aus Linienelementen einer Schrift der Größe  $l$  (genauer:  $4/3 * l$ ) statt der Grundschrift-Größe (genauer:  $4/3 * \text{Grundschrift-Größe}$ ) aufgebaut. Soll sowohl die Linienstärke verändert als auch die Linie eingerückt werden, so kann  $n:l+m$  oder  $n+m:l$  bzw.  $n;nf:l+m$  oder  $n;nf+m:l$  angegeben werden.



IFNMAB	n =	Mindestabstand ( $\gg n \ll$ Punkt bzw. $\gg n \ll$ Vielfache des Grundschrift-Zeilenabstands) zwischen Text und Fußnoten.
IFNGRO	n =	Schriftgrad für die Fußnotenziffern.  Ist zu IFNZNO 9 oder -9 angegeben, so wird der zu IFNGRO angegebene Wert n nicht direkt als Schriftgrad für die Fußnotenziffern übernommen, sondern gibt an, dass die Größe und Hochstellung der Fußnotenziffern an eine um $n/4$ Punkt modifizierte Schriftgröße angepasst werden soll. Die Angaben zu IFNOB werden dann ignoriert.  Ist zu IFNGRO nichts oder 0 angegeben, so werden Schriftgrad und Hochstellung der Fußnotenziffern so berechnet, als würden die Fußnotenziffern mit der Steueranweisung #' x hochgestellt.
IFNOB	n =	Hochstellung für die Fußnotenziffern. Angabe n in Halbpunktschritten.  Zu IFNOB kann nur eine Angabe gemacht werden, wenn zu IFNGRO ein von 0 verschiedener Wert angegeben wird.

## Beispiele:

FN 1 -250 9 10 999 ergibt eingerückte hochgestellte Fußnotennummern mit  $1/4$  Geviert Abstand zum Text, der mit 10 Punkt Abstand vom linken Rand beginnt, falls die Fußnotennummern kurz genug sind. Es gibt keine Doppelbelichtung zwischen Fußnotennummer und Text; die Folgezeilen sind jeweils so weit eingerückt wie der Text der ersten Zeile.

FN 1 -333 1 0 999 ergibt linksbündige (weil der Text der ersten Zeile nicht eingerückt ist) Fußnotennummern aus normalen Ziffern mit  $1/3$  Geviert Abstand zum Text; die Folgezeilen sind jeweils so weit eingerückt wie der Text der ersten Zeile.

FN 1 -333 9 20 20 ergibt eingerückte Fußnotennummern mit  $1/3$  Geviert Abstand zum Text, der mit 20 Punkt Abstand vom linken Rand beginnt. Die Folgezeilen sind um jeweils 20 Punkt vom linken Rand eingerückt.

**FNN** Fußnotenverweisziffern. [ XI ] <9 0 0 0>

IFNVZ	9 =	als Fußnotenverweisziffern werden hochgestellte Standard-Ziffern (das sind die Ziffern, die mit einfachen Ziffern codiert werden) benutzt. Die Größe der Ziffern ist an die jeweils benutzte Schriftgröße angepasst.
	8 =	Wie 9, jedoch ist die Größe der Ziffern an die Schriftgröße des Grundtextes angepasst, wenn der augenblicklich verwendete Schriftgrad größer ist als der des Grundtextes.

- 2 = Wie 8, aber Fußnotenzählung beginnt in jeder Spalte neu mit »1«.
- 1 = Wie 9, aber Fußnotenzählung beginnt in jeder Spalte neu mit »1«.
- 12 = Wie 8, aber Fußnotenzählung beginnt auf jeder Seite neu mit »1«.
- 11 = Wie 9, aber Fußnotenzählung beginnt auf jeder Seite neu mit »1«.
- 22 = Wie 8, aber Fußnotenzählung beginnt in jeder Spalte neu mit »1«. Fußnoten mit identischem Wortlaut werden in einer Spalte nur einmal ausgegeben; die Fußnotenverweisnummern werden entsprechend angepasst.
- 21 = Wie 9, aber Fußnotenzählung beginnt in jeder Spalte neu mit »1«. Fußnoten mit identischem Wortlaut werden in einer Spalte nur einmal ausgegeben; die Fußnotenverweisnummern werden entsprechend angepasst.
- 32 = Wie 8, aber Fußnotenzählung beginnt auf jeder Seite neu mit »1«. Fußnoten mit identischem Wortlaut werden auf einer Seite nur einmal ausgegeben; die Fußnotenverweisnummern werden entsprechend angepasst.
- 31 = Wie 9, aber Fußnotenzählung beginnt auf jeder Seite neu mit »1«. Fußnoten mit identischem Wortlaut werden auf einer Seite nur einmal ausgegeben; die Fußnotenverweisnummern werden entsprechend angepasst.
- 9 = als Fußnotenverweisziffern werden hochgestellte Alternativziffern (das sind die Ziffern, die mit # . 1 usw. codiert werden) benutzt. Die Größe der Ziffern ist an die jeweils benutzte Schriftgröße angepasst.
- 8 = Wie -9, jedoch ist die Größe der Ziffern an die Schriftgröße des Grundtextes angepasst, wenn der augenblicklich verwendete Schriftgrad größer ist als der des Grundtextes.
- 2 = Wie -8, aber Fußnotenzählung beginnt in jeder Spalte neu mit »1«.
- 1 = Wie -9, aber Fußnotenzählung beginnt in jeder Spalte neu mit »1«.
- 12 = Wie -8, aber Fußnotenzählung beginnt auf jeder Seite neu mit »1«.
- 11 = Wie -9, aber Fußnotenzählung beginnt auf jeder Seite neu mit »1«.
- 22 = Wie -8, aber Fußnotenzählung beginnt in jeder Spalte neu mit »1«. Fußnoten mit identischem Wortlaut werden in einer Spalte nur einmal ausgegeben; die Fußnotenverweisnummern werden entsprechend angepasst.

- 21 = Wie -9, aber Fußnotenzählung beginnt in jeder Spalte neu mit »1«. Fußnoten mit identischem Wortlaut werden in einer Spalte nur einmal ausgegeben; die Fußnotenverweisnummern werden entsprechend angepasst.
- 32 = Wie -8, aber Fußnotenzählung beginnt auf jeder Seite neu mit »1«. Fußnoten mit identischem Wortlaut werden auf einer Seite nur einmal ausgegeben; die Fußnotenverweisnummern werden entsprechend angepasst.
- 31 = Wie -9, aber Fußnotenzählung beginnt auf jeder Seite neu mit »1«. Fußnoten mit identischem Wortlaut werden auf einer Seite nur einmal ausgegeben; die Fußnotenverweisnummern werden entsprechend angepasst.
- 0 = es werden keine Fußnotenverweisziffern gesetzt; die Codierung %1 etc. wird nur zum Einsteuern der Fußnoten selbst benutzt. Der Verweis auf die Fußnote kann auf diese Weise durch beliebige Zeichen vorgenommen werden. Das gleiche gilt auch für die Fußnotenziffern am Anfang jeder Fußnote.
- \* = statt der Fußnotenverweisziffern werden Sternchen zum Verweis auf die Fußnote(n) und zur Kennzeichnung der Fußnoten benutzt; ist mehr als eine Fußnote zu einer Spalte vorhanden, so werden für die zweite und die weiteren Fußnoten die entsprechende Zahl von Sternchen (maximal 3) benutzt.
- IFNVZV    n = die Fußnotenverweisziffern sollen zusätzlich um n Halbpunktschritte nach oben (positives n) bzw. nach unten (negatives n) verschoben werden.
- IFNRUCK    0 = Normalfall: Fußnotenaufruf und Fußnotennummer müssen aufeinander abgestimmt sein.  
               1 = Wenn eine Fußnote aufgerufen wird, die eine niedrigere Nummer als die vorherige Fußnote hat, aber noch Fußnoten mit höherer Nummer vorhanden sind, werden letztere zuvor nachgetragen.
- IFNVGR    0 = Normalfall: keine Abweichung von der Standardgröße der Fußnotenverweisziffern (1 Punkt mehr als die Hälfte der zuletzt verwendeten Schriftgröße; vgl. Kapitel 12.3.)  
               n = Die Größe der Fußnotenverweisziffern soll an eine um  $n/4$  Punkt modifizierte Schriftgröße angepasst werden (auch negatives n ist möglich).

## Apparate

<b>AP<sub>n</sub></b>	Angaben zum (textkritischen) Apparat $n$ ( $n = 1$ bis $9$ ). [ 1 ] <>
NAPBR	Breite von Apparat $n$ (in Punkt)
NAPBR1	Breite der 1. Zeile des Apparates $n$ in Punkt
IAPPL	Abstand zum vorhergehenden Apparat (Angabe negativ, wenn die einzelnen Apparateinträge jeweils in einer neuen Zeile beginnen sollen; positiv, wenn die Apparateinträge fortlaufend – mit größerem Wortzwischenraum zwischen den einzelnen Einträgen – als Block gesetzt werden sollen).  Der Wert kann in Vielfachen des Zeilenabstands der Grundschrift (auch als Bruch) oder in typographischen Punkt angegeben werden. Im letzteren Fall ist ein ».« hinter die Zahl zu setzen.
IAPPGR	Schriftgrad des $n$ -ten Apparates
IAPPD	Durchschuss des $n$ -ten Apparates
IAPPBL	Zusatzangabe zur Berechnung des Platzbedarfs der Apparateinträge von Apparat $n$ . Hier kann eine 1 angegeben werden, wenn die vorläufige Berechnung des Platzbedarfs des Apparats zu niedrige Werte ergibt. Dies kann insbesondere bei langen (textlastigen) Apparateinträgen sinnvoll sein. Voreinstellung ist 0

Bei den Parametern AP 2 bis AP 9 kann statt dieser Angaben in Spalte 11 die Nummer eines zuvor angegebenen Parameters AP 1 bis AP 8 stehen, um anzugeben, dass die betreffenden Apparate gleichgesetzt (d. h. als ein und derselbe Apparat behandelt) werden sollen. Dies ermöglicht die Verschachtelung von Apparateinträgen, die sich auf mehr als ein Wort des Textes beziehen. Kettenbildung ist möglich (d. h. der Parameter AP<sub>n</sub>, auf den sich die Nummer in Spalte 11 bezieht, kann selbst wieder einen solchen Verweis enthalten).

Steht in Spalte 11 eine 0, so wird der betreffende Apparat nicht berücksichtigt (d. h. wie Kommentar behandelt).

Die Parameter AP 1 bis AP 9 müssen lückenlos aufsteigend angegeben werden. Kommt ein bestimmter Apparat im Text nicht vor, so muss der entsprechende Parameter mit leerem Informationsfeld (keine Angabe ab Spalte 11) angegeben werden.

## Makros

### MAC

Makro-Definitionen. [ XI ] <>

Ab Spalte 11 wird die Makro-Abkürzung in der Form `&.ab` oder `<name>` und unmittelbar dahinter die Makro-Auflösung angegeben. Reicht eine Zeile für die Auflösung nicht aus, so kann die Auflösung in der nachfolgenden Zeile fortgesetzt werden. Diese muss die gleiche Makro-Abkürzung haben wie die fortzusetzende Zeile.

Rechtsbündige Leerzeichen werden ignoriert.

Makros dürfen nicht geschachtelt sein. Ausnahme: Makros der Gruppe A dürfen Makros der Gruppe B enthalten.

Makro-Gruppe A: Alle Makros, die nicht zu Gruppe B gehören.

Makro-Gruppe B: Mit `&.ab` codierte leere Makros (das sind Makros, zu denen hinter dem Makronamen keine Auflösung angegeben ist) sowie mit `&.ab` codierte Makros, deren Auflösung mit `&! (` beginnt und die nur hardware-nahe Anweisungen für die Setzmaschine (vgl. Kapitel 22 bei den Steueranweisungen) enthalten. Die Auflösung dieser Makros muss mit der entsprechenden schließenden Klammer enden.

Die mit `&.ab` codierten Makros der Gruppe A werden nicht nur für die Satzausgabe, sondern auch für die ZIEL-Datei und die PROTOKOLL-Datei aufgelöst.

Die mit `<name>` codierten Makros der Gruppe A werden nur dann in der ZIEL-Datei und der PROTOKOLL-Datei aufgelöst, wenn ihre Auflösung Leerzeichen (ggf. außer einem einzigen Leerzeichen unmittelbar nach der spitzen Klammer hinter dem Makronamen) enthält oder wenn dies mit dem Parameter MAZ verlangt wird. Andernfalls werden die Makro-Abkürzungen unverändert in die ZIEL-Datei und in die PROTOKOLL-Datei übernommen.

Makros der Gruppe B werden nur für die Satzausgabe aufgelöst; in der ZIEL-Datei und in der PROTOKOLL-Datei bleiben die Makro-Abkürzungen stehen.

Soll bei Makros der Gruppe A ein Leerzeichen am Ende der Makro-Auflösung stehen bleiben, so kann hinter das Leerzeichen die Zeichenfolge `&.`  geschrieben werden; diese wird bei der Auflösung der Makros nicht übernommen.

Bei Makros, die SGML/XML-Tags mit Attributen entsprechen, führen unterschiedliche Attribute nicht immer zu unterschiedlichen Makro-Auflösungen. Sollen diese Tags identische Auflösungen erhalten, so genügt es, als Makro nur den Namen des Tags, gefolgt von einem Leerzeichen, in den spitzen Klammern anzugeben. Dieses Makro gilt dann

für alle im Text vorkommenden Tags mit diesem Namen, unabhängig von ihren Attributen, und (falls solche nicht explizit angegeben sind) für Tags mit gleichen Namen, aber ohne Attribute.

Sind die Attribute für die Auflösung der Makros von Bedeutung, so muss jeweils das volle Tag mit allen auszuwertenden Attributen als Makro angegeben werden. Solche Makros können in der Regel, wie unten beschrieben, gleichgesetzt werden.

Sind nicht alle Attribute für die Auflösung eines Makros von Bedeutung, so genügt es, das Tag nur bis zum letzten auszuwertenden Attribut anzugeben und vor der schließenden Spitzklammer ein Leerzeichen einzufügen. Dieses Makro gilt dann für alle Tags, die bis zum letzten angegebenen Attribut gleich lauten.

Es können (einschließlich der mit dem Parameter MAH angegebenen, bis zu 2000 hierarchischen Makros) maximal 3000 Makros angegeben werden; die Auflösungen der mit den Parametern MAC , MAA und MAH angegebenen Makros können insgesamt 64000 Zeichen umfassen.

**MAA** Makro-Definitionen. [ XI ] <>

Wie MAC, jedoch werden die mit <name> codierten Makros auch durch die Standard-Makros \*AUMBRUCH und \*SUMBRUCH nicht aufgelöst.

Die mit Parameter MAA definierten Makros dürfen keine Anweisungen enthalten, die in \*AUMBRUCH oder \*SUMBRUCH verarbeitet werden müssen (z. B. Zeilenwechsel, Überschriften, Einrückungen, Auszeichnungen).

**MAH** Makro-Definitionen für hierarchische Makros. [ XI ] <>

Wie MAC für mit <name> codierte Makros. Bei der Auflösung dieser Makros werden SGML/XML-Konventionen zugrundegelegt; die Stellung der (den SGML/XML-Tags entsprechenden) Makros innerhalb von übergeordneten, durch entsprechende Tags bezeichneten Elementen wird berücksichtigt. Dabei wird eine Hierarchiestufe durch ein einem Start-Tag entsprechendes Makro (z. B. <div>) eröffnet und durch ein einem End-Tag entsprechendes Makro mit identischem Namen, vor dem ein »/« steht (z. B. </div>), abgeschlossen.

Die Reihenfolge der Makros muss der hierarchischen Ordnung der durch sie bezeichneten Elemente des zu setzenden Textes entsprechen. Außerdem wird ab Spalte 11 für jedes übergeordnete Makro, das einem noch offenen (nicht durch ein End-Tag abgeschlossenen) Tag entspricht, ein Punkt erwartet. Fehlen diese Punkte, so wird bei jedem nicht mit </ beginnenden, vom vorhergehenden verschiedenen Makro um eine Hierarchiestufe weiterschaltet und mit jedem mit </ beginnenden, vom vorhergehenden verschiedenen Makro um eine Hierarchiestufe zurückgeschaltet.

Es empfiehlt sich, Makros, die unabhängig von ihrer Stellung in der Hierarchie immer die selbe Auflösung haben (z. B. Tags von Inline-Elementen), mit `MAC` statt mit `MAH` zu definieren. Makros, die in der Regel die selbe Auflösung haben und nur gelegentlich hierarchieabhängig anders aufgelöst werden müssen, können sowohl mit `MAC` als auch mit `MAH` definiert werden. Dabei muss die Definition mit `MAC` vor der Definition des selben Makros mit `MAH` erfolgen. Die hierarchieabhängige Definition hat jeweils Vorrang.

Für Makros, die den End-Tags von Tags mit Attributen entsprechen, gelten die von SGML/XML gewohnten Konventionen (`</name>`, ohne Leerzeichen und ohne Attribute).

Parameter für Satzmakros, die diesen Konventionen entsprechen, können mit dem Standard-Makro `*TAGS` (siehe unten) aus dem zu setzenden, nach SGML/XML-Konventionen ausgezeichneten Text erzeugt werden. In die so erzeugten Parameter muss nur noch die Auflösung der Makros eingetragen werden.

Unterschiedliche hierarchische Makros, die sich auf der selben Hierarchiestufe befinden und die den selben übergeordneten Makros untergeordnet sind, können für das Satzprogramm gleichgesetzt werden, auch wenn sie unterschiedlich aufgelöst werden sollen. Voraussetzung ist, dass die Auflösung der diesen Makros untergeordneten Makros nicht von den Namen dieser gleichgesetzten Makros, sondern nur von ihrer Stellung innerhalb der Hierarchie der Tags abhängt.

Die Gleichsetzung von Makros wird dadurch angegeben, dass beim ersten der gleichzusetzenden Makros in Spalte 9 ein »-«, bei den mit diesem gleichgesetzten Makros (die unmittelbar darauf folgen müssen) ein »=« angegeben wird. Auch die Makros, die eine durch gleichgesetzte Makros eingeleitete Hierarchiestufe abschließen, müssen in Spalte 9 entsprechend gekennzeichnet sein. Soll (abweichend von SGML- bzw. XML-Konventionen) ein einziges (End-)Makro diese Funktion für mehrere gleichgesetzte (Start-)Makros übernehmen, so muss dieses (End-)Makro in Spalte 9 durch ein »-« gekennzeichnet sein, obwohl ihm keine weiteren, mit ihm gleichgesetzten (End-)Makros folgen.

Eine auf der obersten Hierarchiestufe beginnende Folge von verschachtelten `MAH`-Makros, die in anderen Folgen von `MAH`-Makros nicht angegeben ist, wird dort automatisch eingesetzt, wenn die Tag-Folge im Text dies erfordert. Derzeit ist es nicht möglich, dies mehrfach verschachtelt zu nutzen.

Sollen bei Tags mit Attributen Attribut-Werte unverändert in die Makro-Auflösung übernommen werden, so kann statt des Attributs ein »\*« angegeben werden; in der Makro-Auflösung wird an der Stelle, an der der Wert des Attributs eingesetzt werden soll, `{=an}` angegeben, wobei für `n` die laufende Nummer des im Tag durch »\*« ersetzten Attributs ist.

Beispiel:

Mit dem Parameter

MAH `<page n="*" />` (Seite {=a1})

wird erreicht, dass aus dem Quelltext

"weiter unten `<page n="13" />` beschrieben."

beim Satz folgender Text erzeugt wird:

»weiter unten (Seite 13) beschrieben.«

MAH `<see kap="*" pg="*">` (Seite {=a2}, "{=a1}")

ergibt beim Quelltext

"haben wir bereits `<see kap="Einleitung" pg="3">` beschrieben"

in der Satzausgabe:

»haben wir bereits (Seite 3, »Einleitung«) beschrieben.«

## MAZ

Zusatzangaben zu den Makros. [ I ] <0 0>

NGKU Angabe zur Groß- und Kleinschreibung in den Makronamen.

0 = Groß- und Kleinbuchstaben gelten in Makronamen als gleichwertig.

1 = Groß- und Kleinbuchstaben werden unterschieden; auch Makros, die sich nur durch Groß- bzw. Kleinschreibung unterscheiden, gelten als unterschiedliche Makros.

MAUFL Angabe zur Auflösung von Makros in der ZIEL-Datei und der PROTOKOLL-Datei.

0 = Mit `<name>` codierte Makros werden unaufgelöst in die ZIEL-Datei und die PROTOKOLL-Datei übernommen, es sei denn, dass ihre Auflösung – außer einem einzigen Leerzeichen unmittelbar hinter der den Makronamen abschließenden spitzen Klammer – Leerzeichen enthält. Diese Angabe ist nicht erlaubt, wenn Makros mit dem Parameter MAH definiert werden.

1 = Statt der mit `<name>` codierten Makros wird immer deren Auflösung in die ZIEL-Datei und die PROTOKOLL-Datei übernommen.

-1 = Mit `<name>` codierte Makros werden unaufgelöst in die ZIEL-Datei und die PROTOKOLL-Datei übernommen, sofern nicht zwischen Bestandteilen ihrer Auflösung beim Satz ein Zeilenwechsel erfolgt. In diesem Fall wird die Auflösung des Tags zwischen XML-Kommentare der Form `<!-- {tagname} -->` und `<!-- \{tagname} -->` eingeschlossen. Solche Tags können mit dem Standard-Makro \*XMLZIEL wiederhergestellt werden. Dies ist auch dann notwendig, wenn die ZIEL-Datei eines Satzlaufes, bei



- dem zu MAUFL -1 angegeben war, als QUELL-Datei eines weiteren Satzlaufes benutzt werden soll, in dem der Parameter MAH benutzt wird.
- 2 = Mit <name> codierte Makros werden immer unaufgelöst in die ZIEL-Datei und die PROTOKOLL-Datei übernommen. In diesem Fall ist i. d. R. die ZIEL-Datei nur dann als QUELL-Datei für einen weiteren Satzlauf geeignet, wenn sie nicht die Anweisung &!u enthält.

Wird der Parameter MAH verwendet, und ist der Parameter MAZ nicht angegeben, so gilt MAZ 1 -1 als Voreinstellung. Außerdem darf dann als zweiter Wert (MAUFL) nur 1, -1 oder -2 angegeben werden.

## Umdefinition von Zeichen und Dicken

- Geviertstrich soll statt des Halbgeviertstrichs für Gedankenstrich und »bis«-Strich benutzt werden. [ I ] <>

DICKN Dickenwert des Geviertstriches in Bildlinien.

- AFZ** Angaben für die Behandlung der mit " bzw. #. ? codierten Anführungs- und Schlusszeichen. [ XI ] <0>

- IFRZA 0 = guillemets, deutsche Belegung: Spitze nach innen  
 1 = guillemets, französische Belegung: Spitze nach außen  
 2 = guillemets, finnisch: Spitze nach rechts  
 -1 = deutsche Anführungszeichen: vorn Komma bzw. Doppelkomma, hinten einfacher bzw. doppelter umgekehrter Apostroph  
 -2 = englische Anführungszeichen: vorn einfacher bzw. doppelter umgekehrter Apostroph, hinten einfacher bzw. doppelter Apostroph  
 -3 = italienische Anführungszeichen: vorn einfacher bzw. doppelter umgekehrter Apostroph, hinten Komma bzw. Doppelkomma  
 -4 = polnische Anführungszeichen: vorn Komma bzw. Doppelkomma, hinten einfacher bzw. doppelter Apostroph  
 -5 = finnische Anführungszeichen: vorn und hinten einfacher bzw. doppelter Apostroph

Dieser Parameter betrifft nur die Codierungen " und #. ?; die mit #.> bzw. #.< und mit #.: bzw. #.; codierten Zeichen werden nicht umgedreht, die Codierungen #., #.' und #." behalten ihre Bedeutung unverändert.

<b>BIL</b>	Bild-Information für Zeichen: Austauschen bzw. Ergänzen von Schriftzeichen in den Schrift-Umschaltbereichen 1–16. [ XI ] <>
ZNR	Zeichennummer. Die Zeichennummer setzt sich zusammen aus Schriftnummer $\times 1000$ + (oktaler) Zeichennummer.
BER	Schrift-Umschaltbereich (1–16, vgl. Parameter SCH)
NRSCHR	Nummer der im Bereich BER verwendeten Schrift
PA	Adresse, auf der das Zeichen stehen soll. Die vom Satzprogramm intern verwendeten Zeichenadressen orientieren sich an den sogenannten »Primäradressen« des Digiset (siehe Tabelle S. 1308). Falls in Umschaltbereich 3 eine Kapitälchenschrift angegeben ist, sind dort die Adressen 129–148 frei für Zeichen, die mit dem BIL-Parameter ergänzt werden. Weitere freie Adressen sollten bei Bedarf erfragt werden.
DICK	Dicke (in Bildlinien) des Zeichens
	[Kommentar]
<b>UNT</b>	Angaben für zusätzliche Unterstreichung mit #^0+ bis #^9+ bzw. für die Höhe des Unterstreichungsstrichs.
NRUA	Nummer der Unterstreichungsanweisung (0 für #^0+, 1 für #^1+, ..., 9 für #^9+)
ZNR	Zeichennummer: Schriftnummer $\times 1000$ + (oktale) Zeichennummer des Zeichens, das für die Unterstreichung verwendet werden soll
DICK	Dicke des mit ZNR ausgewählten Zeichens
PROZG	Größe (in Prozent der Schriftgröße des zu unterstreichenden Textes), in der das Zeichen für die Unterstreichung benutzt werden soll. Der Wert wird auf das nächste Vielfache von Viertelpunkt-Schritten abgerundet.
TIEF	Angabe zur Tiefstellung (in Prozent der Schriftgröße des zu unterstreichenden Textes) des Zeichens, das zur Unterstreichung benutzt werden soll. Der für die Tiefstellung errechnete Wert wird auf ganze Zahlen gerundet; die Tiefstellung erfolgt um die errechnete Zahl von Halbpunktschritten.
	Wird zu ZNR der Wert 0 angegeben, so wird mit diesem Parameter für die Steueranweisungen #0+ bis #3+ und #5+ bis #6+ die vertikale Lage der Unterstreichungsstriche gegenüber der Voreinstellung geändert. Ein zu TIEF angegebener positiver Wert n verschiebt den Unterstreichungsstrich um $n/2$ Punkt nach oben, ein negativer Wert um $ n /2$ Punkt nach unten.

**DIC** Änderung von Dickenwerten für Zeichen, für die Soll-Spatienbreite und für die Sperrung. [ 1 ]

Zur Angabe von zusätzlichen oder neuen Dickenwerten sind folgende Zahlenwerte anzugeben:

DICKN Neuer Dickenwert in Bildlinien  
 BER Schrift-Umschaltbereich (1, 2, ..., 16)  
 NRSCHR Nummer der im Bereich BER verwendeten Schrift  
 PA Primäradresse (im Digiset-Code, siehe Tabelle S. 1308) des Zeichens, dessen Dicke geändert werden soll. Zur Änderung der Dicke von #(TM) ist statt der Adresse die Zeichenfolge »tm« oder »TM« (ohne Anführungszeichen) anzugeben.

[Kommentar]

Zur Angabe von neuen Dickenwerten für die Unicode-Zeichen #[2007] bzw.  $\text{^\&^\#x2007}$ ; (figure space), #[2008] bzw.  $\text{^\&^\#x2008}$ ; (punctuation space), #[2009] bzw.  $\text{^\&^\#x2009}$ ; (thin space), #[200A] bzw.  $\text{^\&^\#x200A}$ ; (hair space) und #[202F] bzw.  $\text{^\&^\#x202F}$ ; (narrow no-break space) sind folgende Zahlenwerte anzugeben:

NDICK Neuer Dickenwert in Bildlinien  
 IND1 0  
 IND2 0  
 HEXC Hexadezimaler Zeichencode (4 Hexadezimalziffern)

[Kommentar]

Zur Änderung der Soll-Breite von Spatien und Sperrung sind folgende Zahlenwerte anzugeben:

NSZLR Zähler und  
 NSNENR Nenner (Geviert-Bruchteile) für die veränderte Soll-Breite  
 NSIND 0 = die angegebenen Werte gelten für das Spatium (Voreinstellung: 1/3 Geviert)  
 -1 = die angegebenen Werte gelten für die Sperrung mit #S+ bzw. #S\* (Voreinstellung: 1/8 Geviert)

**SP** Sperrung am Wortanfang und am Wortende. [ 1 ] <0 1>

ISPWA Sperrung am Wortanfang  
 0 = Bei »Sperrung Anfang« nach einem Spatium wird erst nach dem ersten Zeichen dahinter mit der Sperrung begonnen.  
 1 = Bei »Sperrung Anfang« nach einem Spatium wird sofort mit der Sperrung begonnen.  
 ISPW3 Sperrung am Wortende  
 0 = Bei »Sperrung Ende« unmittelbar vor einem Spatium endet die Sperrung schon vor dem Spatium.

- 1 = Bei »Sperrung Ende« unmittelbar vor einem Spatium wird auch hinter dem »Sperrung Ende« noch ein entsprechender Zwischenraum erzeugt (analog zu »Sperrung Ende« innerhalb eines Wortes).

## Übersicht über die Voreinstellungen

Wird zur Spezifikation PARAMETER nichts oder »-« angegeben, so ist dies gleichwertig mit den folgenden Angaben:

```

PRO      <Projektname>
LAU      1 1
BER      0.0-999999.999
FAN      0.0
MAX      99999999 99999999 999999
ABB      20 2 1000 2400
SCH      31801 31802 31804 31803 31901
DRT      PS-10
GRO      9+2 8+1 8+1 10+2 12+2 9+2 8+2 8+2
BRE      288 288 288 288 288 288 288 288 288
HOE      438 2 0 0 1
SEI      1 1 1 0 0 0 0 1 0
SPA      1 1 0
MON      1 0 288 12 12
AUS      0 0 0 1/10 1/3 1 0
SIL      0 1/6 1/12 5 2 2 4
RSR      1
ZLN      0
KOL      0 0 0 0 0
$        0 3/2 10.
$$       7. 5. 23. 10.
&        3/2 1/2 0 0 5
&&       2 1 0 0 8
&&&      0 1 0 0 999
&1&     3/2 1/2 0 0 5 9+2 288 0
        usw. bis
&9&     3/2 1/2 0 0 5 9+2 288 0
FN       1 0 9 10 10 1 0 0 36 1/2 0 0
FNN      9 0 0 0
AFZ      0
DIC      1 3 0
DIC      1 8 -1

```

## Alphabetisches Verzeichnis der Parameter

<b>\$</b>	Absatz . . . . .	1178
<b>\$\$</b>	Einschaltungen . . . . .	1179
<b>&amp;</b>	Titelzeilen Stufe 1 . . . . .	1180
<b>&amp;&amp;</b>	Titelzeilen Stufe 2 . . . . .	1181
<b>&amp;&amp;&amp;</b>	Titelzeilen Stufe 3 . . . . .	1181
<b>&amp;n&amp;</b>	Titelzeilen Stufe 1.n . . . . .	1181
<b>–</b>	Geviertstrich . . . . .	1193
<b>ABB</b>	Zahl der erlaubten Fehlerkommentare . . . . .	1154
<b>AFZ</b>	Belegung der Anführungszeichen . . . . .	1193
<b>APn</b>	Angaben zum (textkritischen) Apparat n . . . . .	1188
<b>AUS</b>	Austreiben der Zeilen . . . . .	1170
<b>BER</b>	Angabe eines Bereichs aus der QUELL-Datei . . . . .	1153
<b>BIL</b>	Bild-Information für Zeichen . . . . .	1194
<b>BRE</b>	Breite des Satzspiegels . . . . .	1163
<b>DIC</b>	Änderung von Dickenwerten . . . . .	1195
<b>DRT</b>	Druckertyp für die PROTOKOLL-Datei . . . . .	1153
<b>FAN</b>	Fußnotenanzfang . . . . .	1154
<b>FN</b>	Fußnoten-Gestaltung . . . . .	1183
<b>FNN</b>	Fußnotenverweisziffern . . . . .	1185
<b>GRO</b>	Schriftgrad und Durchschuss . . . . .	1161
<b>HBU</b>	Hebräische und arabische Textteile umdrehen . . . . .	1161
<b>HOE</b>	Höhe des Satzspiegels . . . . .	1164
<b>KAP</b>	Sonderbehandlung der Kapitälchenschrift . . . . .	1160
<b>KOL</b>	Lebende Kolumnentitel . . . . .	1176
<b>LAU</b>	Behandlung des Satzprogrammlaufs . . . . .	1152
<b>MAA</b>	Makro-Definitionen . . . . .	1190
<b>MAC</b>	Makro-Definitionen . . . . .	1189
<b>MAH</b>	Makro-Definitionen für hierarchische Makros . . . . .	1190
<b>MAL</b>	Marginalien am linken Rand . . . . .	1175
<b>MAR</b>	Marginalien am rechten Rand . . . . .	1175
<b>MAX</b>	Maximum für Testzwecke . . . . .	1154
<b>MAZ</b>	Zusatzangaben zu den Makros . . . . .	1192
<b>MLS</b>	Marginalien innerhalb des Satzspiegels . . . . .	1175
<b>MON</b>	Seitenmontage . . . . .	1169
<b>MRS</b>	Marginalien innerhalb des Satzspiegels . . . . .	1175
<b>PRO</b>	Projekt-Identifikation . . . . .	1152
<b>RSR</b>	Rechtschreibreform von 1996: Trennregeln . . . . .	1173
<b>SCH</b>	Zuordnung von Schriften . . . . .	1154
<b>SEI</b>	Seitennummerierung . . . . .	1167
<b>SGM</b>	Schriftgrößen-Modifikation . . . . .	1161
<b>SIL</b>	Silbentrennung . . . . .	1171
<b>SLW</b>	Schriftlaufweitenänderungen . . . . .	1161
<b>SP</b>	Sperrung am Wortanfang und am Wortende . . . . .	1195
<b>SPA</b>	Spaltenzahl . . . . .	1169
<b>TXB</b>	Textbeginn . . . . .	1154
<b>UNT</b>	Angaben für Unterstreichungen mit #^0+ bis #^9+ . . . . .	1194

---

<b>VSS</b>	Umsetzung von Versal-ß . . . . .	1160
<b>ZLN</b>	Zeilennummerierung . . . . .	1173

## Steueranweisungen

Für alle Steueranweisungen ist die Eingabecodierung angegeben, die auf Geräten zu wählen ist, die mit der internationalen Referenz-Version des ASCII-Zeichensatzes (entspricht DIN 66003, Code-Tabelle 1 bzw. ISO-Norm 646) ausgestattet sind.

Die mit »^« beginnenden Codierungen (Kapitel 12.2 bis 18) werden über die TUSTEP-Eingabekommandos (z. B.: #EDIERE, #UMWANDLE) in die entsprechenden TUSTEP-internen Codes umgewandelt, die das Satzprogramm erwartet.

Die (durch Großbuchstaben wiedergegebenen) Buchstaben in den Anweisungen können Groß- oder Kleinbuchstaben sein. Die Kleinbuchstaben in der Beschreibung der Anweisungen stehen für Zahlenwerte; dafür ist, wenn nichts anderes angegeben ist, eine vorzeichenlose ganze Zahl einzusetzen. Ist eine Zahl mit einer bestimmten Stellenzahl verlangt, so sind kleinere Zahlen ggf. nach links mit Nullen auf die verlangte Stellenzahl aufzufüllen.

Die Steueranweisungen sind im Folgenden nach Sachgruppen geordnet beschrieben. Im Anhang findet sich eine alphabetische Liste der Steueranweisungen.



## I. Unterteilung des Textes

### 1. Seitenumbruch

Die unter 1. genannten Anweisungen müssen zwischen Leerzeichen stehen. Zeilenanfang und Zeilenende gelten als Leerzeichen.

Eine neue Spalte wird nur begonnen, wenn in der aktuellen Spalte schon Text (nicht nur übergelaufene Fußnoten) gesetzt wurde.

Beim Aufruf des Satzprogramms kann über die Spezifikationsangabe `MODUS=A` verlangt werden, dass die automatische Seiten- bzw. Spaltenaufteilung unterbleibt und nur an den Stellen Seiten- bzw. Spaltenwechsel vorgenommen wird, an denen eine der hier genannten Anweisungen angegeben ist.

Die Anweisungen zum Seiten- bzw. Spaltenwechsel stehen in zwei Formen, mit drei bzw. mit zwei `&`, zur Verfügung. Bei den mit drei `&` geschriebenen Anweisungen werden evtl. noch wirksame Einschaltungen und Einzüge aufgehoben; es wird auf den Grundtext-Schriftgrad umgeschaltet. Mit den mit zwei `&` geschriebenen Seiten- bzw. Spaltenwechselanweisungen ist es möglich, mitten in einem Abschnitt Seiten- bzw. Spaltenwechsel vorzunehmen: der Schriftgrad wird nicht verändert, Einschaltungen und Einzüge werden nicht aufgehoben; ist Blocksatz verlangt, so wird die Zeile vor einer dieser Anweisungen ausgetrieben, wenn sie nicht durch eine Zeilenwechselanweisung abgeschlossen ist.

Entstehen durch die Anweisungen `&&&R&&&{`, `&&&L&&&{`, `&&&-n&&&{`, `&&&+n&&&{`, `&&&*n&&&{` Leerseiten, so werden diese mit ausgegeben, wenn es sich dabei um nicht mehr als 7 unmittelbar aufeinander folgende Leerseiten handelt. Dies gilt auch für die entsprechenden, mit nur zwei `&` geschriebenen Anweisungen.

Siehe auch Anweisung `$$$-n$$$` (bedingter Spaltenwechsel, Kapitel 7.1)

<b><code>&amp;&amp;&amp;&amp;</code></b>	Logisches Dateieinde Der Rest des Textes aus der Eingabedatei wird ignoriert. Dies gilt auch, wenn <code>&amp;&amp;&amp;&amp;</code> innerhalb von Kommentar (also nach <code>&amp;X</code> ) steht.
<b><code>&amp;&amp;&amp;R&amp;&amp;&amp;{</code></b>	Neue rechte Seite (= Seite mit ungerader Nummer)
<b><code>&amp;&amp;&amp;L&amp;&amp;&amp;{</code></b>	Neue linke Seite (= Seite mit gerader Nummer)
<b><code>&amp;&amp;&amp;N&amp;&amp;&amp;{</code></b>	Neue rechte oder linke Seite
<b><code>&amp;&amp;&amp;S&amp;&amp;&amp;{</code></b>	Neue Spalte (bei einspaltigem Satz gleichbedeutend mit <code>&amp;&amp;&amp;N&amp;&amp;&amp;{</code> )
<b><code>&amp;&amp;&amp;-n&amp;&amp;&amp;{</code></b>	Seitenwechsel und Weiterschalten der Seitennummer auf Seite <code>n</code> Ist das Programm bereits auf der Seite <code>n</code> oder einer Seite mit einer höheren Nummer angekommen, so wird keine neue Seite begonnen; die Steueranweisung <code>&amp;&amp;&amp;-n&amp;&amp;&amp;{</code> wirkt dann wie eine Zeilenwechselanweisung.

- &&&-0&&&{** Seitenwechsel auf linke Seite (Seite mit gerader Nummer), Unterdrücken der Ausgabe von Seitennummer und Kolumnentitel.
- Diese Steueranweisung ist vor allem am Dateiende sinnvoll, um die Ausgabe mit einer leeren linken Seite abzuschließen, falls die letzte nicht leere Seite eine rechte Seite (= eine Seite mit ungerader Seitennummer) ist. Ist das Programm bereits auf einer linken Seite angekommen, so wird keine neue Seite begonnen; die Steueranweisung **&&&-0&&&{** wird dann ignoriert.
- &&&+n&&&{** Seitenwechsel und Weiterschalten der Seitennummer um n Nummern
- &&&=n&&&{** Spaltenwechsel und Weiterschalten der Spaltennummer auf Seite n
- Ist das Programm bereits auf der Spalte n oder einer Spalte mit einer höheren Nummer angekommen, so wird keine neue Spalte begonnen; die Steueranweisung **&&&-n&&&{** wirkt dann wie eine Zeilenwechselanweisung.
- &&&\*n&&&{** Spaltenwechsel und Weiterschalten der (nicht mitgesetzten) Spaltennummer um n Nummern

Bei den Anweisungen **&&&+n&&&{** und **&&&\*n&&&{** wird die Seiten- bzw. Spaltennummer in jedem Fall um die angegebene Zahl weitergeschaltet, auch wenn gerade eine neue Seite bzw. Spalte begonnen war (z. B. nach Einschaltung-Ende).

Wenn beispielsweise genau eine Leerseite gewünscht ist, ist es also sicherer, die Kombination **&&&N&&&{ &&&+1&&&{** anstelle der Anweisung **&&&+2&&&{** zu verwenden.

- &&R&&&{** Seitenwechsel im Abschnitt, analog zu **&&&R&&&{**
- &&L&&&{** Seitenwechsel im Abschnitt, analog zu **&&&L&&&{**
- &&N&&&{** Seitenwechsel im Abschnitt, analog zu **&&&N&&&{**
- &&S&&&{** Spaltenwechsel im Abschnitt, analog zu **&&&S&&&{**
- &&-n&&&{** Seitenwechsel im Abschnitt, analog zu **&&&-n&&&{**
- &&+n&&&{** Seitenwechsel im Abschnitt, analog zu **&&&+n&&&{**
- &&=n&&&{** Spaltenwechsel im Abschnitt, analog zu **&&&=n&&&{**
- &&\*n&&&{** Spaltenwechsel im Abschnitt, analog zu **&&&\*n&&&{**

Bei den Anweisungen zum Seiten- bzw. Spaltenwechsel im Abschnitt werden Einschaltungen, Einzüge und Auszeichnungen nicht aufgehoben, der Schriftgrad wird nicht verändert; bei Blocksatz wird die Zeile vor diesen Anweisungen ausgetrieben, wenn sie nicht durch eine Zeilenwechselanweisung abgeschlossen ist.

Ein weiterer Unterschied betrifft die Behandlung von Seiten mit übergelaufenen Fußnoten bzw. vorgemerkten Freiräumen für zu lange Apparate: Mit der Anweisung **&&&N&&&{** wird auch dann eine neue Seite begonnen, wenn nur übergelaufene Fußnoten oder vorgemerkte Freiräume für zu lange Apparate auf der Seite vorhanden sind. Ggf. enthält die Seite vor der mit **&&&N&&&{** beginnenden Seite also nur (übergelaufene) Fußnoten. Mit der Anweisung **&&N&&&{** dagegen wird der Text auf solchen

Seiten weitergeführt. Übergelaufene Fußnoten bzw. Apparate stehen also unten auf der durch `&&N&&{` begonnenen Seite.

- &!N(n)** Einstellen der auszugebenden Seitennummer auf den Wert n. Die Seitennummern in der Ziel- und der Protokoll-Datei werden dadurch nicht beeinflusst.
- &!N(R)** Die Seitennummern sollen ab hier in römischen Ziffern (Großbuchstaben) ausgegeben werden.
- &!N(r)** Die Seitennummern sollen ab hier in römischen Ziffern (Kleinbuchstaben) ausgegeben werden.
- &!N(A)** Die Seitennummern sollen ab hier in arabischen Ziffern ausgegeben werden.
- &!N(a)** wie `&!N(A)`
- &!N(\_)** Die Seitennummern sollen ab hier unsichtbar ausgegeben werden, bis sie durch eine der Anweisungen `&!N(A)` `&!N(a)` `&!N(R)` oder `&!N(r)` wieder sichtbar gemacht werden.
- &!S(2)** Ab dem nächsten Spaltenwechsel die Spaltennummer jeweils um 2 erhöhen
- &!S(1)** Ab dem nächsten Spaltenwechsel die Spaltennummer jeweils um 1 erhöhen.
- &!W** Warten mit Spaltenwechsel bis zum nächsten Wort.  
 Neue Spalte frühestens nach dem Zeilenwechsel beginnen, der nach dem nächsten zu setzenden Wort bzw. Wortteil folgt.  
 In den Anweisungen `&{ &&{ &&&{` für »Titelzeilen-Ende« ist diese Funktion automatisch enthalten (nicht aber in `&n&{`).  
 Die Wirkung der Anweisung `&!W` wird durch die Anweisung `&!Y` sowie durch Spalten- und Seitenwechselanweisungen wieder aufgehoben.  
 Zeilen, die mit `&!W` enden, zählen nicht als Hurenkind, wenn sie als erste Zeile in einer Spalte oder auf einer Seite stehen.
- &!W&!W** Warten mit Spaltenwechsel bis zur nächsten Spaltenwechselanweisung.  
 Neue Spalte frühestens bei der nächsten Anweisung für Spaltenwechsel beginnen.  
 Kommt eine Zeile, in der diese Anweisung steht, beim Satz als erste Zeile in eine neue Spalte, so gilt sie bei nachfolgender Abschnittsgrenze nicht als Hurenkind.
- &!Y** Aufheben von `&!W` und `&!W&!W`

- &!Y(n)** bedingten Spaltenwechsel vormerken  
Sind beim nächsten automatischen Zeilenwechsel in der Spalte nicht noch mindestens  $n$  Zeilen ( $n$  ist eine einstellige Zahl) frei, so wird eine neue Spalte begonnen (wirkt nicht vor Zeilenwechselanweisung!).
- &!Y(n.)** bedingten Spaltenwechsel vormerken.  
Sind beim nächsten automatischen Zeilenwechsel in der Spalte nicht noch mindestens  $n$  Punkt frei, so wird eine neue Spalte begonnen (wirkt nicht vor Zeilenwechselanweisung!).
- &!A** Ausgleichsstelle für kurze Spalten.  
Der nächste Zeilenvorschub ist die Stelle, an der bei zu kurzer Spalte – nach dem Versuch, bei den »variablen« Vorschüben maximal NAUSGL Zeilen auszugleichen (vgl. Parameter HOE) – der gesamte zum Austreiben der Spalte nötige Freiraum gegeben werden soll.  
Variabel sind die Vorschübe vor und nach Titelzeilen und Einschaltungen, vor und nach Fußnotenlinien sowie die mit \$\$\$+n\$\$\$ codierten Vorschübe.  
Soll bei &!A der gesamte Ausgleich vorgenommen werden, so muss für NAUSGL 0 angegeben werden; vgl. die Beschreibung des Parameters HOE.  
Kommen in einer Spalte mehrere Anweisungen &!A vor, so wird der Ausgleich beim ersten &!A vorgenommen.  
Wird eine Spalte auf Grund von &!A ausgetrieben, so wird in der PROTOKOLL-Datei in der Meldung über den Spaltenhöhenausgleich der an variablen Stellen eingesetzt zusätzliche Vorschub als »ausgeglichen« angegeben, unter »mit Durchschuss« wird 999 angegeben.
- &!A+** Ausgleichsstelle für kurze Spalten  
Wie &!A, jedoch wird der verbleibende Raum auf alle Stellen verteilt, an denen auf dieser Seite die Anweisung &!A+ steht.  
Kommen auf einer Seite sowohl &!A als auch &!A+ vor, so hat &!A Vorrang; auf der selben Seite vorkommende Anweisungen &!A+ werden übergangen.
- &!A(n)** Begrenzung für das Austreiben  
Ab der Spalte, in der diese Anweisung steht, wird bei &!A und &!A+ nur ausgetrieben, wenn insgesamt höchstens  $n$  Punkt auszutreiben sind.
- &!A(0)** Begrenzung für das Austreiben aufheben  
Ab der Spalte, in der diese Anweisung steht, wird eine ggf. mit &!A(n) angegebene Begrenzung wieder aufgehoben.

## 2. Kolumnentitel; Spaltenkopftext

### 2.1. Kolumnentitel

Die Anweisungen für lebende Kolumnentitel müssen unmittelbar vor oder nach Anweisungen für Zeilenwechsel stehen, dürfen also nicht im laufenden Text vorkommen. Zentrieranweisungen müssen zuvor (z. B. durch eine Zeilenwechselanweisung oder durch @1) abgeschlossen sein.

Die Stellung der lebenden Kolumnentitel auf der Seite (links, Mitte, rechts, innen, außen) wird beim Aufruf des Satzprogramms mit dem Parameter KOL angegeben. Mit diesem Parameter wird außerdem angegeben, ob die lebenden Kolumnentitel jeweils erst ab dem nachfolgenden Seiten- bzw. Spaltenwechsel berücksichtigt werden oder noch für die aktuell bearbeitete Seite bzw. Spalte gelten sollen. Diese Angaben können für Seiten bzw. Spalten mit ungerader Nummer (rechte Seiten) und für Seiten bzw. Spalten mit gerader Nummer (linke Seiten) verschieden sein. So kann beispielsweise beim Satz von Wörterbüchern auf jeder linken Seite der zuletzt vor dem Seitenwechsel angegebene Kolumnentitel, auf jeder rechten Seite der letzte auf dieser Seite angegebene Kolumnentitel ausgegeben werden.

Als untere Kolumnentitel werden immer die zuletzt vor dem Wechsel zur nächsten Seite bzw. Spalte angegebenen Kolumnentitel eingesetzt. Untere Kolumnentitel können zusätzlich zu den oberen Kolumnentiteln ausgegeben werden; sie können jedoch nur gesetzt werden, wenn die Seitennummer ebenfalls unten gesetzt wird. Sie stehen in der gleichen Zeile wie diese Seitennummer.

Sollen bei mehrspaltigem Satz die Kolumnentitel nicht seitenbezogen, sondern spaltenbezogen gesetzt werden, so sind @= . . . @ { und @" . . . @ { Kolumnentitel für gerade Spaltennummern, @/ . . . @ { und @' . . . @ { Kolumnentitel für ungerade Spaltennummern.

@= . . . @ { Lebende Kolumnentitel oben

Das zwischen den Anweisungen @= und @ { Stehende wird als Kolumnentitel über die Seite bzw. Spalte gesetzt, bei geteiltem Kolumnentitel nur auf den Seiten bzw. Spalten mit gerader Nummer.

@/ . . . @ { Lebende Kolumnentitel oben für rechte Seiten

Das zwischen den Anweisungen @/ und @ { Stehende wird auf den Seiten bzw. Spalten mit ungerader Nummer als Kolumnentitel über die Seite bzw. Spalte gesetzt. Der zuvor mit @= . . . @ { definierte Kolumnentitel gilt damit nur für linke Seiten bzw. für Spalten mit gerader Nummer.

Die Anweisung @/ ist nur sinnvoll, wenn über Parameter KOL geteilte Kolumnentitel verlangt werden.

Sind in einem Auftrag die oberen Kolumnentitel für linke und rechte Seiten gleich, so genügt die Angabe der »Kolumnentitel oben für linke Seite«, falls auf beiden Seiten der zuletzt vor einem Seitenwechsel oder der zuerst nach einem Seitenwechsel angegebene Kolumnentitel eingesetzt werden soll.

- @" . . . @{** Lebende Kolumnentitel unten für linke Seiten  
Das zwischen den Anweisungen @" und @{ Stehende wird auf den Seiten bzw. Spalten mit gerader Nummer als Kolumnentitel unter die Seite bzw. Spalte gesetzt.
- @' . . . @{** Lebende Kolumnentitel unten für rechte Seiten  
Das zwischen den Anweisungen @' und @{ Stehende wird auf den Seiten bzw. Spalten mit ungerader Nummer als Kolumnentitel unter die Seite bzw. Spalte gesetzt.  
Sind in einem Auftrag die unteren Kolumnentitel für linke und rechte Seiten gleich, so genügt die Angabe der »Kolumnentitel unten für linke Seiten«.
- &!Q-** Kolumnentitel ausschalten  
Ausgabe der lebenden Kolumnentitel ab der nächsten Seite bzw. Spalte ausschalten.
- &!Q- (n)** Kolumnentitel ausschalten.  
Wie &!Q-, aber nur für insgesamt n Seiten bzw. Spalten.
- &!Q+** Kolumnentitel einschalten.  
Ausgabe der lebenden Kolumnentitel einschalten.
- &!Q.** Nächsten Kolumnentitel unterdrücken.  
Ausgabe des Kolumnentitels auf der nächsten Seite bzw. Spalte unterdrücken.  
War die Ausgabe des lebenden Kolumnentitels bereits ausgeschaltet, so wird sie auf der übernächsten Seite bzw. Spalte wieder eingeschaltet.
- &!Q. (n)** Nächste Kolumnentitel unterdrücken.  
Wie &!Q., aber für insgesamt n Seiten bzw. Spalten.
- &!L-** Linie unter dem Kolumnentitel ausschalten.  
Ausgabe der Linie unter dem lebenden Kolumnentitel ab der nächsten Seite bzw. Spalte ausschalten.
- &!L- (n)** Linie für n Seiten bzw. Spalten ausschalten.  
Wie &!L-, aber nur für insgesamt n Seiten bzw. Spalten.
- &!L+** Linie unter dem Kolumnentitel einschalten.  
Ausgabe der Linie unter dem lebenden Kolumnentitel einschalten.
- &!L.** Nächste Linie unter dem Kolumnentitel unterdrücken.  
Ausgabe der Linie unter dem Kolumnentitel auf der nächsten Seite bzw. Spalte unterdrücken.  
War die Ausgabe der Linie unter dem lebenden Kolumnentitel be-

reits ausgeschaltet, so wird sie auf der übernächsten Seite bzw. Spalte eingeschaltet.

- &!L. (n)** Nächste n Linien unterdrücken.  
Wie &!L., aber für insgesamt n Seiten bzw. Spalten.
- &!N-** Seitennummer ausschalten.  
Ausgabe der Seitennummer ab der nächsten Seite bzw. Spalte ausschalten.
- &!N- (n)** Nächste n Seitennummern ausschalten  
Wie &!N-, aber nur für insgesamt n Seiten bzw. Spalten.
- &!N+** Seitennummer einschalten.  
Ausgabe der Seitennummer einschalten.
- &!N.** Nächste Seitennummer unterdrücken.  
Ausgabe der Seitennummer auf der nächsten Seite bzw. Spalte unterdrücken.  
War die Ausgabe der Seitennummer bereits ausgeschaltet, so wird sie auf der übernächsten Seite bzw. Spalte eingeschaltet.
- &!N. (n)** Nächste Seitennummern unterdrücken.  
Wie &!N., aber für insgesamt n Seiten bzw. Spalten.

Mit den Anweisungen &!Q. &!L. &!N. bzw. &!Q- &!L- &!N- wird die Ausgabe von Kolumnentitel, Linie bzw. Seitennummer auf bzw. ab der auf die Anweisung folgenden Spalte bzw. Seite unterdrückt.

Die Anweisungen &!Q. &!L. und &!Q- &!L- setzen voraus, dass bereits ein Kolumnentitel definiert ist, dessen Ausgabe unterdrückt werden kann.

Ist die Ausgabe von Kolumnentitel, Linie bzw. Seitennummer schon ausgeschaltet, wenn eine der Anweisungen &!Q. &!L. &!N. gegeben wird, so wird ein Fehlerkommentar ausgegeben und die entsprechende Ausgabe wieder eingeschaltet.

Bei den Anweisungen &!Q+ &!L+ &!N+ wirkt sich das Einschalten jeweils auf die nächsten nach dem Einschalten auszugebenden Seitennummern bzw. Kolumnentitel und Linien aus.

Eine oben stehende Seitennummer wird jeweils vor der ersten Zeile einer Seite ausgegeben, eine unten stehende Seitennummer jeweils nach der letzten Zeile einer Seite. Auch ein unten stehender lebender Kolumnentitel wird jeweils nach der letzten Zeile einer Seite ausgegeben.

Bei den lebenden Kolumnentiteln und den zugehörigen Linien richtet sich der Zeitpunkt der Ausgabe nach der Angabe KOLTL im Parameter KOL. Ist die Einerstelle von KOLTL einer der Werte 1 oder 2, so wird der (zuletzt vor dem Seitenwechsel angegebene) Kolumnentitel vor der Ausgabe der ersten Zeile einer Seite eingesetzt. Bei diesen Werten muss die Ausgabe des Kolumnentitels also vor dem Seitenwechsel ausgeschaltet bzw. wieder eingeschaltet sein. Das gleiche gilt beim Wert 3 für den

Kolumnentitel der linken (geraden) Seiten und beim Wert 4 für den Kolumnentitel der rechten (ungeraden) Seiten.

Bei den Werten 5–9 wird der (auf der Seite selbst angegebene) Kolumnentitel erst nach der Ausgabe der letzten Zeile einer Seite eingesetzt. Bei diesen Werten wirkt sich das Aus- bzw. Einschalten also auf den Kolumnentitel der Seite selbst aus, auf der er aus- bzw. eingeschaltet wird. Das gleiche gilt beim Wert 3 für den Kolumnentitel der rechten (ungeraden) Seiten und beim Wert 4 für den Kolumnentitel der linken (geraden) Seiten.

## 2.2. Spaltenkopftext

Die Steueranweisung für Spaltenkopftext-Anfang muss am Anfang einer neuen Zeile (nach einer Zeilenwechselanweisung oder einer anderen Anweisung, die einen Zeilenwechsel beinhaltet wie Überschrift-Ende) stehen.

**&!T= . . . &!T{** Spaltenkopftext für nachfolgende Seiten.

Der zwischen den Anweisungen `&!T=` und `&!T{` stehende (u. U. mehrzeilige) Text wird als Spaltenkopftext vorgemerkt. Dieser Text wird nicht an der Stelle ausgegeben, an der er als Spaltenkopftext definiert wird, sondern für die Ausgabe in den nachfolgenden Spalten bzw. Seiten innerhalb des Satzspiegels (jeweils oben auf der Seite) vorgemerkt. Ist gleichzeitig auch ein Spaltenkopftext mit `&!T\ . . . &!T{` vorgemerkt, so wird der mit `&!T= . . . &!T{` zur Ausgabe in der ersten Spalte der rechten Seite einer Doppelseite vorgemerkt.

**&!T/ . . . &!T{** Spaltenkopftext für rechte Seiten.

Wie `&!T= . . . &!T{`, jedoch wird der so definierte Spaltenkopftext für die Ausgabe in rechten Seiten bzw. in Spalten mit ungerader Nummer vorgemerkt. Der zuvor mit `&!T= . . . &!T{` definierte Spaltenkopftext gilt damit nur für linke Seiten bzw. für Spalten mit gerader Spaltennummer. Ist gleichzeitig auch ein Spaltenkopftext mit `&!T\ . . . &!T{` vorgemerkt, so wird der mit `&!T/ . . . &!T{` zur Ausgabe in der letzten Spalte einer Doppelseite vorgemerkt.

**&!T\ . . . &!T{** Spaltenkopftext für erste Spalte der Doppelseite.

Wie `&!T= . . . &!T{`, jedoch wird der so definierte Spaltenkopftext bei zweispaltigem Satz für die Ausgabe jeweils in der ersten Spalte von linken Seiten vorgemerkt.

Gleichzeitig mit `&!T\ . . . &!T{` kann nur ein weiterer Spaltenkopftext mit `&!T= . . . &!T{` oder `&!T/ . . . &!T{` vorgemerkt sein.

**&!T= . . . &!T:** Spaltenkopftext für nachfolgende Seiten vormerken und gleichzeitig ausgeben

**&!T/ . . . &!T:** Spaltenkopftext für rechte Seiten vormerken und gleichzeitig ausgeben



**&!T\...&!T:** Spaltenkopftext für erste Spalte der Doppelseite vormerken und gleichzeitig ausgeben

Wird bei der Definition des Spaltenkopftextes als abschließende Steueranweisung **&!T:** statt **&!T{** benutzt, so wird der Spaltenkopftext nicht nur, wie oben beschrieben, vorgemerkt, sondern an der Stelle, an der die Definition erfolgt, auch ausgegeben.

**&!T.** Spaltenkopftext löschen

Der mit **&!T=...&!T{** bzw. **&!T/...&!T{** definierte Spaltenkopftext wird gelöscht. Ab der nächsten Seite bzw. Spalte wird kein Spaltenkopftext mehr ausgegeben. Soll der gleiche oder ein anderer Spaltenkopftext in späteren Spalten wieder ausgegeben werden, so muss er neu definiert werden.

### 3. Titelzeilen (Rubriken, Zwischenüberschriften)

Die Anweisungen für Titelzeilen-Anfang müssen am Zeilenanfang oder nach Leerzeichen stehen, die Anweisungen für Titelzeilen-Ende vor Leerzeichen oder am Zeilenende.

Folgen Titelzeilen unmittelbar aufeinander, so wird als Abstand zwischen diesen der größere der beiden hier zusammentreffenden Freiräume eingesetzt. Statt dessen kann über die Parameter `&`, `&&`, `&&&` und `&n&` ein fester Abstand oder der Wegfall des Freiraums vor der zweiten Titelzeile verlangt werden.

Um den Freiraum nach den Titelzeilen zu variieren, kann statt der zu der jeweiligen Titelzeilen-Anfang-Anweisung gehörenden Titelzeilen-Ende-Anweisung jede beliebige Titelzeilen-Ende-Anweisung verwendet werden.

Für alle Titelzeilen (außer den mit `& . . . & {` codierten Titelzeilen der Stufe 1) kann mit den Parametern `BRE` bzw. `&n&` eine von der Satzspiegelbreite abweichende Zeilenlänge verlangt werden.

Wenn Titelzeilen, die mit `& . . . & {`, `&& . . . && {` oder `&&& . . . &&& {` codiert sind, mehr als 10 Zeilen umfassen, wird im Protokoll eine entsprechende Fehlermeldung ausgegeben. Dies gilt nicht für Titelzeilen, die mit `&n& . . . &n& {` codiert sind. Diese Codierung kann deshalb auch für Einschaltungen etc. verwendet werden.

- |  |  |
|--|--|
| <b>&amp; . . . &amp; {</b>                     | <p>Titelzeile Stufe 1. Schriftgrad wie Grundschrift</p> <p>Stellung (mit Parameter <code>&amp;</code> veränderbar):<br/> 3/2 Leerzeilen (aufgerundet auf ganze Punkt) davor, 1/2 Leerzeile (abgerundet) danach. Sind vor dem Setzen der Titelzeile in der Spalte nicht noch mindestens 5 Zeilen frei, so wird mit der Titelzeile (ohne Freiraum davor) eine neue Spalte begonnen.</p>  |
| <b>&amp;&amp; . . . &amp;&amp; {</b>           | <p>Titelzeile Stufe 2. Eigener Schriftgrad (mit Parameter <code>GRO</code> angebar)</p> <p>Stellung (mit Parameter <code>&amp;&amp;</code> veränderbar):<br/> 2 Leerzeilen davor, 1 Leerzeile danach. Sind vor dem Setzen der Titelzeile in der Spalte nicht noch mindestens 8 Zeilen frei, so wird mit der Titelzeile (ohne Freiraum davor) eine neue Spalte begonnen.</p>  |
| <b>&amp;&amp;&amp; . . . &amp;&amp;&amp; {</b> | <p>Titelzeile Stufe 3. Eigener Schriftgrad (mit Parameter <code>GRO</code> angebar)</p> <p>Stellung (mit Parameter <code>&amp;&amp;&amp;</code> veränderbar):<br/> Beginn einer neuen Spalte, ohne Freiraum vor der Titelzeile, 1 Leerzeile danach.</p>  |
| <b>&amp;n&amp; . . . &amp;n&amp; {</b>         | <p>Titelzeile Stufe 1.n (Stufen 1.1 bis 1.9 und 1.a bis 1.w). Schriftgrad und Stellung wie Titelzeile Stufe 1 (beides mit Parameter <code>&amp;n&amp;</code> veränderbar)</p> <p>Im Unterschied zu den mit <code>&amp;</code>, <code>&amp;&amp;</code> und <code>&amp;&amp;&amp;</code> codierten Titelzeilen entfällt hier die Überprüfung der Zeilenzahl. Mit <code>&amp;n&amp;</code> können daher auch Einschaltungen, Apparate etc. codiert werden.</p> |
| <b>&amp;0&amp; {</b>                           | <p>Ende einer Titelzeile beliebiger Stufe, normaler Zeilenwechsel</p>  |

## 4. Einschaltungen

### 4.1. Einspaltige Einschaltungen (»Petit-Satz«)

- \$\$** Beginn einer Einschaltung
- Eigener Schriftgrad; die ganze Einschaltung ist um 10 Punkt eingerückt und mit zusätzlich 5 Punkt Vorschub (variabel) vom umgebenden Haupttext abgesetzt. Einrückung und zusätzlicher Vorschub können beim Aufruf des Satzprogramms mit dem Parameter \$\$ geändert werden.
- Die Anweisung steht nach Leerzeichen oder am Zeilenanfang; nach dieser Anweisung darf (außer in einem mit der Anweisung &!S(n,mmm) eingeleiteten mehrspaltigen Teil) Leerzeichen oder Zeilenwechsel stehen.
- \$\$ {** Einschaltung Ende
- Die Anweisung steht ohne Leerzeichen nach dem letzten Wort; nach dieser Anweisung steht Leerzeichen oder Zeilenende.
- Einschaltungen können, wenn mit dem Parameter AP<sub>n</sub> Apparate verlangt werden, ineinander geschachtelt werden. Sind keine Apparate verlangt, so macht jedes \$\$ { alle Einschaltungen rückgängig.
- Sollen mehrere Arten von Einschaltungen unterschieden werden, so kann dazu auch die Anweisung &n& (siehe Kapitel 3) benutzt werden.

### 4.2. Mischen von ein- und mehrspaltigem Blocksatz

Sollen ein- und mehrspaltiger Satz gemischt werden, so erfordert dies ein Vorgehen in drei Schritten: nach dem ersten Lauf des Satzprogramms, der für den mehrspaltigen Teil nur die Zeileneinteilung festlegt und den Platzbedarf berechnet, wird mit dem Standard-Makro \*SUMBRUCH die Verteilung der Zeilen auf die Spalten vorgenommen; der endgültige Satz wird mit einem anschließenden weiteren Lauf des Satzprogramms vorgenommen.

Mischen von ein- und mehrspaltigen Teilen ist nur im Textteil, nicht in den Fußnoten möglich.

- &!S(n,mmm)** Beginn des mehrspaltigen Teils, der in n Spalten zu je mmm Punkt Breite zu setzen ist
- Die Anweisung muss als erstes hinter einer Zeilenwechsellanweisung stehen. Für mmm ist eine maximal vierstellige Zahl oder ein Bruch der Form m/n einzusetzen, der den gewünschten Teil der (Grundtext-)Satzspiegelbreite angibt.

Mindestbreite der Spalten: 30 Punkt; Höchstzahl der Spalten: 10 (statt 10 ist für  $n$  eine 0 einzusetzen). Innerhalb des mehrspaltigen Teils darf Zeilenwechsel nur durch  $\$ \$ \$$  verlangt werden.

Absätze, Blindzeilen, Aussparungen und Freiräume (vgl. Kapitel 7.1 bis 7.3) sind im mehrspaltigen Teil nicht erlaubt; Einrückungen (zusätzlich zur Einrückung der Spalte) dürfen nur durch Anweisungen der Form  $\&=nnn$  bzw.  $\&=-nnn$  oder  $\&=(nnn)$  bzw.  $\&=(nnn)$  (d. h.  $\&=$  bzw.  $\&=-$  mit nachfolgender dreistelliger Zahl oder mit nachfolgender bis zu 4-stelliger Zahl in Klammern; siehe unten bei 10.1.2.) codiert werden.

Bevor eine weitere Anweisung  $\&!S(n, mmm)$  folgt, muss der mehrspaltige Satz durch  $\&!S\{$  erst beendet werden; eine Schachtelung ist nicht möglich. Die Anweisungen  $\&$ ,  $\&\&$ ,  $\&\&\&$ ,  $\&n\&$  beenden den mehrspaltigen Satz zwangsweise.

**$\&!S\{$**  Ende des mehrspaltigen Teils

Vor und nach der Anweisung  $\&!S\{$  muss ein Leerzeichen oder Zeilenwechsel stehen; unmittelbar danach muss eine Zeilenwechselanweisung folgen.

$\&!S\{$  ist gleichzeitig Endpunkt für das Austreiben der Zeilen (einschl. der Zentrieranweisungen); zu kurze Zeilen werden, wenn keine Zentrieranweisungen mehr wirksam sind, nicht ausgetrieben; zu lange Zeilen werden zusammengeschoben.

### 4.3. Verändern der Satzbreite

**$\&!S(1, mmm)$**  Verändern der Satzbreite auf  $mmm$  Punkt

Für  $mmm$  ist eine maximal vierstellige Zahl oder ein Bruch der Form  $m/n$  einzusetzen, der den gewünschten Teil der (Grundtext-)Satzspiegelbreite angibt.

Wird in der unter 4.2 beschriebenen Steueranweisung für den Beginn eines mehrspaltigen Teils als Spaltenzahl  $n$  eine 1 angegeben, so dient diese Anweisung dazu, die Satzbreite zeitweilig zu verändern. Die Satzbreite kann auch größer werden als sie für den gerade zu setzenden Textteil mit entsprechenden Parametern angegeben ist. Die Einschränkungen, die für die mehrspaltigen Teile genannt sind, gelten dann nicht. Die Anweisungen  $\&!S(1, mmm)$  und die zugehörige Anweisung  $\&!S\{$  brauchen nicht zwischen Leerzeichen zu stehen.

Die Veränderung der Satzbreite kann nur durch die Anweisung  $\&!S\{$  aufgehoben werden; sie hat insbesondere auch Vorrang vor eigenen Breitenangaben, die für evtl. zwischen  $\&!S(1, mmm)$  und  $\&!S\{$  stehende Titelzeilen beim Aufruf des Satzprogramms mit entsprechenden Parametern angegeben wurden.

- &!S(1,mmm;n)** wie &!S(1,mmm), jedoch für maximal n Zeilen
- &!S(1,Pnn)** wie &!S(1,mmm), jedoch Einstellen der Satzbreite auf den zuletzt mit &!M(nn) gemerkten Wert.
- &!S(1,Pnn;n)** wie &!S(1,Pnn), jedoch für maximal n Zeilen
- &!S(1,Pnn+m)** wie &!S(1,Pnn), jedoch Einstellen der Satzbreite auf den zuletzt mit &!M(nn) gemerkten Wert plus m Punkt.
- &!S(1,Pnn+m;n)** wie &!S(1,Pnn+m), jedoch für maximal n Zeilen
- &!S(1,Pnn-m)** wie &!S(1,Pnn), jedoch Einstellen der Satzbreite auf den zuletzt mit &!M(nn) gemerkten Wert minus m Punkt.
- &!S(1,Pnn-m;n)** wie &!S(1,Pnn-m), jedoch für maximal n Zeilen
- &!S{** Ende des Teils mit veränderter Satzbreite  
 &!S{ ist gleichzeitig Endpunkt für das Austreiben der Zeilen (einschl. der Zentrieranweisungen). Zu kurze Zeilen werden, wenn keine Zentrieranweisungen mehr wirksam sind, nicht ausgetrieben; zu lange Zeilen werden zusammengeschoben.

## 5. Kritische Apparate

Das Satzprogramm kann (zusammen mit dem Standard-Makro \*AUMBRUCH) neben normalen Fußnoten bis zu 9 verschiedene Apparate am Ende der Seite bzw. Spalte verwalten. Es geht davon aus, dass die einzelnen Apparateinträge in der Regel relativ kurz sind (einzelne Wörter bis wenige Zeilen). Im Unterschied zum Satz von Fußnoten ist insbesondere nicht vorgesehen, dass lange Apparateinträge bei Bedarf automatisch auf Folgeseiten bzw. -spalten fortgesetzt werden. Es kann daher sinnvoll sein, längere Anmerkungen als Fußnoten statt als Apparateinträge zu setzen.

Der Satz von Texten, die Apparate enthalten, erfordert drei Arbeitsschritte.

In einem ersten Satzprogrammmlauf (mit MODUS=T) wird der endgültige Zeilenumbruch für die Textzeilen festgelegt und anhand der Angaben in den Parametern AP1 . . . AP9 der Platzbedarf für die Apparate (einschließlich der in den Fußnoten enthaltenen Apparateinträge) berechnet. Die Apparateinträge selbst bleiben dabei noch an den Stellen im Text stehen, an denen sie in der QUELL-Datei stehen.

Dieser Schritt ist ggf. so lange (nach entsprechenden Korrekturen) zu wiederholen, bis keine Fehlerkommentare zu den Apparateinträgen mehr auftreten.

Anschließend werden mit dem Standard-Makro \*AUMBRUCH die Apparateinträge aus dem Text (und ggf. den Fußnoten) herausgezogen, wobei ggf. die Zeilenverweise eingesetzt werden, und mit den für den Satz benötigten Steueranweisungen ans Spaltenende (ggf. unter die normalen Fußnoten) gestellt. Eingabe für dieses Makro ist die PROTOKOLL-Datei des vorhergehenden Satzprogrammmlaufs.

Das Ergebnis von \*AUMBRUCH ist eine Datei, die als QUELL-Datei für einen anschließenden Satzprogrammmlauf (mit MODUS=A, Angabe zum Parameter LAU: 1 oder 3 oder 4) dient. Erst dieser Lauf stellt den endgültigen seitenumbrochenen Satz einschließlich der Apparate her.

**&XXn** Anfang eines Apparateintrags zum Apparat Nr. n. (n ist eine einstellige Zahl zwischen 1 und 9)

Um Verwechslungen mit Kommentaren und Registereinträgen auszuschließen, die mit x oder X beginnen, müssen beide XX in &XXn entweder groß oder klein geschrieben werden. Ist dies nicht gegeben, so wird es als ein mit x bzw. X beginnender Kommentar oder Register-Eintrag interpretiert, z. B. &Xxenophil&x{ oder &xxXenophanes&x{.

**&XX{** Ende des Apparateintrags

Die zwischen &XXn und &XX{ stehende Information wird nur ausgewertet, wenn einer der Parameter AP<sub>n</sub> angegeben ist; andernfalls wirkt die Anweisung wie die Kommentar-Anweisung (vgl. Kapitel 20).

Weitere Einzelheiten sind der Beschreibung des Standard-Makros \*AUMBRUCH zu entnehmen.

## 6. Fußnoten

Der Satz von Texten, die Fußnoten enthalten, erfordert zwei Arbeitsschritte. Zunächst werden die Fußnoten in einem eigenen Satzprogrammmlauf mit `MODUS=F` gesetzt; beim Satz des Textes (Satzprogrammmlauf mit `MODUS=T`) werden sie auf den entsprechenden Seiten bzw. Spalten eingesteuert.

**%n**

Fußnotennummer und Fußnotenverweis

Sowohl die Fußnotennummer vor dem Fußnotentext als auch der Fußnotenverweis im Text besteht aus 1–4 Ziffern, von denen jede durch vorangestelltes % gekennzeichnet ist, und evtl. zusätzlich einem durch % gekennzeichneten Kleinbuchstaben (z. B.: %1%2%3%a; vgl. Kapitel 12.3). %0 ohne vorangehende, durch % gekennzeichnete Ziffer ergibt »\*«, die Folge %0%0 ohne vorangehende durch % gekennzeichnete Ziffer ergibt »\*\*« als Fußnotenverweis und als Fußnoten-Kennung.

Beim Setzen der Fußnoten gilt eine durch ein vorangestelltes % gekennzeichnete Ziffer am Anfang einer Eingabezeile als Steueranweisung für den Beginn einer neuen Fußnote.

Der Aufruf der gesetzten Fußnoten beim Setzen des Textes geschieht über die im Text enthaltenen Fußnotenverweise.

Überlange Fußnoten werden automatisch auf Folgeseiten bzw. –spalten fortgesetzt. Mit dem Parameter `HOE` kann beim Aufruf des Satzprogramms angegeben werden, wieviele Textzeilen auf solchen Folgeseiten bzw. –spalten in jedem Fall gesetzt werden sollen, wenn eine überlaufende Fußnote auch auf dieser Folgeseite bzw. –spalte nicht unterzubringen ist.

**&!T (n)**

Mindestzahl von Textzeilen, die bei überlaufenden Fußnoten auf Folgeseiten gesetzt werden sollen, abweichend von der Angabe im Parameter `HOE` auf `n` setzen.

**&!T ()**

Mindestzahl von Textzeilen, die bei überlaufenden Fußnoten auf Folgeseiten gesetzt werden sollen, wieder auf den im Parameter `HOE` angegebenen Wert setzen.

**&!J (+n)**

Zusätzlicher Raum für Fußnoten

Auf der aktuellen Seite sollen bis zu `n` Zeilen über den Satzspiegel hinaus für (überlaufende) Fußnoten genutzt werden.

**&!J (-n)**

Verringerter Raum für Fußnoten

Auf der aktuellen Seite sollen `n` Zeilen weniger für Fußnoten nutzbar sein als der Satzspiegel erlaubt.

**&!FS**

Ab hier Fußnoten seitenweise nummerieren

**&!FN**

Ab hier Fußnoten mit Original-Nummern ausgeben

Die Anweisungen `&!FS` und `&!FN` müssen im Text und in den Fußnoten an den jeweils entsprechenden Stellen stehen.

**&!FU**

Nächste Fußnotennummer unterdrücken

Werden die Fußnoten spaltenweise / seitenweise nummeriert, wird diese Fußnote auch bei der Nummerierung übergangen.

Wird &!FU nur in der Fußnotendatei verwendet, so wird nur die Fußnotennummer unterdrückt; die Fußnote wird in der Nummerierung aber berücksichtigt.

**&!F-**

Ausschalten der Fußnotenlinie

Auf der aktuellen Seite soll die Fußnotenlinie nicht ausgegeben werden.



## 7. Absätze, Blindzeilen, Freiraum; Einbinden von Grafiken

Das Satzprogramm vermeidet es, die erste Zeile eines Abschnitts als letzte Zeile in einer Spalte (»Schusterjunge«) zu setzen oder mit der letzten (in der Regel kurzen) Zeile eines Abschnitts eine neue Spalte zu beginnen (»Hurenkind«), es sei denn, dass es über Parameter ausdrücklich zugelassen wird.

Die Zeile vor einer Anweisung zum Zeilenwechsel (außer vor \$\$\$\$) wird nicht ausgetrieben. Über Parameter kann für diese Zeilen (außer vor Einschaltung-Anfang und -Ende, Titelzeilen-Anfang und -Ende und vor & :) ein Einzug rechts von mindestens 10 Punkt verlangt werden. Wenn jedoch in der Zeile Anweisungen für »absoluten Einzug« (@-1 etc.) enthalten oder an dieser Stelle noch Zentrieranweisungen wirksam sind, wird kein rechter Einzug vorgenommen.

Das Satzprogramm unterscheidet zwischen festen und variablen Vorschüben; letztere werden bei Bedarf bevorzugt zum Spaltenhöhenausgleich herangezogen.

### 7.1. Absätze

Vor allen unter 7.1 genannten Anweisungen muss Leerzeichen oder Zeilenwechsel stehen; danach dürfen Leerzeichen stehen.

<b>\$</b>	Absatz Neue Zeile mit 10 Punkt Einzug, Abschnittsgrenze; Anordnung mit Parameter \$ änderbar.
<b>&amp;!A-</b>	Einrückung für \$ erzwingen Die Anweisung &!A- vor \$ bewirkt, dass dort auch dann eingerückt wird, wenn die Einrückung wegfallen würde, weil der Abschnitt unmittelbar nach einer Überschrift oder einer anderen Abschnittsgrenze steht.
<b>\$\$\$\$</b>	Neue Zeile stumpf, Abschnittsgrenze
<b>\$\$\$-n\$\$\$</b>	Bedingter Spaltenwechsel, Abschnittsgrenze Neue Zeile stumpf, Abschnittsgrenze; Spalten- bzw. Seitenwechsel, falls in der Spalte bzw. auf der Seite nicht noch n Zeilen frei sind. n ist eine ein- oder zweistellige Zahl zwischen 1 und 99; sollen mehr als 9 freie Zeilen verlangt werden, so kann dies auch in der Form \$\$\$-n\$\$\$-n\$\$\$ angegeben werden, z. B.: \$\$\$-5\$\$\$-9\$\$\$ für 14 Zeilen. Der verlangte Freiraum wird so weit wie möglich für Text reserviert. Falls in einer der nachfolgenden Zeilen eine mehrzeilige Fußnote beginnt, wird – außer der ersten Zeile – nur so viel von der Fußnote noch unten in die Seite bzw. Spalte gebracht, dass nach Möglichkeit der verlangte Freiraum für Textzeilen verfügbar bleibt. Ist an dieser Stelle zusätzlicher Freiraum (Leerzeilen) notwendig, so sollten die entsprechenden Anweisungen vor dieser Anweisung an-

gegeben werden, also z. B. in folgender Reihenfolge:

\$\$\$=\$\$\$/\$\$\$-n\$\$\$

**\$\$\$-n\$\$\$** Wie \$\$\$-n\$\$\$, aber keine Abschnittsgrenze; die Zeile davor wird ausgetrieben, Einrückungen und Zentrierungen werden nicht beendet.

## 7.2. Blindzeilen

Vor allen unter 7.2 genannten Anweisungen muss Leerzeichen oder Zeilenwechsel stehen, danach dürfen (außer in einem mit der Anweisung &!S (n, mmm) eingeleiteten mehrspaltigen Teil) Leerzeichen oder Zeilenwechsel stehen.

Wenn eine der unter 7.2 genannten Anweisungen unmittelbar vor \$\$ steht, hebt sie den zusätzlichen Vorschub vor der Einschaltung auf.

**\$\$\$=\$\$\$** Zeilenwechsel und Leerzeile (fest), Abschnittsgrenze

Die Höhe des Freiraums richtet sich nach dem Zeilenabstand des Textteils, in dem sich diese Anweisung befindet. Der Vorschub ist fest, wird also zum Spaltenhöhenausgleich nicht vorrangig vor normalem Zeilenvorschub verändert.

Am Spaltenende und -anfang fällt die Leerzeile weg.

Mehrere unmittelbar aufeinander folgende Leerzeilenanweisungen müssen zu einer zusammenhängenden Folge zusammengefasst werden (z. B. \$\$\$=\$\$\$=\$\$\$ für zwei Leerzeilen). Werden sie nicht zusammengefasst, so sind die Verhältnisse am Spalten- bzw. Seitenanfang und -ende undefiniert.

**\$\$\$=n\$\$\$** Zeilenwechsel und n Leerzeilen (fest), Abschnittsgrenze

Wie \$\$\$=\$\$\$, jedoch für n Leerzeilen.

**\$\$\$=n. \$\$\$** Zeilenwechsel und n Punkt Freiraum (fest), Abschnittsgrenze

Wie \$\$\$=\$\$\$, jedoch für n Punkt Freiraum. n kann eine mehrstellige Zahl sein.

**\$\$\$/\$\$\$\$** Zeilenwechsel und 1/2 Leerzeile (fest), Abschnittsgrenze

Wie \$\$\$=\$\$\$, jedoch Höhe des Freiraums nur 1/2 Zeilenabstand.

**\$\$\$n\$\$\$** Zeilenwechsel und zusätzlicher Vorschub von n Punkt (fest), Abschnittsgrenze

Wie \$\$\$=\$\$\$, jedoch Höhe des Freiraums n (eine einstellige Zahl von 0–9) Punkt.

**\$\$\$n\$\$\$** Wie \$\$\$n\$\$\$, aber keine Abschnittsgrenze; die Zeile davor wird ausgetrieben.

**\$\$\$+n\$\$\$** Zeilenwechsel mit (variablem) Vorschub von insgesamt n Punkt, Abschnittsgrenze

Im Unterschied zu \$\$\$n\$\$\$ wird insgesamt nur ein Vorschub von n

Punkt ausgeführt.  $n$  ist eine ein- bis dreistellige Zahl. Ist  $n = 0$ , so kann 0 fehlen. Statt  $n$  kann auch »=« ( $$$$+=$$$$$  für einen variablen Vorschub von der Höhe eines Zeilenabstands) oder »/« ( $$$$+/$$$$  für einen variablen Vorschub von der Höhe eines halben Zeilenabstands) eingesetzt werden. Der Vorschub ist variabel, wird also zum Spaltenhöhenausgleich vorrangig vor normalem Zeilenvorschub verändert.

Wenn in einer Sequenz von mehreren Vorschubanweisungen  $$$$+n$$$$  enthalten ist, fällt der normale Zeilenvorschub weg; ggf. überschneidet sich also eine hinter  $$$$+n$$$$  beginnende Zeile mit der vorhergehenden. Dies gilt auch dann, wenn zwischen dieser Anweisung und der nächsten Zeilenwechselanweisung kein Text oder horizontaler Freiraum mehr gesetzt wurde, sondern z. B. nur Kolumnentitel, Marginalien oder Kommentar stehen. Es sollte deshalb ein zusätzlicher Vorschub angegeben werden (also z. B. für Zeilenwechsel an Abschnittsgrenze mit zusätzlich 5 Punkt variablem Vorschub nicht  $$$$+5$$$$  oder  $$$$0$$$+5$$$$ , sondern  $$$$=$$$$+5$$$$ ).

**$$$$+n$$$$**  Wie  $$$$+n$$$$ , aber keine Abschnittsgrenze; die Zeile davor wird ausgetrieben.

### 7.3. Freiraum; Einbinden von Grafiken

Vor und nach den unter 7.3 aufgeführten und mit  $\$$  beginnenden Anweisungen muss ein Leerzeichen oder ein Zeilenwechsel stehen.

Wird in einer solchen Anweisungen hinter # die Nummer einer einzubindenden Grafik angegeben (s. u.), so wird bei der Ausgabe mit dem Standard-Makro \*PSAUS in diese Aussparung die Abbildung mit der angegebenen Nummer eingebunden.

In der Ausgabe des Satzprogramms steht der Aufruf der Abbildung horizontal am linken Satzspiegelrand (bei nach rechts verschobenen Abbildungen an der dabei angegebenen Stelle), vertikal auf der Grundlinie der ersten in der Aussparung nicht mehr gesetzten Zeile. Beginnt eine Grafik in der ersten Zeile einer neuen Spalte, so wird sie um 3/4 der aktuell verwendeten Schriftgröße nach oben verschoben (d. i. an die Oberkante der ersten Zeile statt auf die Schriftgrundlinie).

Einzubindende Abbildungen müssen im EPS-Format (Encapsulated PostScript) vorliegen und, bevor sie ausgegeben werden können, mit dem Standard-Makro \*GRAFIK zum Einmontieren vorbereitet und, mit einer Nummer zwischen 1 und 999999 versehen, in einer Datei gesammelt werden. Diese Datei muss später beim Aufruf des Standard-Makros \*PSAUS zur Spezifikation GRAFIK angegeben werden.

Sollen Abbildungen aus mehr als einer Datei ausgewählt werden, so muss  $\#k/iii$  statt  $\#iii$  (bei mehr als einer Abbildung:  $\#k/iii\#l/jjj$ ) angegeben werden, wobei  $k$  bzw.  $l$  eine Zahl von 1 bis 8 ist und die Nummer der Grafikdatei angibt, aus der die Abbildung mit der Nummer  $iii$  bzw.  $jjj$  ausgewählt werden soll; dabei bezieht sich  $k$  bzw.  $l$  auf die Reihenfolge, in der die Grafikdateien später beim Aufruf von \*PSAUS angegeben werden.

nnn gibt die Höhe des Freiraums jeweils in (Didot-)Punkt an. Wird davor ein ! geschrieben, also z. B. \$\$!7\$\$ { angegeben, so wird damit ein Freiraum von n Grund-schrift-Zeilenabständen verlangt.

**\$\$nnn\$\$ {** Freiraum für eine spaltenbreite Abbildung mit der Höhe von nnn Punkt

Es wird ein Freiraum von nnn Punkt an einem Stück frei gelassen nach der Zeile, in der sich diese Anweisung befindet. nnn ist eine höchstens 4-stellige Zahl, maximal: Satzspiegelhöhe (ohne Kolum-mentitel).

Die Aussparung bzw. der Freiraum beginnt frühestens nach der Zei-le, in der sich die Anweisung befindet. Wenn der verlangte Raum in der Seite bzw. Spalte, in der die Anweisung steht, keinen Platz mehr hat, wird die Seite bzw. Spalte mit Text gefüllt und der Raum auf der folgenden Seite bzw. Spalte oben freigehalten. Das Programm kann sich zwei solcher Freiräume bis zum Beginn der nächsten Spalte merken.

Wurde in einer Spalte schon ein Freiraum für die nächste Spalte deshalb vorgemerkt, weil er in der aktuellen Spalte keinen Platz mehr hat, so wird kein weiterer Freiraum in dieser Spalte freigehalten. Spaltenbreiter Freiraum und Aussparungen, die nicht über die gesamte Satzbreite gehen (vgl. weiter unten), werden dabei ge-trennt behandelt.

Wird Freiraum bzw. Aussparung in Texten mit kritischen Apparaten verlangt, so muss, falls eine entsprechende Anweisung in Zeilen mit Fußnotenverweisen vorkommt, diese nach dem (letzten) Fußno-tenverweis in der Zeile gegeben werden.

Für Aussparung bei Initialen siehe &=(m/n;z) in Kapitel 10.1.2.

**\$\$!nnn\$\$ {** Wie \$\$nnn\$\$ {, aber Vielfache von Grund-schrift-Zeilenabständen (auch mit Grafik-Aufruf, z. B. \$\$!nnn#iii\$\$ {) (siehe unten)

**\$\$!\$\$ {** Freiraum für eine spaltenbreite und spaltenhohe Abbildung (auch mit Grafik-Aufruf, z. B. \$\$!#iii\$\$ {) (siehe unten).  
Steht diese Anweisung direkt nach einem Seitenwechsel, so beginnt der Freiraum am Seitenanfang (und nicht erst nach der ersten Zeile der Seite).

**\$\$0/nnn\$\$ {** Wie \$\$nnn\$\$ {, jedoch wird auch eine nachfolgende nicht seiten-breite Aussparung nicht mehr auf dieser Seite begonnen, wenn die-ser Freiraum für die nächste Seite vorgemerkt werden muss.

**\$\$nnn#iii\$\$ {** Wie \$\$nnn\$\$ {, zusätzlich Einmontieren der Abbildung Nr. iii. Wird für nnn der Wert 0 angegeben, so wird die Abbildung Nummer iii ohne Aussparung beim nächsten Zeilenwechsel eingebunden.

**\$\$nnn#iii:mm\$\$ {** Wie \$\$nnn#iii\$\$ {, dabei Verschieben der Abbildung um mm/2 Punkt nach oben bzw. (bei negativem mm) nach unten. mm kann maximal 2-stellig angegeben werden.

Die Angabe :mm hinter der Grafik-Nummer iii ist bei allen unter 7.3 genannten Anweisungen möglich, mit denen Grafiken eingebunden werden, außer bei  $\$0/0\#iii\$\{$  bis  $\$0/2\#iii=\$\{$ . Die Angabe :mm muss, falls noch Angaben zur horizontalen Positionierung folgen, jeweils als erste Angabe hinter der Grafik-Nummer stehen.

**$\$\$nnn\#iii+eee\$\{\$**  Wie  $\$\$nnn\#iii\$\{\$ , dabei Abbildung vom linken Satzspiegelrand um eee Punkt einrücken.

**$\$\$nnn\#iii+-eee\$\{\$**  Wie  $\$\$nnn\#iii\$\{\$ , dabei in linken Spalten (= Spalten mit gerader Spaltennummer) Abbildung vom linken Satzspiegelrand um eee Punkt einrücken, in rechten Spalten (= Spalten mit ungerader Spaltennummer) Abbildung nicht einrücken.

**$\$\$nnn\#iii+-eee/aaa\$\{\$**  Wie  $\$\$nnn\#iii\$\{\$ , dabei in linken Spalten (= Spalten mit gerader Spaltennummer) Abbildung vom linken Satzspiegelrand um eee Punkt einrücken, in rechten Spalten (= Spalten mit ungerader Spaltennummer) um aaa Punkt einrücken. aaa kann auch negativ sein.

**$\$\$nnn\#iii-+eee\$\{\$**  Wie  $\$\$nnn\#iii\$\{\$ , dabei in linken Spalten (= Spalten mit gerader Spaltennummer) Abbildung nicht einrücken, in rechten Spalten (= Spalten mit ungerader Spaltennummer) Abbildung vom linken Satzspiegelrand um eee Punkt einrücken.

**$\$\$nnn\#iii-+eee/aaa\$\{\$**  Wie  $\$\$nnn\#iii\$\{\$ , dabei in linken Spalten (= Spalten mit gerader Spaltennummer) Abbildung um aaa Punkt einrücken, in rechten Spalten (= Spalten mit ungerader Spaltennummer) Abbildung vom linken Satzspiegelrand um eee Punkt einrücken. aaa kann auch negativ sein.

**$\$\$nnn\#iii+-eee\$\{\$**  Wie  $\$\$nnn\#iii+-eee\$\{\$

Wird vor +eee, +-eee, +eee, +-eee/aaa, +eee/aaa ein Gleichheitszeichen eingefügt (also =+eee, =+-eee, =+eee, =+eee/aaa, =+eee/aaa angegeben), so werden die Einrückungen nicht spaltenbezogen, sondern seitenbezogen vorgenommen.

**$\$\$nnn\#iii\#jjj\$\{\$**  Wie  $\$\$nnn\$\{\$ , zusätzlich Einmontieren der Abbildungen Nr. iii und Nr. jjj. Es können bis zu vier Abbinungsnummern angegeben werden.

**$\$\$nnn\#iii+-eee\#jjj+-eee\$\{\$**  Wie  $\$\$nnn\#iii\#jjj\$\{\$

Für jede der bis zu vier Abbildungen kann durch Angabe von +eee, +-eee, +eee, +-eee/aaa, +eee/aaa bzw. =+eee, =+-eee, =+eee, =+eee/aaa, =+eee/aaa hinter der Abbinungsnummer bestimmt werden, ob und wie weit sie immer, nur in linken oder nur in rechten Spalten bzw. Seiten eingerückt werden soll.

**$\$\$-nnn\$\{\$**  Wie  $\$\$nnn\$\{\$ , jedoch soll der Freiraum erst in der nächsten Spalte oben berücksichtigt werden.

- \$\$!-nnn\$\$**{ Wie \$\$-nnn\$\$, aber Vielfache von Grundschrift-Zeilenabständen (auch mit Grafik-Aufruf, z. B. \$\$!nnn#iii\$\$)
- \$\$-nnn#iii\$\$**{ Wie \$\$-nnn\$\$, zusätzlich Einmontieren der Abbildung Nr. iii.
- \$\$-nnn#iii#jjj\$\$**{ Wie \$\$-nnn\$\$, zusätzlich Einmontieren der Abbildungen Nr. iii und jjj. Es können bis zu vier Abbildungsnummern angegeben werden.
- \$\$\nnn\$\$**{ Wie \$\$\nnn\$\$, jedoch soll der Freiraum unten auf der Seite freigehalten werden.
- \$\$!\nnn\$\$**{ Wie \$\$\nnn\$\$, aber Vielfache von Grundschrift-Zeilenabständen (auch mit Grafik-Aufruf, z. B. \$\$!\nnn#iii\$\$)
- \$\$\nnn#iii\$\$**{ Wie \$\$\nnn\$\$, zusätzlich Einmontieren der Abbildung Nr. iii.
- \$\$\nnn#iii#jjj\$\$**{ Wie \$\$-nnn\$\$, zusätzlich Einmontieren der Abbildungen Nr. iii und jjj. Es können bis zu vier Abbildungsnummern angegeben werden.
- \$\$\nnn\$\$**{ Wie \$\$\nnn\$\$; zusätzlich wird die Satzspiegelhöhe der Folgespalte (bei mehrspaltigem Satz: aller Folgespalten, die zur selben Seite gehören) um die Höhe des Freiraums verringert.
- \$\$\!\nnn\$\$**{ Wie \$\$\nnn\$\$, aber Vielfache von Grundschrift-Zeilenabständen (auch mit Grafik-Aufruf, z. B. \$\$!\nnn#iii\$\$)
- \$\$\nnn#iii\$\$**{ Wie \$\$\nnn\$\$, zusätzlich Einmontieren der Abbildung Nr. iii.
- \$\$\nnn#iii#jjj\$\$**{ Wie \$\$\nnn\$\$, zusätzlich Einmontieren der Abbildungen Nr. iii und jjj. Es können bis zu vier Abbildungsnummern angegeben werden.
- \$\$mmm/nnn\$\$**{ Aussparung für Abbildungen, die nicht über die gesamte Satzbreite gehen.  
Die Zahl mmm (1- bis 4-stellig) vor dem »/« gibt die Breite in Punkt an und darf maximal gleich der Satzspiegelbreite sein; die Zahl nnn (1- bis 4-stellig) nach dem »/« gibt die Höhe der Aussparung in Punkt an und darf die Satzspiegelhöhe nicht überschreiten. Sonst wie \$\$\nnn\$\$.
- \$\$mmm/!nnn\$\$**{ Wie \$\$mmm/nnn\$\$, aber Vielfache von Grundschrift-Zeilenabständen; auch mit Grafik-Aufruf nach #, z. B. \$\$mmm/!nnn#iii\$\$.
- \$\$mmm/nnn+\$\$**{ Wie \$\$mmm/nnn\$\$, Aussparung jedoch am rechten statt am linken Satzspiegelrand freihalten.
- \$\$mmm/nnn+-\$\$**{ Wie \$\$mmm/nnn\$\$, Aussparung jedoch in linken Spalten (= Spalten mit gerader Nummer) am rechten statt am linken Satzspiegelrand freihalten, in rechten Spalten am linken Satzspiegelrand.

**\$\$mmm/nnn-+\$\$**{ Wie \$\$mmm/nnn\$\$, Aussparung jedoch in linken Spalten (= Spalten mit gerader Nummer) am linken Satzspiegelrand freihalten, in rechten Spalten am rechten Satzspiegelrand.

**\$\$mmm/nnn+=-\$\$**{ Wie \$\$mmm/nnn\$\$, Aussparung jedoch auf linken Seiten (= Seiten mit gerader Nummer) am rechten statt am linken Satzspiegelrand freihalten, auf rechten Seiten am linken Satzspiegelrand.

**\$\$mmm/nnn=-+\$\$**{ Wie \$\$mmm/nnn\$\$, Aussparung jedoch auf linken Seiten (= Seiten mit gerader Nummer) am linken Satzspiegelrand freihalten, auf rechten Seiten am rechten Satzspiegelrand.

**\$\$mmm/nnn#iii\$\$**{ Wie \$\$mmm/nnn\$\$, zusätzlich Einmontieren der Abbildung Nr. iii.

In der Ausgabe des Satzprogramms steht der Aufruf der Abbildung horizontal am linken Satzspiegelrand, vertikal auf der Grundlinie der ersten Zeile der Aussparung.

**\$\$mmm/nnn#iii+-eee\$\$**{ Wie \$\$mmm/nnn#iii\$\$

Hinter der Abbildungsnummer kann durch Angabe von +eee, +-eee, -+eee bestimmt werden, ob die Aussparung immer oder nur in linken bzw. nur in rechten Spalten am rechten statt am linken Satzspiegelrand freigehalten werden soll. Die Angabe eee gibt dabei an, wie weit die Abbildung in den Spalten, in denen die Aussparung nach rechts geschoben wird, vom linken Satzspiegelrand eingerückt werden soll.

**\$\$mmm/nnn#iii+-eee/aaa\$\$**{ Wie \$\$mmm/nnn#iii+-eee\$\$

aaa gibt an, wie weit Grafiken vom linken Rand eingerückt werden sollen, die auf Grund der Angaben zu +-eee bzw. -+eee nicht eingerückt werden. aaa kann auch negativ sein.

**\$\$mmm/nnn#iii+=-eee\$\$**{ Wie \$\$mmm/nnn#iii+-eee\$\$

Wird vor +eee, +-eee, -+eee ein Gleichheitszeichen eingefügt (also =+eee, +=-eee, =-+eee angegeben), so werden die Einrückungen nicht spaltenbezogen, sondern seitenbezogen vorgenommen.

**\$\$mmm/nnn#iii+=-eee/aaa\$\$**{ Wie \$\$mmm/nnn#iii+=-eee\$\$

aaa gibt an, wie weit Grafiken vom linken Rand eingerückt werden sollen, die auf Grund der Angaben zu +-eee bzw. -+eee nicht eingerückt werden. aaa kann auch negativ sein.

**\$\$mmm/nnn#iii#jjj\$\$**{ Wie \$\$mmm/nnn\$\$, zusätzlich Einmontieren der Abbildungen Nr. iii und jjj. Es können bis zu vier Abbildungsnummern angegeben werden.

In der Ausgabe des Satzprogramms steht der Aufruf der Abbildung horizontal am linken Satzspiegelrand, vertikal auf der Grundlinie der ersten Zeile der Aussparung.

**\$\$mmm/nnn#iii+-eee#jjj+-eee\$\$**{ Wie \$\$mmm/nnn#iii#jjj\$\$

Hinter der ersten Abbildungsnummer kann durch Angabe von +eee, +-eee, -+eee bestimmt werden, ob die Aussparung immer oder nur in linken bzw. nur in rechten Spalten am rechten Satzspiegelrand freigehalten werden soll. Die Angabe eee gibt dabei an, wie weit die Abbildung in den Spalten, in denen die Aussparung nach rechts geschoben wird, vom linken Satzspiegelrand eingerückt werden soll. Die Angabe von +eee, +-eee, -+eee hinter der zweiten bis vierten Abbildungsnummer betrifft nur die Einrückung der zweiten Abbildung selbst, nicht die Einrückung der Aussparung.

**\$\$mmm/nnn#iii+-eee/aaa#jjj+-eee/aaa\$\$**{

Wie \$\$mmm/nnn#iii+-eee#jjj+-eee\$\$

aaa gibt an, wie weit Grafiken vom linken Rand eingerückt werden sollen, die auf Grund der Angaben zu +-eee bzw. -+eee nicht eingerückt werden. aaa kann auch negativ sein.

**\$\$mmm/nnn#iii=+-eee#jjj+-eee\$\$**{

Wie \$\$mmm/nnn#iii+-eee#jjj+-eee\$\$

Wird vor +eee, +-eee, -+eee, +-eee/aaa, -+eee/aaa ein Gleichheitszeichen eingefügt (also =+eee, =+-eee, =-+eee, =+-eee/aaa, =-+eee/aaa angegeben), so werden die Einrückungen nicht spaltenbezogen, sondern seitenbezogen vorgenommen.

**\$\$-mmm/nnn\$\$**{ Wie \$\$mmm/nnn\$\$, jedoch soll die Aussparung erst in der nächsten Spalte oben berücksichtigt werden.

**\$\$-mmm/!nnn\$\$**{ Wie \$\$-mmm/nnn\$\$ aber Vielfache von Grundschrift-Zeilenabständen; auch mit Grafik-Aufruf nach #, z. B. \$\$mmm/!nnn#iii\$\$.

**\$\$-mmm/nnn#iii\$\$**{ Wie \$\$-mmm/nnn\$\$, zusätzlich Einmontieren der Abbildung Nr. iii.

**\$\$-mmm/nnn#iii#jjj\$\$**{ Wie \$\$-mmm/nnn\$\$, zusätzlich Einmontieren der Abbildungen Nr. iii und jjj. Es können bis zu vier Abbildungsnummern angegeben werden.

**\$\$0/0#iii\$\$**{ Einmontieren der Abbildung Nr. iii ab hier in jeder Spalte.

Die Abbildung wird relativ zum Spaltenanfang positioniert. Dieser liegt auf der Schriftgrundlinie der beim Satz automatisch erzeugten Kopfzeile am linken Satzspiegelrand.

**\$\$0/0#iii=\$\$**{ Einmontieren der Abbildung Nr. iii ab hier in jeder Seite; sonst wie \$\$0/0#iii\$\$.

**\$\$0/1#iii\$\$**{ Einmontieren der Abbildung Nr. iii ab hier in jeder Spalte mit ungerader Nummer.



Die Abbildung wird relativ zum Spaltenanfang positioniert. Dieser liegt auf der Schriftgrundlinie der beim Satz automatisch erzeugten Kopfzeile am linken Satzspiegelrand.

**\$\$0/1#iii=\$\$**{ Einmontieren der Abbildung Nr. *iii* ab hier in jeder Seite mit ungerader Nummer; sonst wie **\$\$0/1#iii\$\$**{.

**\$\$0/2#iii\$\$**{ Einmontieren der Abbildung Nr. *iii* ab hier in jeder Spalte mit gerader Nummer.

Die Abbildung wird relativ zum Spaltenanfang positioniert. Dieser liegt auf der Schriftgrundlinie der beim Satz automatisch erzeugten Kopfzeile am linken Satzspiegelrand.

**\$\$0/2#iii=\$\$**{ Einmontieren der Abbildung Nr. *iii* ab hier in jeder Seite mit gerader Nummer; sonst wie **\$\$0/2#iii\$\$**{.

**\$\$0/0#iii#jjj\$\$**{, **\$\$0/1#iii#jjj\$\$**{, **\$\$0/2#iii#jjj\$\$**{  
Wie **\$\$0/0#iii\$\$**{, **\$\$0/1#iii\$\$**{, **\$\$0/2#iii\$\$**{, jedoch Einmontieren der Abbildungen Nr. *iii* und *jjj*. Es können bis zu vier Abbildungsnummern angegeben werden.

**\$\$0/0#0\$\$**{ Aufheben der Anweisungen **\$\$0/0#iii\$\$**{ und **\$\$0/0#iii#jjj\$\$**{

Ab hier wird nicht mehr in jeder Spalte eine Abbildung einmontiert.

**\$\$0/1#0\$\$**{ Aufheben der Anweisungen **\$\$0/1#iii\$\$**{ und **\$\$0/1#iii#jjj\$\$**{

Ab hier wird nicht mehr in jeder Spalte mit ungerader Nummer eine Abbildung einmontiert.

**\$\$0/2#0\$\$**{ Aufheben der Anweisungen **\$\$0/2#iii\$\$**{ und **\$\$0/2#iii#jjj\$\$**{

Ab hier wird nicht mehr in jeder Spalte mit gerader Nummer eine Abbildung einmontiert.

**\$\$#iii\$\$**{ Abbildungsaufruf: Einmontieren der Abbildung Nr. *iii* an der Stelle, an der diese Steueranweisung steht.

Die Abbildung wird vertikal relativ zur Schriftgrundlinie positioniert, auf der ein an dieser Stelle gesetztes Zeichen stehen würde.

**\$\$#iii:mm\$\$**{ Wie **\$\$#iii\$\$**{, dabei Verschieben der Abbildung um *mm*/2 Punkt nach oben bzw. (bei negativem *mm*) nach unten. *mm* kann maximal 2-stellig angegeben werden.

Die Angabe *:mm* hinter der Grafik-Nummer *iii* ist bei allen unter 7.3 genannten Anweisungen möglich, mit denen Grafiken eingebunden werden, außer bei **\$\$0/0#iii\$\$**{ bis **\$\$0/2#iii=\$\$**{.

**\$\$#iii#jjj\$\$**{ Einmontieren der Abbildungen Nr. *iii* und *jjj* an der Stelle, an der diese Steueranweisung steht.

Die Abbildungen werden vertikal relativ zur Schriftgrundlinie positioniert, auf der ein an dieser Stelle gesetztes Zeichen stehen würde. Vgl. auch &! (#Gi*iii*), Kapitel 12.9.

**\$\$-s\$\$ {**

Freiraum für *s* ganzseitige Abbildungen vormerken.

*s* ist eine einstellige Zahl zwischen 1 und 9. Es können also maximal 9 Seiten am Stück vorgemerkt werden. Ist *s* eine gerade Zahl, so beginnt der Freiraum erst bei der nächsten Seite mit einer geraden Seitennummer (für Abbildungen, die im aufgeschlagenen Buch auf gegenüberliegenden Seiten stehen sollen; dies gilt auch bei mehrspaltigem Satz).

**\$\$0\$\$ {**

Löschen der Vormerkungen für die noch nicht berücksichtigten Freiräume

**\$\$-0\$\$ {**

Wie **\$\$0\$\$ {**

#### 7.4. Vertikaler Tabulator; Ändern von Spaltenhöhe und Durchschuss

Vor und nach den unter 7.4 aufgeführten und mit  $\$$  beginnenden Anweisungen muss ein Leerzeichen oder ein Zeilenwechsel stehen.

**\$\$+nnn\$\$ {**

Zeilenwechsel, vertikaler Tabulator auf *nnn* Punkt vom oberen Satzspiegelrand.

Wenn der Satzspiegel schon bis zur angegebenen vertikalen Position (Angabe *nnn* in Punkt, 1–3 Ziffern, gemessen vom oberen Satzspiegelrand) oder darüber hinaus gefüllt ist, werden nur die evtl. davor vorhandenen variablen Vorschübe zu festen Vorschüben. Sonst wird die Schriftgrundlinie der nachfolgenden Zeile auf die angegebene Position festgelegt. Unmittelbar nachfolgende Vorschubanweisungen werden wie am Spaltenanfang behandelt.

**\$\$+!nnn\$\$ {**

Wie **\$\$+nnn\$\$ {**, aber Positionieren auf die Grundlinie der *nnn*-ten Zeile im Grundschrift-Zeilenraster.

**\$\$+-nnn\$\$ {**

Zeilenwechsel, vertikaler Tabulator auf Position *nnn* für die Spalte, in der diese Anweisung steht, und für die nachfolgende Spalte.

Wie **\$\$+nnn\$\$ {**, jedoch wird auch auf der nächsten Seite (bei mehrspaltigem Satz: in allen nachfolgenden Spalten der Seite, zu der die nachfolgende Spalte gehört) auf die gleiche vertikale Position positioniert, bevor dort die erste Zeile gesetzt wird.

**\$\$+-!nnn\$\$ {**

Zeilenwechsel, vertikaler Tabulator

Wie **\$\$+-nnn\$\$ {**, aber Positionieren auf die Grundlinie der *nnn*-ten Zeile im Grundschrift-Zeilenraster.

**\$\$+-0\$\$ {**

Zeilenwechsel, vertikaler Tabulator für die nachfolgende Spalte.

Nach dem Wechsel zur nächsten Seite bzw. Spalte wird dort auf die mit dieser Anweisung erreichte vertikale Position positioniert, be-

vor die erste Zeile gesetzt wird. Bei mehrspaltigem Satz gilt dies für alle Spalten der Seite, auf der die nachfolgende Spalte zu stehen kommt.

**\$\$+0+n\$\$ {** Zeilenwechsel, vertikaler Tabulator für die nachfolgende Spalte.  
Wie **\$\$+0\$\$ {**, jedoch wird auf der nächsten Seite bzw. Spalte auf eine vertikale Position positioniert, die um  $n$  Punkt tiefer liegt als die mit dieser Anweisung erreichte vertikale Position.

**\$\$+0-n\$\$ {** Zeilenwechsel, vertikaler Tabulator für die nachfolgende Spalte.  
Wie **\$\$+0\$\$ {**, jedoch wird auf der nächsten Seite bzw. Spalte auf eine vertikale Position positioniert, die um  $n$  Punkt höher liegt als die mit dieser Anweisung erreichte vertikale Position.

**\$\$+m:nnn\$\$ {** Zeilenwechsel, vertikaler Tabulator, ggf. nach Spaltenwechsel.  
Wenn mit dieser Anweisung bereits die Spalte  $m$  erreicht ist, wirkt diese Anweisung wie **\$\$+nnn\$\$ {** (d. h.: Zeilenwechsel und vertikaler Tabulator auf die hinter dem »:« angegebene Position  $nnn$ ). Andernfalls wird die Spalte abgeschlossen, ein Spaltenwechsel zur Spalte Nummer  $m$  ausgeführt und dort auf die vertikale Position  $nnn$  Punkt vom oberen Rand positioniert.

In den Folgespalten, auf die sich der vertikale Tabulator bezieht, ist automatisches Vermeiden von Hurenkindern derzeit nicht vorgesehen.

**\$\$+m: !nnn\$\$ {** Zeilenwechsel, vertikaler Tabulator, ggf. nach Spaltenwechsel.  
Wie **\$\$+m:nnn\$\$ {**, aber Positionieren auf die Grundlinie der  $nnn$ -ten Zeile im Grundschrift-Zeilenraster.

**\$\$/nnn\$\$ {** Verringern der Spaltenhöhe um  $nnn$  Punkt für die Spalte, in der diese Anweisung steht.  
 $nnn$  ist eine 1- bis 4-stellige Zahl, die kleiner sein muss als die Satzspiegelhöhe. Mit dieser Anweisung ist kein Zeilenwechsel verbunden.

Ist die Spalte schon weiter gefüllt, so wird ein Fehlerkommentar ausgegeben und mit dem nächsten Zeilenwechsel eine neue Spalte begonnen.

**\$\$/ !nnn\$\$ {** Verringern der Spaltenhöhe um  $nnn$  Grundschrift-Zeilen für die Spalte, in der diese Anweisung steht. Sonst wie **\$\$/nnn\$\$ {**.

**\$\$/-nnn\$\$ {** Verringern der Spaltenhöhe für die Spalte, in der diese Anweisung steht, und für die nachfolgende Spalte.

Wie **\$\$/nnn\$\$ {**, jedoch wird auch die Höhe der nachfolgenden Spalte (bei mehrspaltigem Satz: aller Spalten der Seite, auf der die nachfolgende Spalte zu stehen kommt) um  $nnn$  Punkt verringert.

- \$\$/-!nnn\$\$ {** Verringern der Spaltenhöhe für die Spalte, in der diese Anweisung steht, und für die nachfolgende Spalte.  
Wie \$\$/-nnn\$\$ {, Verringerung jedoch um nnn Grundschrift-Zeilenabstände.
- \$\$/nnn\$\$ {** Verringern der Spaltenhöhe um nnn Punkt für die Spalte, in der diese Anweisung steht.  
Wie \$\$/nnn\$\$ {, jedoch wird eine bereits verringerte Spaltenhöhe nicht weiter verringert, sondern um den größeren der zuvor und aktuell angegebenen Werte.
- \$\$//!nnn\$\$ {** Verringern der Spaltenhöhe für die Spalte, in der diese Anweisung steht, und für die nachfolgende Spalte.  
Wie \$\$//-nnn\$\$ {, Verringerung jedoch um nnn Grundschrift-Zeilenabstände.
- \$\$/+nnn\$\$ {** Vergrößern der Spaltenhöhe um nnn Punkt für die Spalte, in der diese Anweisung steht.  
nnn ist eine 1- bis 4-stellige Zahl. Mit dieser Anweisung ist kein Zeilenwechsel verbunden.
- \$\$/+!nnn\$\$ {** Vergrößern der Spaltenhöhe um nnn Grundschrift-Zeilenabstände für die Spalte, in der diese Anweisung steht.  
nnn ist eine 1- bis 4-stellige Zahl. Mit dieser Anweisung ist kein Zeilenwechsel verbunden.
- \$\$/=nnn\$\$ {** Verändern der Spaltenhöhe auf nnn Punkt für den Rest des Textes.  
nnn ist eine 1- bis 4-stellige Zahl. Mit dieser Anweisung ist kein Zeilenwechsel verbunden.
- &!D-n** Verringerter Durchschuss  
Der Vorschub nach der Zeile, in der diese Anweisung steht, wird um n Punkt verringert. n ist eine einstellige Zahl zwischen 1 und 9.
- &!D-(nn)** Wie &!D-n, aber zweistellige Angabe.
- &!D/n** Verringerter Durchschuss  
Der Vorschub nach der Zeile, in der diese Anweisung steht, wird um n/8 Punkt verringert. n ist eine einstellige Zahl zwischen 1 und 9.  
Die damit erzeugte Verschiebung der Schriftgrundlinie muss bis zum nächsten Spaltenwechsel mit &!D/0 wieder aufgehoben worden sein.
- &!D/(nn)** Verringerter Durchschuss  
Wie &!D/n, jedoch auch mehrstellige Angabe möglich.

- &!D\n** Verringerter Durchschuss  
 Der Vorschub wird nach jeder Zeile, in der diese Anweisung steht, um  $n/8$  Punkt verringert, im Unterschied zur Anweisung  $\&!D/n$ , die den Schreibstrahl für den Rest der Seite um  $n/8$  Punkt verschiebt, falls er nicht schon um diesen oder einen höheren Betrag verschoben ist.
- &!D\ (nn)** Verringerter Durchschuss  
 Wie  $\&!D\n$ , jedoch auch mehrstellige Angabe möglich.
- &!D\ (nn!)** Verringerter Durchschuss  
 Durchschuss nach dieser und nach allen folgenden Zeilen bis vor die Anweisung  $\&!D/0$  um  $nn/8$  Punkt verringern.
- &!D=n** Zusätzlicher Durchschuss von  $n/8$  Punkt nach der Zeile, in der diese Anweisung steht.  
 Die damit erzeugte Verschiebung der Schriftgrundlinie muss bis zum nächsten Spaltenwechsel mit  $\&!D/0$  wieder aufgehoben worden sein.
- &!D= (nn)** Zusätzlicher Durchschuss von  $n/8$  Punkt nach der Zeile, in der diese Anweisung steht.  
 Wie  $\&!D=n$ , für mehr als einstellige Zahlen.
- &!D= (nn!)** Zusätzlicher Durchschuss  
 Durchschuss nach dieser und nach allen folgenden Zeilen bis vor die Anweisung  $\&!D/0$  um  $nn/8$  Punkt erhöhen.  
 Der in Achtelpunkt-Schritten veränderte Durchschuss wird bei der Berechnung der Seitenhöhe nicht mit berücksichtigt, jedoch vor der Einsteuerung der Fußnoten bzw. vor der unten stehenden Seitenzahl wieder ausgeglichen.
- &!Dnn** Zusätzlicher Durchschuss von  $nn$  Punkt.  
 Nach der Zeile, in der diese Anweisung steht, wird ein zusätzlicher Vorschub von  $nn$  Punkt eingefügt.  $nn$  ist ein Wert zwischen 01 und 87 und muss zweistellig angegeben werden. Der Vorschub wird auch am Seiten- bzw. Spaltenende berücksichtigt. Wird  $\&!Dnn$  mehrmals in einer Zeile benutzt, so wird nur der größte der angegebenen Werte berücksichtigt.
- &!D99** Registerhaltigkeit herstellen durch Verschieben nach unten.  
 Steht die Zeile, die diese Steueranweisung enthält, nicht auf einer Position, die einem ganzzahligen Vielfachen des Grundschrift-Zeilenabstands entspricht, wird sie so weit nach unten verschoben, dass sie auf der nächstfolgenden derartigen Position steht.

- &!D88** Registerhaltigkeit herstellen durch Verschieben nach unten oder oben.  
Steht die Zeile, die diese Steueranweisung enthält, nicht auf einer Position, die einem ganzzahligen Vielfachen des Grundschrift-Zeilenabstands entspricht, wird sie so weit nach unten oder oben verschoben, dass sie auf der nächstgelegenen derartigen Position steht.
- &!D00** Registerhaltigkeit herstellen durch Verschieben nach oben.  
Steht die Zeile, die diese Steueranweisung enthält, nicht auf einer Position, die einem ganzzahligen Vielfachen des Grundschrift-Zeilenabstands entspricht, wird sie so weit nach oben verschoben, dass sie auf der letzten vorhergehenden derartigen Position steht.
- &!D!!** Vertikale Position fixieren  
Kein Vorschub; die vorangehenden Zeilenvorschübe dieser Spalte werden beim Spaltenhöhenausgleich nicht verändert.  
Positionierungen, die durch eine der Anweisungen &!D99, &!D88, &!D00 oder &!D!! vorgenommen werden, werden durch den Spaltenhöhenausgleich nicht verändert. Die Abstände der Zeilen, die nach der letzten so positionierten Zeile auf der Seite bzw. Spalte stehen, sind jedoch für Spaltenhöhenausgleich nach wie vor veränderbar. Um registerhaltigen Satz zu erhalten, muss also beim Aufruf des Satzprogramms zusätzlich die Veränderung der Spaltenhöhe durch Verringerung oder Vergrößerung des Durchschusses ausgeschaltet werden. Dies erreicht man mit Hilfe des Parameters HOE, mit dessen zweiter Angabe NAUSGL Angaben zum Spaltenhöhenausgleich gemacht werden. Addiert man auf den Betrag der dort anzugebenden Zahl den Wert 100 auf, so schaltet man damit die Veränderung des Durchschusses zum Spaltenhöhenausgleich aus.
- &!D?0** Variablen Vorschub fixieren  
Wird diese Anweisung unmittelbar hinter einer Zeilenwechselanweisung mit variablem Vorschub (z. B. Überschriftsanfang oder -ende) gegeben, so wird der mit diesem Zeilenwechsel verbundene variable Vorschub zu einem festen Vorschub.
- &!D?n** Festen Vorschub variabel machen  
Wird diese Anweisung unmittelbar hinter einer Zeilenwechselanweisung mit festem Vorschub (z. B. \$\$\$=\$\$\$) gegeben, so erhält der mit diesem Zeilenwechsel verbundene feste Vorschub einen variablen Anteil von n Punkt.  
Variable Vorschübe werden bei Bedarf vorrangig zum Spaltenhöhenausgleich herangezogen. Dabei werden sie um maximal 1/3 ihres Betrags verringert bzw. um maximal 1/2 ihres Betrags vergrößert.

## II. Gestaltung der Zeilen

### 8. Zeilenumbruch; Spatien

Vor allen unter 8. genannten Anweisungen außer `&!V` `&!X` `&!O` `&!I` muss ein Leerzeichen oder Zeilenwechsel stehen. Hinter `$$$` und `$$$$` stehende Leerzeichen werden ignoriert.

Für Ausgangszeilen (Zeile vor einem mit einer Steueranweisung verlangten Zeilenwechsel) kann über den Parameter `AUS` ein rechter Einzug von mindestens 10 Punkt verlangt werden. Wenn jedoch `@-(nnn)` bzw. `@-n` in der Zeile vorkam, unterbleibt der rechte Einzug in jedem Fall.

**\$\$\$** Neue Zeile stumpf, keine Abschnittsgrenze

Solche Zeilen sind als letzte Zeilen einer Seite bzw. Spalte zugelassen, die Zeile davor wird auch als erste auf einer neuen Seite bzw. Spalte zugelassen.

Innerhalb von Einschaltungen wird der Text hinter `$$$` so weit eingerückt wie die Einschaltung; vgl. Kapitel 4.

Frei stehendes `$$$` wird ignoriert, wenn unmittelbar darauf eine weitere Vorschubanweisung folgt. Aufeinander folgende Zeilenwechselanweisungen, zwischen denen nichts gesetzt wurde, erzeugen keine Leerzeilen.

**\$\$\$\$** Wie `$$$`, jedoch Zeile davor auf Satzbreite austreiben; Einrückungen und Zentrierungen werden nicht beendet.

**\$\$\$\\\$\$\$\$** Kein Zeilenwechsel

`$$$\\$$$$` unterdrückt eine unmittelbar darauf folgende Zeilenwechselanweisung wie `$`, `$$$`, `$$$=$$$$`, `$$$-9$$$$=$$$$`

**&!V** »volle Zeile«: Warten mit Zeilenwechsel

Die Zeile, in der diese Anweisung steht, wird erst bei der nächsten Anweisung für Zeilenwechsel oder hinter der nächsten Anweisung `&!X` abgebrochen; der Satzspiegel kann dabei auch überschritten werden.

Steht `&!V` nicht zwischen Leerzeichen, sondern in einem Wort, so wird ggf. Silbentrennung in diesem Wort nur bis vor dieser Stelle versucht. Wird dort keine Trennstelle gefunden, so wird das ganze Wort noch in die »volle Zeile« übernommen.

**&!X** Aufheben der Anweisung `&!V`

**&!Z (nnn)** bedingter Zeilenwechsel

Wenn in der Zeile, in der diese Anweisung steht, nicht noch mindestens `nnn` Punkt (1–3 Stellen) frei sind, nachdem ggf. die davor stehenden Spatien maximal um den mit dem Parameter `AUS` an-

gegebenen Wert `ISPNEG` verringert wurden, wird eine neue Zeile begonnen. Die Zeile vor dieser Anweisung wird in diesem Fall nicht ausgetrieben, es sei denn, dass eine zuvor gegebene Anweisung `&!R` bzw. `@-A` noch wirksam ist.

**&!Z (m/n)** Wie `&!Z (nnn)`, jedoch steht in Klammern statt einer festen Punktzahl ein Bruchteil der Satzbreite (z.B: `&!Z (2/5)`)

**&!Z- (nnn)** und

**&!Z- (m/n)** Wie `&!Z (nnn)` bzw. `&!Z (m/n)`, aber Austreiben der durch diese Anweisung evtl. abgebrochenen Zeile

**\_** Fester Ausschluss, Viertelgeviert (vgl. auch 12.7.)

Frei stehender fester Ausschluss (`>_<` zwischen Leerzeichen) wird wie ein normales Spatium behandelt (u. a. also auf normale Spatienbreite ausgedehnt). Kommt ein frei stehender fester Ausschluss beim Zeilenumbruch ans Zeilenende oder an den Zeilenanfang, so wird er zusammen mit dem davor bzw. dahinter stehenden Leerzeichen unterdrückt.

**@+\_\_** Folge von festen Ausschlüssen

Soll eine (beliebig lange) frei stehende Folge von festen Ausschlüssen am Zeilenende bzw. Zeilenanfang wegfallen, so ist `@+` davor zu schreiben (siehe unten).

**@+&! (+nn)** Fester Zwischenraum

Soll ein mit `&!(+nn)` codierter frei stehender Zwischenraum am Zeilenende bzw. Zeilenanfang wegfallen, so ist `@+` davor zu schreiben.

**&!F (nnn)** fester Ausschluss von `nnn` Punkt

Zusammenhängender größerer Zwischenraum von `nnn` Punkt (1- bis 4-stellig, maximal Satzspiegelbreite) in der Zeile. Der Zwischenraum wird zwischen Spatien gesetzt. Er bleibt zusammen in einer Zeile; wenn er in die aktuelle Zeile nicht passt, wird eine neue Zeile begonnen; die aktuelle Zeile wird wie eine normale Zeile behandelt.

**&.ab** Doppelt breites Spatium

Doppelt breite Spatien, die an der Zeilengrenze wie normale Spatien behandelt werden (dort also wegfallen), können durch zwischen Leerzeichen stehende leere Makros (vgl. Kapitel 19) erzeugt werden.

Ein ähnliches Ergebnis kann erzielt werden, indem die Steueranweisungen `\` bzw. `\\` oder `^`, die innerhalb eines Wortes für die Steuerung der Silbentrennung vorgesehen sind, zwischen Leerzeichen geschrieben werden. Kommt `\` bzw. `\\` oder `^` dabei jedoch ans Zeilenende oder den Zeilenanfang, so bleibt das davor stehende bzw. dahinter stehende Spatium erhalten.



- &!S (+n)** Spatienbreite vergrößern  
Für den Rest des Abschnitts wird die Spatienbreite um +n Bildlinien gegenüber der voreingestellten bzw. mit DIC angegebenen Spatienbreite verändert.
- &!S (-n)** Spatienbreite verringern  
Für den Rest des Abschnitts wird die Spatienbreite um -n Bildlinien gegenüber der voreingestellten bzw. mit DIC angegebenen Spatienbreite verändert.
- &!S (+0)** Veränderung der Spatienbreite aufheben  
Mit dieser Anweisung wird die voreingestellte bzw. mit DIC angegebene Spatienbreite wieder eingestellt.
- @+** Vorgehendes und/oder nachfolgendes Spatium unterdrücken  
Steht @+ am Wortanfang, so wird dieses Wort ohne Spatium an das vorhergehende Wort angehängt. Dies ermöglicht es, in zusammenhängenden Zeichenfolgen Stellen zu kennzeichnen, an denen Zeilenwechsel (ohne Divis) vorgenommen werden darf; wird die Zeichenfolge noch in der gleichen Zeile begonnen, in der das vorhergehende Wort endet, so erscheint an dieser Stelle kein Spatium.  
Steht @+ am Wortende oder zwischen Blanks, so wird dieses Wort ohne Spatium mit dem nachfolgenden Wort verbunden, wenn an dieser Stelle kein Zeilenwechsel vorgenommen wird.  
Steht vor @+ ein freistehender fester Ausschluss, so fällt dieser (einschließlich der umgebenden Spatien) ebenfalls weg.
- &!O** Belichtung aus  
Alle nach dieser Anweisung stehenden Zeichen werden nicht belichtet, aber für Zeilen- und Seitenaufbau berücksichtigt. Die Belichtung wird durch &!I wieder eingeschaltet. Ist nach &!O bis zum nächsten Abschnittsende kein &!I gegeben, so wird es dort ergänzt.
- &!I** Belichtung ein  
Hebt vorhergehendes &!O wieder auf.
- &!U** unveränderte Zeileneinteilung  
Ab dieser Anweisung wird der Zeilenumbruch der QUELL-Datei übernommen. Aufeinander folgende Zeilen mit gleicher Zeilennummer gelten dabei als eine Zeile.  
Die Anweisung &!U muss zwischen Leerzeichen stehen. Sie wird durch die Anweisung &!N wieder aufgehoben.
- &!N** normaler Zeilenumbruch  
Ab dieser Anweisung wird der Zeilenumbruch wieder normal durchgeführt; der Zeilenumbruch der QUELL-Datei bleibt wieder unberücksichtigt.

Die Steueranweisung &!N muss zwischen Leerzeichen stehen.

**&!B-**

Schaltet von Blocksatz auf Rausatz

**&!B+**

Schaltet Blocksatz (wieder) ein

Die im Parameter AUS zum Wert IFLATT angegebenen Werte gelten als Blocksatz-Bedingungen für den Satz, bis sie mit &!B- ausgeschaltet werden. Zwischen &!B- und &!B+ wird unabhängig von den im Parameter AUS angegebenen Bedingungen nicht ausgetrieben. Bei IFLATT=-1 sind &!B- und &!B+ wirkungslos.

## 9. Silbentrennung

Der automatischen Silbentrennung des Satzprogramms liegen die Regeln für die deutsche Sprache zugrunde. Sollen anderssprachige Texte gesetzt werden, so empfiehlt es sich, falls nicht (wie häufig für das Englische bei normal breitem Satzspiegel) auf Trennung ganz verzichtet werden kann, schon vor dem Satz den Text mit Kann-Trennstellen zu versehen. Als Hilfsmittel stehen hierzu die Standard-Makros \*SILMARKO und \*SILMARKE zur Verfügung.

Zeichenfolgen, die »-« enthalten, werden bei Bedarf nach dem »-« getrennt, außer wenn das »-« Bestandteil einer Steueranweisung ist. Wenn das letzte Zeichen vor dem »-« eine Ziffer oder das Zeichen »\« ist, so wird hier nur getrennt, wenn es über den ersten Zahlenwert zum Parameter SIL ausdrücklich verlangt wird. Das »-« wird dann nach den gleichen Regeln Divis oder Halbgeviertstrich, die für dieses Zeichen vor Spatium gelten würden. Siehe auch unten bei 12.7 (»-«).

Enthält ein zu trennendes Wort (genauer: die an das Silbentrennprogramm übergebene Zeichenfolge zwischen zwei Leerzeichen) mehr Sonderzeichen als Buchstaben, so wird dieses Wort nicht getrennt. Um Wörter, an deren Anfang viele Sonderzeichen oder umfangreiche Kombinationen von Steueranweisungen stehen, dennoch trennen zu können, kann vor den ersten Buchstaben des Wortes die Steueranweisung für Trennverbot »\« geschrieben werden. Dies bewirkt, dass die davor stehenden Sonderzeichen für die Entscheidung über die Trennung nicht mitgezählt werden.

Steht »-« in den Eingabedaten als letztes (nicht allein stehendes) Zeichen in der Zeile, so wird es als Silbentrennzeichen interpretiert. Diese Silbentrennung wird rückgängig gemacht; sie wird jedoch ggf. zur Trennung eines Wortes vorrangig vor der automatischen Silbentrennung herangezogen, wenn nicht über den Parameter SIL etwas anderes verlangt ist. Deshalb sollten nur korrekte Silbentrennungen in der QUELL-Datei vorkommen.

Bei der Datenerfassung ist darauf zu achten, dass das »-« am Zeilenende in der QUELL-Datei wirklich nur als Silbentrennzeichen vorkommt:

- Wurden Wörter wie »Satz-Verfahren« in der QUELL-datei nach »Satz-« getrennt, so sollte die folgende Zeile mit »-Verfahren« beginnen, damit ein »-« auch dann erhalten bleibt, wenn das ganze Wort in die Zeile übernommen wird. Beim Zusammenfassen des getrennten Wortes fällt das »-« hinter »Satz« weg.
- Bei Bildungen wie »Satz- und Druckverfahren« würde, wenn die Zeile mit »Satz-« endet, das »und« am Anfang der nächsten Zeile mit dem am Ende der vorhergehenden Zeile stehenden Wort zu »Satzund« verschmelzen. Ein »\« bzw. »\« oder »^« hinter dem »-« verhindert, dass das »-« ans Zeilenende kommt und damit als Silbentrennzeichen interpretiert wird.

Ist das »-« am Zeilenende Bestandteil eines Makros (z. B. & . --) oder einer Steueranweisung (z. B. #F-), so wird es nicht als Silbentrennzeichen interpretiert.

Trennung ohne Divis: siehe die Anweisung »@+« (Kapitel 8).

\ am Wortanfang: Trennverbot für dieses Wort außer an den im Wort mit »\« gekennzeichneten Stellen.

\ \ im gleichen Wort (siehe unten) hebt die Wirkung eines am Wortanfang stehenden \ auf.

- \** im Wort: Kann-Trennstelle für Silbentrennung
- Mit dieser Anweisung können Stellen markiert werden, an denen bei der Silbentrennung ein Wort bevorzugt getrennt werden soll, um falsche oder unschöne Silbentrennungen zu verhindern.
- Dazu wird von der Soll-Trennstelle aus (das ist die Stelle, bis zu der das Wort einschl. Divis noch in die Zeile passt, ohne dass die Spatien verringert werden müssen) drei Zeichen weit nach links und nach rechts geprüft, ob dort »\« steht; wenn ja, wird dort getrennt.
- Enthält ein Wort, das in der QUELL-Datei getrennt ist, eine durch »\« markierte Kann-Trennstelle, die nicht weiter als drei Zeichen von der ursprünglichen Trennstelle entfernt ist, so wird das Wort an der ursprüngliche Trennstelle keinesfalls getrennt.
- \\** vor dem ersten Buchstaben des Wortes: Wortanfang für die automatische Silbentrennung
- Damit kann verhindert werden, dass Wörter mit vielen Sonderzeichen am Wortanfang nicht getrennt werden (siehe oben).
- \\** im Wort: punktueller Trennverbot
- Die Zeichenfolge »\\« verhindert Silbentrennung an dieser Stelle.
- ^^** im Wort: punktueller Trennverbot
- Das Zeichen »^^« (als ^^ eingegeben) verhindert Silbentrennung an dieser Stelle.
- Folgen im Text drei gleiche Buchstaben aufeinander, und steht »^« (als ^^ eingegeben) oder »^^« (als ^^ eingegeben) zwischen dem ersten und dem zweiten sowie »\« zwischen dem zweiten und dem dritten dieser drei gleichen Buchstaben, so wird, falls diese Buchstabenfolge nicht (bei dem Zeichen »\«) getrennt wird, der hinter dem »\« stehende Buchstabe nicht gesetzt. Beispiel: das Wort »Schiffahrt« (und »Schiffahrt«; diese Form ist zu wählen, wenn im benutzten Font keine ff-Ligatur vorhanden ist) wird, wenn es nicht getrennt wird, zu »Schiffahrt«; wenn es getrennt wird, wird es zu »Schiffahrt«.
- &!S-** Silbentrennung ausschalten
- Danach wird nur an den durch »\« markierten Stellen getrennt.
- Steht diese Anweisung nach einer Anweisung für Einschaltung-Ende oder Überschrift-Ende, so muss diese Anweisung zwischen Leerzeichen stehen.
- Die Wirkung dieser Anweisung wird durch &!N oder &!S+ wieder aufgehoben.
- &!S+** Silbentrennung einschalten
- Durch &!S- oder &!T ausgeschaltete automatische Silbentrennung wird wieder eingeschaltet.

Steht diese Anweisung nach einer Anweisung für Einschaltung-Ende oder Überschrift-Ende, so muss diese Anweisung zwischen Leerzeichen stehen.

**&!T**

wie &amp;!S-

Steht diese Anweisung nach einer Anweisung für Einschaltung-Ende oder Überschrift-Ende, so muss diese Anweisung zwischen Leerzeichen stehen.

**&!SV**

Trennregeln vor der Rechtschreibreform

Unabhängig von der Angabe in Parameter RSR werden ab hier die Trennregeln angewandt, die vor der Rechtschreibreform von 1996 galten.

**&!SN**

Trennregeln nach der Rechtschreibreform

Unabhängig von der Angabe in Parameter RSR werden ab hier die Trennregeln angewandt, die nach der Rechtschreibreform von 1996 gelten.

**&!SV(itrenw, itrnv, itrnh)** und**&!SN(itrenw, itrnv, itrnh)** Trennregeln weiter spezifizieren

Wie &!SV bzw. &!SN; gleichzeitig werden die Mindestlänge *itrenw*, die ein Wort für die Trennung haben muss, sowie die Mindestzahl der Buchstaben vor (*itrnv*) bzw. nach (*itrnh*) dem Divis gegenüber den Angaben in Parameter SIL bzw. gegenüber der Voreinstellung geändert.

**&!S:n**

Umstellen der Zahl der hintereinander erlaubten Trennungen.

Die Zahl der aufeinander folgenden Zeilen, an deren Ende ein Wort getrennt werden darf, wird auf den angegebenen Wert *n* eingestellt. *n* ist eine einstellige Zahl. Negatives *n* unterdrückt die Silbentrennung, sobald dieser Wert überschritten wird.

**&!S::**

Rückstellen der Zahl der hintereinander erlaubten Trennungen.

Die Zahl der aufeinander folgenden Zeilen, an deren Ende ein Wort getrennt werden darf, wird auf den im Parameter SIL angegebenen Wert zurückgestellt.

## 10. Einrücken; Zentrieren; Tabellen; Marginalien

Einrückungen werden jeweils vom linken Satzspiegelrand an vorgenommen, in linksläufigen Textteilen (z. B. in hebräischen Einschüben, wenn mit Parameter HBU angegeben ist, dass hebräische und arabische Teile automatisch umgedreht werden sollen) jedoch vom rechten Satzspiegelrand aus.

Bei Zentrierungen im linksläufigen Teil ist statt »links« jeweils »rechts« zu lesen und umgekehrt. Hebräischer Rausatz (Flattersatz) wird mit #H+@-N oder @-N#H+ codiert; linksbündiger Satz im Hebräischen mit #H+@-0 (Reihenfolge beachten; @-0#H+ würde den hebräischen Teil nach rechts schieben).

Ob die letzte Zeile eines hebräischen oder arabischen Textteils rechtsbündig oder linksbündig gesetzt wird, hängt davon ab, ob diese Zeile innerhalb des linksläufigen Teils (z. B. durch @1 oder &!Z(1/1)) abgeschlossen wird oder nicht.

### 10.1. Einzüge, Einrückungen

#### 10.1.1. Einzug ohne Rücksicht auf zuvor erreichte Position

Mit den unter 10.1 genannten Anweisungen ist Mehrfachbelichtung in der gleichen Zeile möglich. Im Satzprotokoll erscheint in diesem Fall ein entsprechender Hinweis. Ein Spatium nach diesen Anweisungen bleibt erhalten.

**@-n**  $n \times 10$  Punkt Einzug in gleicher Zeile vom Satzspiegelrand an; Zahlenangabe  $n$  einstellig von 1 bis 9

**@-(nnn)** Wie @-n, jedoch Einzug in Punkt

Die Zahl nnn in der Klammer bedeutet typographische Punkt. nnn ist eine 1- bis 4-stellige Zahl, die kleiner sein muss als die Satzspiegelbreite.

Bei der Steueranweisung @-(0) wird im Satzprotokoll kein Hinweis auf mögliche Doppelbelichtung gegeben.

**@-(m/n)** Wie @-n; der Bruch in der Klammer bedeutet Bruchteile der Satzbreite

#### 10.1.2. Einzug unter Berücksichtigung der erreichten Position

**&=nnn** Einrücken des Restes der Zeile und der Folgezeilen um nnn Punkt

Der Rest der Zeile und die Folgezeilen bis zur nächsten Anweisung für Zeilenwechsel werden um nnn Punkt eingerückt. Die Angabe nnn muss dreistellig erfolgen und kleiner sein als die Satzspiegelbreite. Wenn das vor der Anweisung Stehende im angegebenen Raum keinen Platz hat, wird an der erreichten Stelle fortgefahren; keine Doppelbelichtung.

Steht  $\&=nnn$  (und die übrigen mit  $\&=$  beginnenden Anweisungen) zwischen Leerzeichen, so gilt das Leerzeichen nach  $\&=nnn$  als nicht vorhanden.

Die mit diesen Anweisungen angegebene Position ist gleichzeitig Endpunkt für eine noch nicht abgeschlossene Rechts- oder Mitte-Zentrierung.

- $\&= (nnn)$**  Wie  $\&=nnn$ ;  $nnn$  kann eine 1- bis 4-stellige Zahl sein
- $\&= (\$)$**  Wie  $\&=nnn$ , aber (nur innerhalb von Grundtext:) Einrückung auf den in den Parametern für  $\$$  angegebenen Wert.
- $\&= (\$\$)$**  Wie  $\&=nnn$ , aber (nur innerhalb von Einschaltungen:) Einrückung auf den in den Parametern für  $\$\$$  angegebenen Wert.
- $\&=999$**  Einrückung der Folgezeilen auf die vor der Anweisung erreichte Satzbreite
- Die in der selben Zeile vor  $\&=999$  stehende Spalten werden nicht zum Zeilenausgleich herangezogen.
- $\&= (999)$**  Wie  $\&=999$
- $\&= (m/n)$**  Wie  $\&=nnn$ ; der Bruch in der Klammer bedeutet Bruchteile der Satzbreite
- $\&= (Pm+)$**  Wie  $\&= (nnn)$ , jedoch Einrückung auf die mit  $\&!M_m$  gemerkten Position, falls diese weiter rechts liegt als die aktuell erreichte Position; andernfalls wird  $m$  so lange um 1 erhöht, bis eine solche Position gefunden wird. Wird dabei eine undefinierte Merkstelle erreicht oder liegt diese Merkstelle weiter links als die zuletzt geprüfte Merkstelle, so wird mit einer Fehlermeldung abgebrochen und die aktuelle Position beibehalten.
- $\&= (Pm+nnn)$**  Wie  $\&= (nnn)$ , jedoch Einrückung auf die Position, die  $nnn$  Punkt rechts von der mit  $\&!M_m$  gemerkten Position liegt. Die Angabe  $+000$  kann weggelassen werden.
- $\&= (Pm-nnn)$**  Wie  $\&= (nnn)$ , jedoch Einrückung auf die Position, die  $nnn$  Punkt links von der mit  $\&!M_m$  gemerkten Position liegt. Die Angabe  $-000$  kann weggelassen werden.
- $\&=-nnn$**  Einrücken nur der Folgezeilen (bis zur nächsten Anweisung für Zeilenwechsel) um  $nnn$  Punkt.
- Die Angabe  $nnn$  muss dreistellig erfolgen und kleiner sein als die Satzspiegelbreite. Die Anweisung kann an beliebiger Stelle in der Zeile stehen.
- Wenn  $\&=nnn$  und  $\&=-nnn$  kombiniert werden sollen, muss die Reihenfolge ihrer Wirkung berücksichtigt werden. Beispiel:  
 $\$\$\$\&=040\&=-010\text{Text} \dots$  bedeutet: Beginn einer neuen Zeile, Textbeginn 40 Punkt vom Satzspiegelrand, in Folgezeilen 10 Punkt vom Satzspiegelrand.

$\$ \$ \$ \&=040 \&=010$ Text . . . hat die gleiche Wirkung.  
 $\$ \$ \$ \&=-010 \&=040$ Text . . . würde jedoch auch die Folgezeilen um 40 Punkt einziehen, da die Wirkung von  $\&=-010$  durch  $\&=040$  aufgehoben würde.

- $\&=- (nnn)$**  Wie  $\&=-nnn$ ;  $nnn$  ist eine 1- bis 4-stellige Zahl
- $\&=- (\$)$**  Wie  $\&=-nnn$ , aber (nur innerhalb von Grundtext:) Einrückung auf den in den Parametern für  $\$$  angegebenen Wert.
- $\&=- (\$ \$)$**  Wie  $\&=-nnn$ , aber (nur innerhalb von Einschaltungen:) Einrückung auf den in den Parametern für  $\$ \$$  angegebenen Wert.
- $\&=- (Pm+nnn)$**  Wie  $\&=- (nnn)$ , jedoch Einrückung auf die Position, die  $nnn$  Punkt rechts von der mit  $\&!Mm$  gemerkten Position liegt.
- $\&=- (Pm-nnn)$**  Wie  $\&=- (nnn)$ , jedoch Einrückung auf die Position, die  $nnn$  Punkt links von der mit  $\&!Mm$  gemerkten Position liegt.
- $\&=- (m/n)$**  Wie  $\&=-nnn$ ; der Bruch in der Klammer bedeutet Bruchteile der Satzbreite
- $\&= (nnn; z)$**  Aussparung für Initialie: Einrücken des Restes der Zeile um weitere  $nnn$  Punkt für insgesamt  $z$  Zeilen
- Ist der Text vor dieser Anweisung bereits eingerückt auf Grund einer der übrigen unter 10.1.2 genannten Steueranweisungen, so wird mit dieser Anweisung für insgesamt  $z$  Zeilen zusätzlich um  $nnn$  Punkt eingerückt.
- Die mit dieser Anweisung befohlene Aussparung für Initialien darf sich mit den in Kapitel 7.3 beschriebenen Aussparungen nicht überschneiden.
- $\&= (nnn; Vn)$**  wie  $\&= (nnn; z)$ , jedoch Aussparung bis zu der auf  $Vn$  gemerkten (weiter unten liegenden) vertikalen Position.
- $\&= (0; 1)$**  Noch wirksame Aussparung für Initialie aufheben.
- $\&= (999; z)$**  Wie  $\&= (nnn; z)$ , jedoch Einrücken der Folgezeilen auf die vor dieser Anweisung erreichte Satzbreite
- $\&= (m/n; z)$**  Wie  $\&= (nnn; z)$ ; der Bruch in der Klammer bedeutet Bruchteile der Satzbreite
- $\&= (Pm+nnn; z)$**  Wie  $\&= (Pm+nn)$ , für insgesamt  $z$  Zeilen.
- $\&= (Pm-nnn; z)$**  Wie  $\&= (Pm+nn)$ , für insgesamt  $z$  Zeilen.
- $\&=\$**  Nächste mit  $\&=$  beginnende Anweisung ignorieren
- $\&:$**  »Bibliographie-Eintrag« (hängender Einzug, 10 Punkt)
- Neue Zeile, linksbündig, Abschnittsgrenze; alle Folgezeilen bis zur nächsten Anweisung für neue Zeile 10 Punkt einziehen. Ausführliche Codierung mit gleicher Bedeutung:  $\$ \$ \$ 0 \$ \$ \$ \&=-010$



Wird diese Anweisung innerhalb von (eingerückten) Einschaltungen benutzt, so werden die Folgezeilen nicht zusätzlich eingezogen.

## 10.2. Zentrieren

Textteile, die zentriert werden sollen, müssen durch eine der Anweisungen \$, \$\$, \$\$\$, \$\$\$\$, &&, &&{, &&, &{, &, &{, &n&, &n&{, &!S{, eine der Anweisungen für Tabellensatz (Kapitel 10.3), eine der in Abschnitt 10.1.2 genannten Anweisungen oder durch Positionieren auf eine Merkstelle (durch &!Pn, nicht jedoch &!Bn; vgl. Kapitel 10.4) abgeschlossen sein (Ausnahme: Bereichszentrierung mit der Anweisung @-M). Soll danach weiterhin zentriert werden, so muss – außer nach \$\$\$\$, – die Zentrieranweisung wiederholt werden.

Steht die Zentrieranweisung unmittelbar vor einer Zeilenwechsellanweisung, so wird die ganze Zeile davor zentriert (gilt für alle unter 10.2 aufgeführten Anweisungen außer @.0, @..0, @...0, @:0, @::0, @:::0, @--0). Die Zentrieranweisung muss in diesem Fall ohne Leerzeichen hinter dem letzten Wort stehen.

Bei normalem Zeilenwechsel (der nicht durch Steueranweisung, sondern durch automatischen Zeilenumbruch entsteht) wird eine noch wirksame Zentrieranweisung ebenfalls ausgeführt, jedoch nicht aufgehoben.

Ist mehr als eine der unter 10.2 aufgeführten Anweisungen für einen Textteil angegeben, so wird nur die zuletzt angegebene Anweisung berücksichtigt.

- |                    |   |
|--------------------|---|
| <b>@-0</b>         | <b>Zeilenteilung</b><br>Der Rest der Zeile wird (von der Stelle der Anweisung an) nach rechts (auf Breite einer Tabellenspalte, Satzbreite, Merkstelle) abgeschlossen.  |
| <b>@-G</b>         | Wie @-0, aber nur in Spalten mit gerader Nummer   |
| <b>@-U</b>         | Wie @-0, aber nur in Spalten mit ungerader Nummer   |
| <b>@-L</b>         | Wie @-0, aber nur auf linken Seiten   |
| <b>@-R</b>         | Wie @-0, aber nur auf rechten Seiten  |
| <b>&amp;=- (R)</b> | Folgezeilen nach rechts ausschließen  |
| <b>@.0</b>         | <b>Zeilenteilung mit Auspunktieren</b><br>Wie @-0; der beim Ausschließen frei gebliebene Raum wird mit Punkten gefüllt, die untereinander stehen, falls in aufeinander folgenden Zeilen die gleiche Anweisung bei gleicher Schriftgröße verwendet wird. Die Punkte stehen im Abstand von 1 Geviert der jeweiligen Schriftgröße, jeweils am rechten Rand des Gevierts. Der erste und der letzte Punkt hat jeweils mindestens 1/3 Geviert Abstand zum umgebenden Text.<br><br>Steht unmittelbar nach @.0 eine Anweisung für Zeilenwechsel bzw. Spaltenwechsel bzw. für Positionierung auf eine Merkstelle, so werden die Punkte wie bei der Steueranweisung @:0 positioniert: der |

bis zum Zeilenende bzw. bis zur angegebenen Position frei bleibende Raum wird mit Punkten ausgefüllt, der letzte Punkt steht unmittelbar am Ende des auszupunktierenden Bereichs; der erste Punkt hat mindestens 1/3 Geviert Abstand von der zuvor erreichten Stelle.

Eine Zeile, in der nur die Anweisung @.0 bzw. @..0 steht, gilt nicht als leer.

@.0 Wie @.0; Punktabstand = 1/2 Geviert

@..0 Wie @.0; Punktabstand = 2/3 Geviert

@.0 Wie @.0; aber nicht weniger als zwei Punkte ausgeben

@..0 Wie @..0; aber nicht weniger als zwei Punkte ausgeben

@...0 Wie @...0; aber nicht weniger als zwei Punkte ausgeben

@:0 Auffüllen mit Punkten

Nach @:0 muss eine Anweisung für Zeilenwechsel bzw. Spaltenwechsel bzw. für Positionierung auf eine Merkstelle folgen. Der bis zum Zeilenende bzw. bis zur angegebenen Position frei bleibende Raum wird mit Punkten ausgefüllt.

Der Abstand der Punkte untereinander beträgt 1 Geviert; die Punkte stehen jeweils am rechten Geviertrand. Der letzte Punkt steht unmittelbar am Ende des auszupunktierenden Bereichs; der erste Punkt hat mindestens 1/3 Geviert Abstand von der zuvor erreichten Stelle.

@::0 Wie @:0; Punktabstand = 1/2 Geviert

@:::0 Wie @:0; Punktabstand = 2/3 Geviert

@--0 Zeilenteilung, Auffüllen mit einer Linie

Wie @.0, aber Auffüllen des ausgetriebenen Raums mit einer Linie auf der Schriftgrundlinie.

@-Z auf Mitte zentrieren, sonst wie @-0

Vgl. auch die Angabe IZENTE beim Parameter AUS.

@-M(na1, ne1) Zentrierung in einem horizontalen Feld

Der Bereich, der mit der Merkstelle na1 beginnt und an der aktuell erreichten Stelle endet, wird, falls er kürzer ist, in einem Feld zentriert, das an der selben Stelle beginnt und bei der in ne1 gemerkten Stelle endet. Die Merkstelle na1 muss die selbe horizontale Position bezeichnen, an der das Feld beginnt, in dem zentriert werden soll, aber eine davon verschiedene Nummer haben.

@-M(na1, ne1, na2) Zentrierung im kürzeren von zwei Feldern

Die beiden Bereiche, die in der gleichen Zeile liegen und an der gleichen horizontalen Position beginnen müssen, werden be-

stimmt durch die Merkstellen, deren Nummern in der Klammer angegeben werden, und die Position, an der diese Anweisung selbst steht. Das erste Zahlenpaar gibt die Nummer der Merkstellen an, die den Anfang und das Ende des ersten der beiden Bereiche bezeichnen; die dritte Zahl ist die Nummer der Merkstelle, an der der zweite Bereich beginnt; dieser endet an der Stelle, an der die Anweisung selbst steht. Die mit  $na_1$  und  $na_2$  angegebenen Merkstellen müssen verschiedene Nummern haben, aber die selbe horizontale Position bezeichnen. Der kürzere der beiden Bereiche wird horizontal so zentriert, dass er in der Mitte über bzw. unter dem längeren der beiden Bereiche steht. Hauptsächliches Anwendungsgebiet ist die Ausrichtung von Zähler und Nenner über bzw. unter dem Bruchstrich von gemeinen Brüchen.

**@-A**

Zeile austreiben

Die Zeile bzw. der Zeilenteil, in dem diese Anweisung steht, und die folgenden Zeilen sollen trotz nachfolgender Anweisung für Zeilenwechsel oder für Positionierung ausgetrieben werden. Steht die Anweisung unmittelbar vor einer Zeilenwechselanweisung, so sollte sie ohne Leerzeichen hinter dem letzten Wort stehen.

**&!R**

Wie @-A; vor und hinter &amp;!R muss ein Leerzeichen stehen

**@-N**

nicht austreiben

Die Zeile, in der diese Anweisung steht, und die folgenden Zeilen sollen trotz eingeschaltetem Blocksatz (vgl. Parameter AUS) nicht ausgetrieben werden.

Die Wirkung der Anweisung wird an den gleichen Stellen wieder aufgehoben, an denen auch die übrigen mit @- beginnenden Anweisungen beendet werden, die in 10.2 beschrieben sind.

### 10.3. Tabellen- und Spaltensatz

Zum Satz von Tabellen bieten die Anweisungen zum Verändern der Satzbreite (&!s (1, mmm), Kap. 4.3), zum Einrücken (&=nnn etc., Kap. 10.1.2) und zum Definieren und Wiederanfahen von vertikalen Merkstellen (&M(Vn) und &P(Vn), Kap. 10.4) in der Regel die notwendigen Funktionen. Zusätzlich gibt es die Möglichkeit, die Tabellen konsequent zeilenweise zu erfassen und die Positionierungen mit folgenden Anweisungen vorzunehmen.

**@n**

Ende einer Tabellenspalte

Der vor der Anweisung in der Zeile stehende Text wird auf  $1/n$  (möglich von  $@1 = 1/1$  bis  $@0 = 1/10$ ) der gesamten Satzbreite untergebracht (normalerweise mit Spatien aufgefüllt = Rausatz (häufig ungenau als Flattersatz bezeichnet); rechts und auf Mitte zentrierter Satz ist möglich durch Kombination mit @-0 und @-Z); danach steht die gesamte Satzspiegelbreite in der gleichen Zeile erneut zur Verfügung. Soll eine Tabellenspalte ausgetrieben werden, so muss vor dem ersten Wort der Tabellenspalte die Anweisung @-A stehen.

Der Text der ganzen Tabelle muss zeilenweise erfasst werden (d. h. erst Zeile 1 aller Tabellenspalten, dann Zeile 2 etc.). In den einzelnen Zeilen muss hinter dem Text für jede Tabellenspalte (auch hinter dem für die letzte) die Anweisung für das Ende der Tabellenspalte stehen. Die Zeilen sind durch eine der Anweisungen für Zeilenwechsel abzuschließen.

Beispiel: Eine Zeile einer Tabelle, die in der ersten Tabellenspalte auf maximal 1/3 der Satzspiegelbreite linksbündig eine Erläuterung, dahinter in 4 Tabellenspalten zu je 1/6 Satzspiegelbreite rechtsbündig Zahlenwerte enthält, könnte wie folgt codiert werden:

```
$$$ Text @3 @-0 11 @6 @-0 22 @6 @-0 33 @6 @-0 44 @6
$$$ nächste Zeile ...
```

Mit der Anweisung @1 (= ganze Satzbreite) kann, wenn sie als erste von den unter 10.3 genannten Anweisungen in einer Zeile benutzt wird, ab dem rechten Satzspiegelrand weitersetzt werden.

**@(nnn)** Wie @n; die Zahl nnn (1- bis 4-stellig, maximal Satzspiegelbreite) in der Klammer sind jedoch typographische Punkt

**@(m/n)** Wie @n; der Bruch in der Klammer gibt Bruchteile der Satzbreite an.

Zum Ablauf: Nach jeder Ausführung einer der unter 10.3 genannten Anweisungen steht in der gleichen Zeile die gesamte Satzspiegelbreite von der erreichten Position an noch einmal zur Verfügung. Nach der ersten Ausführung der Anweisung @2 in einer Zeile würde also, wenn die Zeile nicht vorher abgeschlossen wird, der Satzspiegel um 1/2 Satzspiegelbreite überschritten, nach der zweiten Ausführung der gleichen Anweisung in einer Zeile bereits um eine volle Satzspiegelbreite usw. Es würde somit beliebig weit (bis zur Breite des verwendeten Materials) über den rechten Satzspiegelrand hinaus gesetzt werden. Deshalb muss auch die letzte Tabellenspalte mit einer Anweisung für den dafür vorgesehenen Bruchteil der Satzbreite abgeschlossen werden.

## 10.4. Merkstellen; Positionieren; Textfelder

**&!Mn** Horizontale Merkstelle Nummer n definieren

Das Programm merkt sich die Stelle innerhalb der Zeile, an der die Anweisung gegeben wird, und berücksichtigt evtl. durch das Austreiben der Zeile verursachte Verschiebungen innerhalb der Zeile. Bis zu 10 horizontale Merkstellen sind möglich (für  $n = 1...9,0$ ). Sollen mehr als 10 Stellen gemerkt werden, so ist die Anweisung &!M(n) zu verwenden (siehe unten).

Steht &!Mn zwischen Leerzeichen, so gilt das Leerzeichen hinter &!Mn als nicht vorhanden. Es wird also die Position des nächsten Wortanfangs (die nach dem Spatium, ggf. nach erfolgtem Austreiben der Zeile, erreichte Stelle) gemerkt; dies kann auch der Anfang einer neuen Zeile sein.

Wird für die 10. Merkstelle statt der Ziffer 0 der Buchstabe o verwendet, so wird die neue Stelle nur dann gemerkt, wenn die alte bisher als 10. gemerkte Stelle weiter links liegt.

**&!M(n)** Horizontale Merkstelle Nummer n definieren

Wie &!Mn; diese Form muss für die horizontalen Merkstellen 11–100 benutzt werden. In Klammern können die Zahlen 0 bis 100 angegeben werden; die Anweisung &!M(0) ist gleichbedeutend mit der Anweisung &!M(10).

Wird für die 100. Merkstelle statt der Anweisung &!M(100) die Anweisung &!MZ verwendet, so wird (analog zur Anweisung &!MO statt &!M0) die neue Stelle nur dann gemerkt, wenn die alte bisher als 100. gemerkte Stelle weiter links liegt.

**&!M(!n)** Wie &!M(n); die so gemerkte horizontale Position wird nach evtl. Austreiben bzw. Zentrieren der Zeile in die PROTOKOLL-Datei ausgegeben

**&!M(n1–n2)** Wie &!M(n), jedoch werden die Merkstellen mit den Nummern n1 bis n2 (je einschließlich) definiert

**&!M-** Alle horizontalen Merkstellen auf 0 setzen

In Textteil und Fußnoten sind die Merkstellen jeweils unabhängig voneinander, es sei denn, dass gleichzeitig Apparateinträge zu verarbeiten sind und die Fußnoten deshalb zusammen mit dem Text gesetzt werden.

**&!Pn** Positionieren auf Merkstelle Nummer n

Das Programm positioniert in der aktuellen Zeile auf die zuletzt (auch über Spalten- und Seitengrenze hinweg) unter der gleichen Nummer gemerkte horizontale Position.

&!Pn kann auch Endpunkt für Rechts- oder Mitte-Zentrierung sein.

**&!P(n)** Wie &!Pn; diese Form muss für Positionierung auf die Merkstellen 11–100 gewählt werden.

Beispiel:

```
$$$ Der Wind &!m1rauscht__ &!m2in___ &!m3den
&!m4Zweigen
$$$&!p1raschelt &!p2in &!p3den &!p4Blättern
$$$&!p1weht &!p2durch &!p4Bäume
$$$&!p4Blätter fallen.
```

wird:

```
Der Wind rauscht in den Zweigen
           raschelt in den Blättern
           weht durch Bäume
                           Blätter fallen
```

- &!Bn** Positionieren auf Merkstelle Nummer  $n$  innerhalb der Zeile
- Wie &!P $n$ , jedoch wird vorausgesetzt, dass eine in der gleichen Zeile gemerkte Position gemeint ist; beim Zeilenausgleich wird sie (im Gegensatz zu &!P $n$ , die unabhängig vom Zeilenausgleich fest bleibt) mitverschoben. Kann nicht Endpunkt für Zentrieranweisungen sein. Im Satzprotokoll wird kein Hinweis auf mögliche Doppelbelichtung gegeben.
- Unterschied in der technischen Realisierung:  
 &!P $n$  : Schreibstrahl zum Zeilenanfang, Leerbildlinien bis zur gemerkten Stelle; davor stehende Leerzeichen werden nicht zum Austreiben der Zeile herangezogen.  
 &!B $n$  : positives oder negatives Verschieben des Schreibstrahls von der erreichten Position aus; davor stehende Leerzeichen bleiben für das Austreiben der Zeile verfügbar.
- &!B (n)** Wie &!B $n$ ; diese Form muss für Positionierung auf die Merkstellen 11–100 gewählt werden.
- &!P (+n)** Satzspiegel um  $n$  Punkt nach rechts verschieben
- Der Satzspiegel wird für den Rest der Spalte um  $n$  Punkt nach rechts verschoben. Die Verschiebung kann durch &!P ( $-n$ ) mit dem gleichen Wert für  $n$  rückgängig gemacht werden. Sie wird am Beginn der nächsten Spalte automatisch aufgehoben.
- &!P (-n)** Satzspiegel um  $n$  Punkt nach links verschieben
- Der Satzspiegel wird für den Rest der Spalte um  $n$  Punkt nach links verschoben, ggf. auch über den Bereich hinaus, der durch den Parameter MON für den linken Rand vorgesehen ist. Eine Prüfung erfolgt nicht. Die Verschiebung kann durch &!P ( $+n$ ) mit dem gleichen Wert für  $n$  rückgängig gemacht werden. Sie wird am Beginn der nächsten Spalte automatisch aufgehoben.
- &!P (+-0)** vorangegangene Verschiebung des Satzspiegels aufheben
- Eine noch aktive Verschiebung des Satzspiegels wird aufgehoben, der Satzspiegel wird wieder auf die über Parameter angegebenen Werte eingestellt.
- &!M (Vn)** Vertikale Merkstelle Nummer  $n$  definieren
- In den Klammern können hinter einem »v« die Nummern 1 bis 99 für bis zu 99 Merkpositionen angegeben werden, auf denen jeweils die augenblicklich erreichte vertikale Position immer dann gemerkt wird, wenn auf der jeweiligen Merkstelle nicht schon eine vertikale Position der aktuellen Seite (bei mehrspaltigem Satz: der aktuellen Spalte) gemerkt ist, die weiter unten auf der Seite bzw. Spalte liegt als die augenblicklich erreichte. Diese Positionen können durch die Steueranweisung &!P (V $n$ ) in der aktuellen Seite bzw. Spalte wieder angesteuert werden. Die  $x$ - und  $y$ -Koordinaten der gemerkten Stellen können gleichzeitig als Endpunkte für Linien

dienen, die mit der Anweisung  $\&! / (a, b, n)$  auf der aktuellen Seite bzw. Spalte gezogen werden können, auf der diese Punkte definiert wurden. Variable Vorschübe, die auf der Seite vor dieser Anweisung liegen, werden zu festen Vorschüben.

**$\&!M(Wn)$**  Wie  $\&!M(Vn)$ , jedoch bleiben die variablen Vorschübe vor dieser Anweisung variabel.

**$\&!M(Vn1-n2)$**  Wie  $\&!M(Vn)$ , jedoch werden die vertikalen Merkstellen mit den Nummer  $n1$  bis  $n2$  (je einschließlich) definiert

**$\&!P(Vn)$**  Positionieren auf vertikale Merkstelle Nummer  $n$  auf der aktuellen Seite bzw. Spalte

Die augenblicklich erreichte horizontale Position wird durch das vertikale Positionieren nicht verändert. Variable Vorschübe, die auf der Seite vor dieser Anweisung liegen, werden zu festen Vorschüben. Unmittelbar nachfolgende Leerzeilen werden wie am Seitenanfang behandelt. Wird auf eine vertikale Position Bezug genommen, die auf der Seite / in der Spalte noch nicht definiert ist, so kann die Satzausgabe später nicht in eine PostScript-Datei umgewandelt werden.

**$\&!P(Wn)$**  Wie  $\&!P(Vn)$ , jedoch bleiben die variablen Vorschübe vor dieser Anweisung variabel.

**$\&!P(Hn)$**  Vertikales Positionieren auf die Mitte zwischen aktueller vertikaler Position und vertikaler Merkstelle Nummer  $n$ . Diese muss oberhalb der aktuellen vertikalen Position liegen

**$\&!E$**  Erreichte Position ausdrucken

Die horizontale und vertikale Position, die mit dieser Anweisung in der Spalte erreicht ist, wird als Testhilfe ins Satzprotokoll ausgegeben. Die in der Zeile horizontal erreichte Position (vom Satzspiegelrand an gemessen) wird in Bildlinien angegeben. Für die in der Spalte erreichte vertikale Position (das ist die Position der Schriftgrundlinie eines dort zu setzenden Zeichens) werden zwei Zahlenwerte ausgegeben; der erste gibt die vom oberen Satzspiegelrand an erreichte Position in Punkt an, die zweite, hinter »+« stehende Zahl gibt den Platzbedarf (in Punkt) für bis dahin aufgerufene Fußnoten und Apparate an. Die in der Spalte erreichte vertikale Position wird für alle mit  $\&!E$  bezeichneten Positionen nach dem Spaltenhöhenausgleich ein zweites Mal ausgegeben. Diese Positionen werden dabei spaltenweise laufend durchnummeriert; die Angabe der vertikalen Position entspricht dem Abstand vom oberen Satzspiegelrand nach dem Spaltenhöhenausgleich.

Wird innerhalb einer Zeile (einschl. Zeilenende) die Schriftgrundlinie verschoben (z. B. mit  $\&!P(Vn)$ ), so wird in die zweite Protokoll-Datei die nach der Verschiebung erreichte Position der Schriftgrundlinie ausgegeben, es sei denn, dass innerhalb der Zeile die Anweisung  $\&!E$  benutzt wurde. In diesem Fall wird die beim letzten Aufruf von  $\&!E$  in der Zeile geltende Position ausgegeben.

- &!Tn+** Anfang eines Textfeldes
- Diese Anweisung definiert den Anfang eines Textfeldes mit der Nummer  $n$ , das später noch einmal (z. B. über ein Graurasterfeld mit der gleichen Ausdehnung wie das Textfeld) ausgegeben werden soll;  $n$  ist eine Zahl zwischen 1 und 100 (je einschließlich).
- &!Tn-** Ende des mit &!Tn+ beginnenden Textfeldes mit der gleichen Nummer
- Das Textfeld wird ausgegeben und zu einer evtl. Wiederholung der Ausgabe an der gleichen Stelle der Seite vorgemerkt.
- &!Tn.** Erneute Ausgabe des Textfeldes Nummer  $n$
- Das zwischen den Anweisungen &!Tn+ und &!Tn- gesetzte Textfeld Nummer  $n$  wird erneut ausgegeben, und zwar an der selben Stelle der selben Seite wie bei der ersten Ausgabe.

## 10.5. Zeilenzähler; Marginalien

Zu Beginn des Satzprogrammablaufs ist die Zeilennummerierung eingeschaltet, wenn mit dem Parameter ZLN eine Zeilennummerierung verlangt wurde. Die mit dem Parameter ZLN näher bestimmte Nummerierung wird mit den folgenden Anweisungen aus- und wieder eingeschaltet.

- &!R.** Ausschalten der automatischen Zeilennummerierung am Rand
- &!R-** gleichbedeutend mit &!R.
- &!R+** Einschalten der automatischen Zeilennummerierung am Rand

Die Zeilennummerierung erfolgt seiten- bzw. spaltenweise. Mit den folgenden Anweisungen kann eine abweichende Nummerierung verlangt werden.

- &!R(nn)** Umschalten von seitenweiser Nummerierung auf fortlaufende Nummerierung.

Für die Nummer der aktuellen Zeile wird  $nn$  festgesetzt.

Ist schon auf fortlaufende Nummerierung umgeschaltet, so wird nur die für die aktuelle Zeile geltende Nummer neu festgesetzt. Derzeit werden nur die letzten 4 Ziffern der Zeilennummern ausgegeben.

Ist die am Rand automatisch ausgegebene (d. h. nicht als Marginalien-Text gesetzte) Zeilennummer nicht die laufende Nummer der Zeile auf der aktuellen Seite / Spalte, sondern eine mit &!R(nn) erzeugte Nummer, und ist gleichzeitig zu Parameter ZLN als 7. Wert (IZNPR) ein von 0 verschiedener Wert angegeben, so wird in der Zielfeld hinter dem Text der betreffenden Zeilen eine eigene Fortsetzungszeile mit der Unterscheidungsnummer 990 und dem Text <!-- ln8888 --> ausgegeben, wobei für 8888 die aktuelle Nummer eingesetzt wird.

Sind in der Quelldatei Zeilen enthalten, die nur aus <!-- ln8888 --> bestehen, so werden diese bei der Eingabe übergangen.



<b>&amp;!R(U)</b>	Unterbrechen der Zählung der Zeilen (nur bei fortlaufender Nummerierung)
<b>&amp;!R(W)</b>	Wiederaufnahme der Zählung der Zeilen (nur bei fortlaufender Nummerierung)
<b>&amp;!R(-1)</b>	Umschalten von fortlaufender Nummerierung auf seitenweise Nummerierung. Statt -1 kann ein beliebiger negativer Wert angegeben werden.
<b>&amp;!R(:3)</b>	Ab hier bei zweistelligen Zeilennummern ein zusätzliches (ziffernbreites) Spatium vor der Zeilennummer einfügen, um auf gleicher Seite vorkommende dreistellige Zeilennummern rechtsbündig übereinander zu setzen.
<b>&amp;!R(:2)</b>	Zurücksetzen auf Voreinstellung: ab hier kein zusätzliches Spatium vor zweistelligen Zeilennummern.

Die Marginalien-Anweisungen stehen in der Regel zwischen Leerzeichen. Nur wenn in den Marginalien selbst keine Leerzeichen und keine Zentrieranweisungen vorkommen, dürfen sie mitten im Wort codiert werden.

Mit diesen Anweisungen sind nur einzeilige Marginalien zu verwalten. Zu einer Textzeile können maximal 4 Marginalien gesetzt werden. Ab der 3. Marginalie in einer Zeile wird eine Warnung ins Satzprotokoll ausgegeben.

Schriftgröße, Schriftart, Breite der für die Marginalien vorgesehenen Felder, Abstand dieser Felder vom Rand, Lage der Marginalien in diesen Feldern werden dem Satzprogramm mit den Parametern `MAL` und `MAR` bzw. `MLS` und `MRS` getrennt für linken und rechten Rand mitgeteilt. Bei »Marginalien innen« und »Marginalien außen« müssen die Angaben für das linke und das rechte Feld jedoch identisch sein.

Auszeichnungen innerhalb der Marginalien: Die Marginalien werden als ganze in der mit dem entsprechenden Parameter angegebenen Schrift gesetzt. Soll davon abgewichen werden, so ist mit den genannten Parametern als Marginalien-Schrift die Grundschrift des Textes anzugeben; evtl. im Text selbst enthaltene Auszeichnungen müssen vor der Marginalien-Anweisung aufgehoben sein. Andere Lösungen sind in begrenztem Umfang über Makros möglich.

Positionierung der Marginalien: Alle linken Marginalien (auch die durch `@I` und `@A` entstandenen) kommen an den linken Seitenrand, alle rechten Marginalien (auch die durch `@I` und `@A` entstandenen) kommen an den rechten Rand der jeweiligen Spalte.

<b>@L . . @f</b>	Marginalien für den linken Seitenrand
<b>@R . . @f</b>	Marginalien für den rechten Spaltenrand
<b>@I . . @f</b>	Marginalien innen

Diese Anweisung ist nur bei einspaltigem Satz sinnvoll; die Marginalien werden bei Spalten mit ungerader Nummer auf dem linken Rand, bei Spalten mit gerader Nummer auf dem rechten Rand gesetzt.

**@A...@{**

Marginalien außen

Diese Anweisung ist nur bei einspaltigem Satz sinnvoll; die Marginalien werden bei Spalten mit ungerader Nummer auf dem rechten Rand, bei Spalten mit gerader Nummer auf dem linken Rand gesetzt.

**@S...@{**

Marginalien links innerhalb des Satzspiegels (der Text muss entsprechend eingezogen sein)

**@T...@{**

Marginalien rechts innerhalb des Satzspiegels (der Text muss entsprechend eingezogen sein)

### III. Die darzustellenden Zeichen

## 11. Auszeichnungen

### 11.1. Auszeichnungsschriften, Sperrung

Die unter 11. genannten Anweisungen müssen ohne Leerzeichen vor bzw. nach dem auszuzeichnenden Text stehen.

Auszeichnungen gelten jeweils höchstens bis zum Ende des Abschnitts, in dem sie angegeben wurden. Auszeichnungs-Ende wird automatisch an den Abschnittsenden (d. i. bei `$`, `$$`, `$$$`, `$$$n`, `$$$-n`, `$$$+n`, `$$$=$$$` etc. sowie vor und nach Titelzeilen) und vor Kolummentiteln vom Programm ergänzt. Soll eine Auszeichnung über eine solche Abschnittsgrenze hinweg erhalten bleiben, so kann vor der Abschnittsgrenze die Steueranweisung `\\` (siehe unten) gegeben oder die Auszeichnungsanweisung am Beginn des nächsten Abschnitts (nach den entsprechenden Anweisungen) wiederholt werden.

Verschachtelung von Schriftumschaltungen: Wird von einer Auszeichnungsschrift direkt auf eine andere Auszeichnungsschrift umgeschaltet, so wird die vor der Umschaltanweisung benutzte Schrift gemerkt und die verlangte Umschaltung wird ausgeführt. Nach Beendigung der zuletzt verlangten Schriftumschaltung wird auf die Schrift zurückgeschaltet, aus der heraus die Schriftumschaltung vorgenommen wurde. Auf diese Weise kann z. B. innerhalb eines kursiven Textstückes auf griechische Schrift umgeschaltet werden, ohne dass die Umschaltung auf Kursivschrift beendet wird. Wenn die griechische Schrift beendet wird, wird wieder mit (lateinischer) Kursivschrift weitergesetzt.

Umschaltungen mit mnemonischen Anweisungen auf kursive und (halb)fette Schriften sowie Umschaltungen mit mnemonischen Anweisungen aus kursivem oder (halb)fettem Bereich heraus auf eine andere Schrift (z. B. Griechisch) erfolgen nur, wenn eine Schrift, die die Eigenschaften der vor der Umschaltung benutzten Schrift und die mit der Umschaltung verlangte Eigenschaft besitzt, mit dem Parameter `SCH` angegeben ist. Beispiel: `#/+...#G+` verlangt zunächst eine kursive und dann eine kursive griechische Schrift, `#F+...#/+...#R+` verlangt zunächst ein halbfette, dann eine halbfette kursive und schließlich eine halbfette kursive russische Schrift. Ist keine solche Schrift mit dem Parameter `SCH` angegeben, so wird eine Fehlermeldung ins Protokoll ausgegeben und auf eine Schrift umgeschaltet, die wenigstens die mit der Umschaltung geforderte Eigenschaft besitzt (z. B. `#R+aaa#/+bbb#F+ggg` zeigt als letzte Buchstabenfolge ein lateinisches fettes schräg gestelltes (nicht kursives) `ggg`, wenn keine halbfette kursive russische Schrift angegeben ist).

Schrägstellung und Sperrung können mit Schriftumschaltungen kombiniert werden, z. B.: `} } }` für 2. Schrift schräg Anfang, `#K+#S+` für Kapitalchen gesperrt Anfang.

Die Sperrung wird so lange beibehalten, bis sie durch #S-, #?- , { oder automatisch am Abschnittsende aufgehoben wird.

Die Schrägstellung wird nicht nur am Abschnittsende oder durch } { oder #/ (nn) - bzw. { aufgehoben, sondern auch zusammen mit der Ende-Anweisung für eine Schriftumschaltung, wenn die Anweisung zur Schrägstellung nach der Anweisung zur Schriftumschaltung gegeben wurde. Wird im schräg gestellten Bereich auf eine andere Schrift umgeschaltet, so wird auch die neue Schrift schräg gestellt. Wird eine Schrift beendet, so wird mit schräg gestellter oder aufrecht stehender Schrift weitergefahren, je nachdem, ob vor der Umschaltung auf die so beendete Schrift die Schrägstellung eingeschaltet war oder nicht (unabhängig davon, ob innerhalb der mit dieser Anweisung beendeten Schrift die Schrägstellung aufgehoben wurde oder nicht).

Falls in einem Satzprogramm ein Kapitälchen verwendet werden, müssen diese im Umschaltbereich 3 (Anweisung ++, vgl. Parameter SCH) liegen. Bei Textstücken, in denen Großbuchstaben und Kapitälchen gemischt vorkommen und die (im Kapitälchen-Zeichensatz nicht vorhandene) Interpunktionszeichen enthalten, genügt es, zu Anfang die Anweisung #K+ zu geben. Vor Großbuchstaben und Sonderzeichen (jedoch nicht vor Ziffern) wird automatisch auf die Hauptschrift umgeschaltet, anschließend wieder auf Kapitälchen zurückgeschaltet, bis eine Auszeichnungs-Ende-Anweisung gefunden wird. Dies gilt jedoch nicht, wenn mit dem Parameter KAP dem Satzprogramm mitgeteilt wird, dass außer den Kapitälchen auch die übrigen Zeichen im gleichen Umschaltbereich vorhanden sind.

- \\\ Bei der nächsten Abschnittsgrenze Auszeichnungen nicht aufheben. Diese Anweisung muss zwischen Leerzeichen stehen.
- #?+ Auszeichnungen bei Abschnittsgrenze nicht automatisch aufheben (wird durch #?- wieder rückgängig gemacht, s. u.)

### 11.1.1. Auszeichnung über mnemonische Anweisungen

Diese Anweisungen setzen (außer für Sperrung) voraus, dass die entsprechende Schrift mit dem Parameter SCH für einen der 16 Umschaltbereiche angegeben wird.

- #S+ Sperrung Anfang. Der erste zusätzliche Zwischenraum wird – außer am Wortanfang – an dieser Stelle (also z. B. vor dem nachfolgenden Buchstaben) eingefügt.
- #S\* Sperrung Anfang. Der erste zusätzliche Zwischenraum wird – außer am Wortanfang – erst hinter dem nächsten Zeichen eingefügt.
- #S- Sperrung Ende. An der Stelle, an der diese Anweisung steht, wird – außer am Zeilenende – ein zusätzlicher Zwischenraum eingefügt.
- #S= Sperrung Ende. An der Stelle, an der diese Anweisung steht, wird kein zusätzlicher Zwischenraum mehr eingefügt.

Der Umfang der mit #S+ und #S\* codierten Sperrung ist über den Parameter DIC festlegbar. Voreinstellung: 1/8 Geviert.

<b>#Z+</b>	Sperrung (fest, 1/8 Geviert) Anfang
<b>#Z (nn) +</b>	Sperrung (fest) mit nn/1000 Geviert Anfang
<b>#Z-</b>	Sperrung (fest) Ende
	Um die Regelung für den Abstand am Anfang und am Ende der Sperrung zu ändern, kann mit der gleichen Wirkung wie bei #S+/#S- auch #Z*/#Z= benutzt werden.
<b>#K+</b>	Kapitälchen Anfang
<b>#K-</b>	Kapitälchen Ende
	Diese Steueranweisung kann auch dann am Anfang eines Wortes gegeben werden, wenn es in Kapitälchen mit Anfangsversalien gesetzt werden soll. Nur die Kleinbuchstaben werden in Kapitälchen verwandelt.
	Kapitälchenschriften müssen im Umschaltbereich 3 liegen.
<b>#Q+</b>	Kapitälchen Anfang
<b>#Q-</b>	Kapitälchen Ende
	Mit dieser Steueranweisung werden auch Großbuchstaben in Kapitälchen verwandelt. Dies erlaubt z. B. bei der Codierung von Überschriften oder Kolummentiteln, die in Kapitälchen gesetzt werden sollen, die Beibehaltung der korrekten Groß-Klein-Schreibung.
<b>#V+</b>	Versalien Anfang
<b>#V-</b>	Versalien Ende
	Mit dieser Steueranweisung werden alle Buchstaben in Versalien (Großbuchstaben) verwandelt.
<b>#W+</b>	Gemeine Anfang
<b>#W-</b>	Gemeine Ende
	Mit dieser Steueranweisung werden alle Buchstaben in Gemeine (Kleinbuchstaben) verwandelt.
	Achtung: { hebt die Umschaltung #V+ bzw #W+ nicht auf.
<b>#/+</b>	Kursivschrift Anfang
<b>#/-</b>	Kursivschrift Ende
<b>#F+</b>	(halb)fett Anfang
<b>#F-</b>	(halb)fett Ende
<b>#D+</b>	Fraktur (»Deutsche Schrift«) Anfang
<b>#D-</b>	Fraktur (»Deutsche Schrift«) Ende
	Die Belegung der am Markt erhältlichen Fraktur-Fonts ist nicht einheitlich. Das Satzprogramm setzt für Fraktur-Schriften derzeit die in den Fonts der Firma Gerda Delbanco benutzte Belegung und Schriftnummern zwischen 48001 und 48199 voraus.

---

<b>#B+</b>	Umschaltung auf Basis-Schnitt (»roman«) des gerade benutzten (z. B. kursiven oder fetten) Fonts
<b>#B-</b>	Rückschaltung vom Basis-Schnitt (»roman«) auf den zuletzt vor der Umschaltung mit #B+ benutzten Schriftschnitt
<b>#X+</b>	Austauschen der Codierung für Standardziffern und Alternativziffern
<b>#X-</b>	Rückschaltung der Codierung für Standardziffern und Alternativziffern auf die jeweils im Parameter SCH angegebenen bzw. voreingestellten Werte.
<b>#A+</b>	Arabisch Anfang
<b>#A-</b>	Arabisch Ende
<b>#G+</b>	Griechisch Anfang
<b>#G-</b>	Griechisch Ende
<b>#T+</b>	Koptisch Anfang
<b>#T-</b>	Koptisch Ende
<b>#H+</b>	Hebräisch Anfang
<b>#H-</b>	Hebräisch Ende
<b>#P+</b>	Phonetisches Alphabet (IPA) Anfang
<b>#P-</b>	Phonetisches Alphabet (IPA) Ende
<b>#R+</b>	Kyrillisch (Russisch) Anfang
<b>#R-</b>	Kyrillisch (Russisch) Ende
<b>#/(nn) +</b>	Schrägstellung um nn Grad Anfang
	Mit dieser Steueranweisung können die Schriftzeichen um nn (maximal 45) Grad schräg gestellt werden. nn kann auch eine negative Zahl sein. Auch die Zeichen einer Kursivschrift werden ggf. um den angegebenen Winkel zusätzlich schräg gestellt. Eine nach #/(nn) + stehende Anweisung #/(mm) + stellt nicht zusätzlich, sondern insgesamt um mm Grad schräg. Eine zwischen #/(nn) + und #/(nn) - bzw. #/() - stehende Anweisung »}« zur Schrägstellung wird ignoriert.
<b>#/(00) +</b>	Schrägstellung Ende
<b>#/(nn) -</b>	Schrägstellung Ende
<b>#/() -</b>	Wie #/(nn) -
<b>\#?-</b>	Ende der zuletzt codierten Schriftumschaltung (Sperrung und Unterstreichen werden nicht aufgehoben), Aufheben von #?+

#?– Ende aller unter 11.1.1, 11.1.2 und 11.2 aufgeführten Auszeichnungen, Aufheben von #?+

### 11.1.2. Auszeichnung über Anwahl des Umschaltbereichs

Diese Anweisungen können alternativ zu den oben beschriebenen (auch mit diesen gemischt) benutzt werden. Sie sind dann notwendig, wenn Schriften angewählt werden sollen, für die es keine mnemonischen Anweisungen gibt (z. B. eine lateinische Grotteskschrift als Auszeichnungsschrift bei einer Antiqua als Grundschrift).

Wird mit diesen Anweisungen auf eine Kursivschrift umgeschaltet, so muss diese zusätzlich schräg gestellt werden.

Alle Auszeichnungen, die über die Anwahl eines Umschaltbereichs vorgenommen werden, können durch die Anweisung »{ « (siehe unten) beendet werden. Die Steueranweisungen, mit denen das Verlassen eines bestimmten Umschaltbereichs angegeben wird, sind vor allem dann sinnvoll, wenn damit nicht auf die Grundschrift zurückgeschaltet, sondern in einen zuvor mit der Steueranweisung »{ « (siehe unten) gemerkten Umschaltbereich zurückgeschaltet werden soll.

} Schrägstellung der Schrift Anfang

Diese Steueranweisung wird benötigt, um Kursivschriften, die durch Anwahl des Umschaltbereichs angegeben wurden, schräg zu stellen. Mit dieser Anweisung können auch alle anderen Schriften schräg gestellt (also pseudo-kursiv gesetzt) werden. Im Unterschied zur Schrägstellung mit der Anweisung »#/ (nn) +« werden Kursivschriften mit der (auch dafür erforderlichen) Anweisung »} « nicht zusätzlich schräg gestellt.

Die Steueranweisung } zur Schrägstellung einer Schrift gilt so lange, bis sie durch } { oder { oder #?– aufgehoben wird oder bis sie durch die Steueranweisung für das Ende der Schrift, innerhalb derer sie begonnen wurde, zusammen mit dieser beendet wird.

Wird im schräggestellten Bereich mit einer unter 11.1.2 aufgeführten Anweisung auf eine andere Schrift umgeschaltet, so gilt die Schrägstellung auch für diese Schrift.

Wird mit einer der »mnemonischen Anweisungen« (Kap. 11.1.1) auf eine andere Schrift umgeschaltet, so wird die Schrägstellung nur für die Schriften beibehalten, die bis auf die Einerstelle die gleiche Schriftnummer haben wie die Schrift, innerhalb derer die Schrägstellung begonnen wurde (z. B. für die nach #f+ stehenden Zeichen nach schräggestellten Grundschriftzeichen). Nicht beibehalten wird die Schrägstellung auch bei einer Schrift, auf die mit #b+ umgeschaltet wurde, auch wenn es sich um die selbe Schrift wie die mit } schräggestellte handelt.

Analoges gilt für die Schrägstellung von Schriften nach der Anweisung #/+ und deren Aufhebung durch #/- oder { oder #?–.

} { Ende der Schrägstellung

}}	zweite Schrift Anfang (Umschaltbereich 2 im Parameter SCH, z. B. Halbfett, Griechisch, echte Kursive; echte Kursive muss zusätzlich schräg gestellt werden)
}}{	Ende der zweiten Schrift Mit dieser Anweisung wird auf die zuletzt vor der Umschaltung } } benutzte Schrift oder, falls keine Verschachtelung von Schriftumschaltungen vorliegt, auf die Grundschrift (Umschaltbereich 1) zurückgeschaltet. Eine evtl. noch wirksame Sperrung wird nicht aufgehoben; die Schrägstellung der nach dieser Anweisung stehenden Textteile richtet sich danach, ob vor der entsprechenden Umschaltanweisung } } die Schrägstellung eingeschaltet war oder nicht.
++	dritte Schrift Anfang (Umschaltbereich 3 im Parameter SCH; normalerweise Kapitälchen, siehe unten)
++{	Ende der dritten Schrift; sonst wie } } {
@@	vierte Schrift Anfang (Umschaltbereich 4 im Parameter SCH)
@@{	Ende der vierten Schrift; sonst wie } } {
##	fünfte Schrift Anfang (Umschaltbereich 5 im Parameter SCH)
##{	Ende der fünften Schrift; sonst wie } } {
""	sechste Schrift Anfang (Umschaltbereich 6 im Parameter SCH)
""{	Ende der sechsten Schrift; sonst wie } } {
>>	siebte Schrift Anfang (Umschaltbereich 7 im Parameter SCH)
>>{	Ende der siebten Schrift; sonst wie } } {
<<	achte Schrift Anfang (Umschaltbereich 8 im Parameter SCH)
<<{	Ende der achten Schrift; sonst wie } } {
{ {	(vorübergehendes) Umschalten auf die erste Schrift (Grundschrift, Umschaltbereich 1) Die vor dieser Anweisung zuletzt gewählte Schrift und evtl. Schrägstellung wird gemerkt; Schrägstellung wird nicht aufgehoben. Im Unterschied zu den übrigen Schriftumschaltungen wird für { { bei verschachtelten Umschaltungen keine Warnung ausgegeben.
{ { {	Ende der ersten Schrift; sonst wie } } {
!! (n)	n-te Schrift Anfang (für $n = 1-16$ ) Die Umschaltbereiche 9-16 können nur mit dieser Anweisung angesprochen werden. Für die ersten 8 Umschaltbereiche kann diese Anweisung alternativ zu den vorgenannten Anweisungen benutzt werden.
!! (n) {	Ende der n-ten Schrift (für $n = 1-16$ ); sonst wie } } {



\{	Ende der zuletzt codierten Schriftumschaltung (Sperrung und Unterstreichungen werden nicht aufgehoben)
{	<p>Ende aller Auszeichnungen</p> <p>Hebt auch alle unter 11.1.1 genannten Auszeichnungen auf, nicht jedoch die unter 11.2 genannten; die (evtl. noch nicht abgearbeitete) Liste für wegen verschachtelter Schriftumschaltungen gemerkte Schriften wird gelöscht.</p> <p>Innerhalb von Marginalien verwendet, beschränkt sich die Wirkung von »{« bezüglich Schriftumschaltung und Schrägstellung auf die Marginalie. Eine evtl. eingeschaltete Sperrung wird durch »{« jedoch ganz (also auch für den Text außerhalb der Marginalie) aufgehoben.</p>

## 11.2. Über-, Durch-, Unterstreichung, Unterpunktierung

Eine Unterstreichung (Überstreichung, Durchstreichung) kann mit jeder anderen Art der Auszeichnung (nicht jedoch mit anderen Unterstreichungen) kombiniert werden. Die Auszeichnungs-Ende-Anweisung »{« hebt eine Unterstreichung (Überstreichung, Durchstreichung) nicht auf.

In einer Zeile können beliebig viele Bereiche beliebiger Länge unterstrichen (überstrichen, durchstrichen) werden; ein unterstrichener (überstrichener, durchstrichener) Bereich darf sich über beliebig viele Zeilen erstrecken; am Abschnittsende wird jedoch ein evtl. noch ausstehendes Ende der Unterstreichung (Überstreichung, Durchstreichung) automatisch vom Programm ergänzt.

In jeder Zeile können unterstrichene (überstrichene, durchstrichene) Spatien nur in einem einzigen unterstrichenen (überstrichenen, durchstrichenen) Bereich zum Austreiben der Zeile mit herangezogen werden. Sind in einer Zeile in mehr als einem solchen Bereich Spatien enthalten, so können im zweiten und den folgenden Bereichen die Spatien beim Austreiben nicht verändert werden.

Werden Teile des Textes farbig gesetzt, so erscheint die gesamte Unterstreichung in der Farbe, die bei der Anweisung für den Anfang der Unterstreichung eingestellt war, unabhängig davon, ob die Farbe des unterstrichenen Textes vor dem Ende der Unterstreichung geändert wird.

Werden im laufenden Text Schriftgrößenwechsel mit `&!K(nn)` vorgenommen und soll innerhalb der in veränderter Schriftgröße gesetzten Bereiche Text mit Grauraster hinterlegt werden, so wird der Grauraster nur dann zuverlässig der veränderten Schriftgröße angepasst, wenn innerhalb des hinterlegten Bereichs keine Spatien oder Zeilenwechsel vorkommen.

Sollen innerhalb einer Zeile hoch- oder tiefgestellte Bereiche unterstrichen (überstrichen, durchstrichen) werden, so dürfen innerhalb dieser Bereiche keine Leerzeichen (Spatien) vorkommen; andernfalls wird die Unterstreichung (Überstreichung, Durchstreichung) auf die Schriftgrundlinie der ganzen Zeile bezogen und nicht mit hoch- bzw. tiefgestellt. Ggf. müssen statt der Spatien feste Ausschlüsse verwendet werden.

Die Linienstärke für Über-, Durch-, und Unterstreichung ist von der augenblicklich verwendeten Schriftgröße abhängig. Sie beträgt bis zu einer Schriftgröße von 9 Punkt 0,2 Punkt, darüber für jeden weiteren Punkt zusätzlich 0,025 Punkt.

#0+	Durchstreichung Anfang
#0-	Ende der Durchstreichung
#1+	einfache Unterstreichung Anfang
#1-	Ende der einfachen Unterstreichung
#2+	doppelte Unterstreichung Anfang
#2-	Ende der doppelten Unterstreichung
#3+	halbfette Unterstreichung Anfang
#3-	Ende der halbfetten Unterstreichung
#4+	Unterpunktierung Anfang
#4-	Ende der Unterpunktierung
#5+	tiefe Überstreichung (Kleinbuchstaben) Anfang
#5-	Ende der tiefen Überstreichung
#6+	hohe Überstreichung (Großbuchstaben) Anfang
#6-	Ende der hohem Überstreichung
#7+	helle Unterlegung Anfang
#7-	Ende der hellen Unterlegung
#8+	dunkle Unterlegung Anfang
#8-	Ende der dunklen Unterlegung
#9+	tiefe Unterstreichung Anfang
#9-	Ende der tiefen Unterstreichung
#10+ . . . #10-	Kombination von #1+...#1- und #0+...#0-
#20+ . . . #20-	Kombination von #2+...#2- und #0+...#0-
#30+ . . . #30-	Kombination von #3+...#3- und #0+...#0-
#49+ . . . #49-	Kombination von #4+...#4- und #9+...#9-
#50+ . . . #50-	Kombination von #5+...#5- und #0+...#0-
#51+ . . . #51-	Kombination von #5+...#5- und #1+...#1-
#52+ . . . #52-	Kombination von #5+...#5- und #2+...#2-
#53+ . . . #53-	Kombination von #5+...#5- und #3+...#3-
#60+ . . . #60-	Kombination von #6+...#6- und #0+...#0-

#61+ . . . #61-	Kombination von #6+...#6- und #1+...#1-
#62+ . . . #62-	Kombination von #6+...#6- und #2+...#2-
#63+ . . . #63-	Kombination von #6+...#6- und #3+...#3-
#^0+	Unterstreichung Anfang (Art der Unterstreichung wird über Parameter UNT festgelegt)
#^0-	Ende der Unterstreichung
...	
#^9+	Unterstreichung Anfang (Art der Unterstreichung wird über Parameter UNT festgelegt)
#^9-	Ende der Unterstreichung
#?-	Ende der Über-, Durch-, Unterstreichung, Unterpunktierung (kann statt #0- .... #8- benutzt werden) und aller unter 11.1.1 und 11.1.2 aufgeführten Auszeichnungen

### 11.3. Änderung von Schriftgröße und Zeilenabstand

**&!K(nn)** Umschaltung auf Schriftgröße von nn Punkt

nn ist eine Zahl zwischen 4 und 144 (je einschließlich) unabhängig von der für den aktuellen Textteil über Parameter gewählten Schriftgröße.

Die Schriftgrade zwischen 4 und 24 Punkt können sowohl bei der Anweisung &!K(nn) als auch bei der Anweisung &!K(mm:nn) (siehe unten) in Viertelpunkt-Schritten angegeben bzw. mit den Anweisungen &!K(+n) bzw. &!K(-n) (siehe weiter unten) modifiziert werden, wobei eine aus der Modifizierung resultierende Schriftgröße von 24 oder mehr Punkt ganzzahlig sein muss. Die Angaben erfolgen als Dezimalbruch (z. B. &!K(9,25) bzw. &!K(9,50:9,25). Angaben zur Änderung des Zeilenabstands müssen ganzzahlig sein.

Eine Schriftgrößenumschaltung muss spätestens vor der nächsten Abschnittsgrenze rückgängig gemacht werden. Wenn Schriftgrößenumschaltungen innerhalb von Kolumnentiteln oder Marginalien vorgenommen werden sollen, so muss mit dem Parameter GRO als Schriftgröße für die Kolumnentitel bzw. Marginalien der für den Grundtext gewählte Schriftgrad angegeben werden. Alle Schriftgrößenumschaltungen müssen dann im Kolumnentitel bzw. in der Marginalie selbst angegeben werden; vor Zentrier- und Positionierbefehlen muss mit &!K{ auf die Grundschrift umgeschaltet werden, danach ggf. wieder auf die gewünschte Schrift.

Die Rückschaltung auf die für den aktuellen Textteil über Parameter gewählte Schriftgröße muss immer mit der Anweisung &!K{ erfolgen.

- &!K (nn/1)** Wie &!K (nn), gleichzeitig Einstellen des Zeilenabstands auf 1 Punkt.
- &!K (nn/+1)** Wie &!K (nn), gleichzeitig Vergrößern des Zeilenabstands um 1 Punkt.
- &!K (nn/-1)** Wie &!K (nn), gleichzeitig Verringern des Zeilenabstands um 1 Punkt.
- &!K (mm:nn)** Wie &!K (nn), jedoch mit veränderter Breite  
Es wird auf den Schriftgrad  $mm$  Punkt mit der Breite einer  $nn$ -Punkt-Schrift umgeschaltet. Die Größenangabe besteht aus zwei durch »:« getrennten Zahlen zwischen 4 und 99, von denen die erste die Kegelhöhe, die zweite die veränderte Geviertbreite angibt (z. B. &!K (10:9,5) für eine 10-Punkt-Schrift, modifiziert auf 9,5 Punkt Breite).
- &!K (mm:nn/1)** Wie &!K (mm:nn), gleichzeitig Einstellen des Zeilenabstands auf 1 Punkt.
- &!K (mm:nn/+1)** Wie &!K (mm:nn), gleichzeitig Vergrößern des Zeilenabstands um 1 Punkt.
- &!K (mm:nn/-1)** Wie &!K (mm:nn), gleichzeitig Verringern des Zeilenabstands um 1 Punkt.
- &!K (+n)** Umschalten auf um  $n$  Punkt größere Schrift  
 $n$  gibt an, um wieviel Punkt die neue Schriftgröße über der Schriftgröße liegen soll, die über Parameter für den Textteil gewählt wurde, in dem diese Anweisung steht. Angabe in Viertelpunkt-Schritten möglich wie bei &!K (nn)
- &!K (+n/1)** Wie &!K (+n), gleichzeitig Einstellen des Zeilenabstands auf 1 Punkt.
- &!K (+n/+1)** Wie &!K (+n), gleichzeitig Vergrößern des Zeilenabstands um 1 Punkt.
- &!K (+n/-1)** Wie &!K (+n), gleichzeitig Verringern des Zeilenabstands um 1 Punkt.
- &!K (-n)** Umschalten auf um  $n$  Punkt kleinere Schrift  
 $n$  gibt an, um wieviel Punkt die neue Schriftgröße unter der Schriftgröße liegen soll, die über Parameter für den Textteil gewählt wurde, in dem diese Anweisung steht. Angabe in Viertelpunkt-Schritten möglich wie bei &!K (nn)
- &!K (-n/1) &!K (-n/+1) &!K (-n/-1)**  
Wie &!K (-n), gleichzeitig Verändern des Zeilenabstands analog zu &!K (+n/1) &!K (+n/+1) &!K (+n/-1).
- &!K (+m:n)** Umschalten auf um  $m$  Punkt größere Schrift  
 $m$  gibt an, um wieviel Punkt die neue Schriftgröße über der Schrift-

größe liegen soll, die über Parameter für den Textteil gewählt wurde, in dem diese Anweisung steht.  $n$  gibt den Wert an, um den die Breite der Schrift gegenüber der über Parameter gewählten Angabe verändert werden soll. Hat  $n$  kein Vorzeichen, so wird das Vorzeichen von  $m$  übernommen.

**&!K(+m:n/l) &!K(+m:n/+1) &!K(+m:n/-1)**

Wie &!K(+m:n), gleichzeitig Verändern des Zeilenabstands analog zu &!K(+n/l) &!K(+n/+1) &!K(+n/-1).

**&!K(-m:n)** Umschalten auf um  $m$  Punkt kleinere Schrift

$m$  gibt an, um wieviel Punkt die neue Schriftgröße unter der Schriftgröße liegen soll, die über Parameter für den Textteil gewählt wurde, in dem diese Anweisung steht.  $n$  gibt den Wert an, um den die Breite der Schrift gegenüber der über Parameter gewählten Angabe verändert werden soll. Hat  $n$  kein Vorzeichen, so wird das Vorzeichen von  $m$  übernommen.

**&!K(-m:n/l) &!K(-m:n/+1) &!K(-m:n/-1)**

Wie &!K(-m:n), gleichzeitig Verändern des Zeilenabstands analog zu &!K(+n/l) &!K(+n/+1) &!K(+n/-1).

**&!K{** Rückschalten von Schriftgröße und Zeilenabstand

Es wird auf die Schriftgröße und den Zeilenabstand zurückgeschaltet, die über Parameter für den Textteil gewählt wurden, in dem diese Anweisung steht.

**&!K(0)** Wie &!K{

## 11.4. Wahl der Druckfarbe

**&!C(c, m, y, k)** Umschalten von schwarzem Druck auf Farbdruck

Die Farben werden im CMYK-System (Cyan, Magenta, Yellow, Black) definiert. Für  $c$ ,  $m$ ,  $y$  und  $k$  sind jeweils die (ganzzahligen) Prozentwerte des Anteils jeder dieser vier Druckfarben an der gewünschten Farbe anzugeben.

**&!C(Hnm)** Umschalten von schwarzem Druck auf Farbdruck

Die Farben werden als HKS-Farben angegeben. Dabei ist  $n$  = Farbnummer (1-97 mit einigen Lücken)

$m$  = Angabe der Papiersorte:

N = Naturpapier

K = Kunstdruckpapier

E = Endlospapier

Z = Zeitungspapier

Die HKS-Farben werden durch entsprechende CMYK-Farben dargestellt

**&!C(Hnm,proz)** Umschalten von schwarzem Druck auf Farbdruck

Die HKS-Farben werden als Schmuckfarben gesetzt; die Farbseparation wird vorbereitet. Für »proz« ist eine ganzzahliger Wert einzusetzen, der die Flächendeckung in Prozent angibt.

**&!C(g)** Umschalten von schwarzem Druck auf Graudruck

Die Graustufen werden als ganzzahlige Werte zwischen 0 = schwarz und 100 = weiß angegeben. Die Anweisung &!C(0) wirkt wie &!C{ (siehe weiter unten).

**&!C{** Zurückschalten auf Schwarz-Weiß-Druck

Beispiele für wichtige Farben:

&!C(20) dunkelgrau, &!C(80) hellgrau

&!C(0,100,100,0) rot (0% Cyan, 100% Magenta, 100% Yellow, 0% Black)

&!C(0,0,100,0) gelb

&!C(100,0,100,0) grün, &!C(60,0,100,0) hellgrün

&!C(100,100,0,0) blau, &!C(75,40,0,0) hellblau

&!C(25,80,100,0) braun

&!C(50,100,0,0) violett

&!C(0,65,100,0) orange

Es kann beliebig zwischen den Druckfarben hin- und hergeschaltet werden, ohne zuvor auf Schwarz-Weiß-Druck zurückzuschalten.

Jede andere Druckfarbe als Schwarz wird wie eine Auszeichnung behandelt, die am Ende eines Abschnitts aufgehoben wird. Die Anweisungen { bzw. #- heben den Farbdruck bzw. Graudruck nicht auf.

## 12. Akzente, diakritische Zeichen, Sonderzeichen

### 12.1. Akzente

Die Akzente können jedem Buchstaben zugeordnet werden (z. B. auch Kapitalchen). Den einzelnen Buchstaben können bis zu drei Akzente (bis zwei Akzente über dem Buchstaben, ein Akzent unter dem Buchstaben) zugeordnet werden. Den mit »# . « codierten Buchstaben können maximal zwei Akzente zugeordnet werden.

Die Akzent-Codierungen müssen vor den entsprechenden Buchstaben geschrieben werden. Bei mehr als einem Akzent für einen Buchstaben sind die Akzente in der Reihenfolge von oben nach unten anzugeben.

Frei stehende Akzente müssen vor festem Ausschluss (»\_«) codiert werden, also z. B.: %<\_ für frei stehenden Zirkumflex. Bei griechischen Großbuchstaben wird der Akzent automatisch vor statt über den Buchstaben gesetzt.

Über Kleinbuchstaben mit Oberlänge werden (außer bei Schriften, die im Umschaltbereich 3 liegen, vgl. Parameter SC#) die Versalakzente benutzt.

Über und unter besonders breiten Buchstaben (mit einer Dicke von mehr als 620 Bildlinien) wird der mit %- codierte Querstrich um 50% länger gesetzt als im jeweiligen Font vorgegeben. Mit der Anweisung &!% (nnn) kann angegeben werden, dass dies ab dem Dickenwert nnn statt des voreingestellten Wertes von 620 Bildlinien geschehen soll.

#### 12.1.1. Akzente über den Buchstaben

%"	ö	Doppelakut
%'	ó	Betonungszeichen
%(	ò	Bogen (unten offen)
%)	õ	Halbkreis (oben offen)
%*	ô	Kringel, Ringel(chen)
%+	õ	Plus
%,	ó	Komma, Apostroph
%-	ō	Querstrich, Balken
%.	ó	Punkt
%/	ó	Akut
%\	ò	Gravis
%:	ö	Trema
%<	ô	Zirkumflex, Dach
%>	ř	Haček, Haken

%=	ǫ	Kreuz
%?	ō	Tilde
%[	ō	Brücke
%]	ō	umgekehrte Brücke
%{	ó	Halbkreis rechts offen
%}	ó	Halbkreis links offen

### 12.1.2. Akzente unter den Buchstaben

%" "	ō	Doppelakut
%' '	o	Betonungszeichen
% ( (	o	Bogen, unten offen (auch für Griechisch)
%) )	o	Halbkreis, oben offen (auch für Griechisch)
%**	o	Kringel
%++	ó	Plus
%, ,	o	Komma, Sedila
%--	o	Querstrich, Balken (auch für Griechisch)
%..	o	Punkt (auch für Griechisch)
%//	o	Akut
%\ \	o	Gravis
%: :	o	Trema
%;	ç	Cedille
%; ;	o	Krummhaken, Ogonek
%<<	o	Zirkumflex
%^^	o	Zirkumflex

Die Codierung %^^ muss statt %<< dann verwendet werden, wenn mit dem Parameter MAC Makros der Form »<name>« angegeben sind.

%>>	o	Haček, Haken
%==	o	Kreuz
%??	o	Tilde
%[[	o	Brücke
%]]	o	umgekehrte Brücke
%{{	o	Halbkreis rechts offen



%} }	ø	Halbkreis links offen
%[-	ø	front sign, subscript
%-]	ø	back sign, subscript
%[ ]	ø	laminal sign, subscript
%-	ø	raising sign, subscript
% -	ø	lowering sign, subscript

## 12.2. Umlaute, Sonderbuchstaben, Ziffern, Ligaturen

Sind auf der Tastatur bzw. in der verwendeten Code-Page keine Umlaute vorgesehen, so können Umlaute und scharfes s mit folgender Eingabecodierung eingegeben werden:

^a	ä	a Umlaut
^o	ö	o Umlaut
^u	ü	u Umlaut
^s	ß	scharfes s
^A	Ä	A Umlaut
^O	Ö	O Umlaut
^U	Ü	U Umlaut
^S	ß	SS oder scharfes S

Fehlt der Parameter `vss` oder ist bei `vss` als Wert `-` angegeben, so wird `^s` in `SS` umgewandelt.

Ist der Parameter `vss` mit dem Wert `+` angegeben, für den betreffenden Font in der Dicktentabelle jedoch kein von 0 verschiedener Wert für dieses Zeichen ausgewiesen, so wird ein Fehlerkommentar in die Protokolldatei ausgegeben und das Zeichen als `SS` ausgegeben.

Die im folgenden – außer `i` ohne Punkt – aufgeführten, mit »#.« codierten Sonderbuchstaben sind nicht in allen Schriften verfügbar.

#.b	b	b mit Querstrich
#.d	đ	(serbokroat.) d mit Querstrich
#.D	Đ	(serbokroat.) D mit Querstrich
#.^d	ð	(isländisch, altenglisch) eth
#.^D	Ð	(isländisch, altenglisch) Eth
#.h	ħ	(maltesisch) h mit Querstrich
#.H	Ħ	(maltesisch) H mit Querstrich

#.i	ı	i ohne Punkt Für i mit Akzent über dem Buchstaben benutzt das Programm immer das i ohne Punkt; es kann also z. B. die Codierung %/i statt %/#.i für i ohne Punkt mit Akut verwendet werden.
#.ł	ł	(poln.) l mit Querstrich
#.Ł	Ł	(poln.) L mit Querstrich
#.ø	ø	(dän.-norw.) o mit Schrägstrich
#.Ø	Ø	(dän.-norw.) O mit Schrägstrich
#.þ	þ	(isländisch) Thorn, Kleinbuchstabe
#.Þ	Þ	(isländisch) Thorn, Großbuchstabe
#.ſ	ſ	langes s; in Fraktur-Fonts: rundes s
#.ß	ß	scharfes s
#.ˆS	ß	scharfes S
#.t̄	ṯ	(samisch) t mit Querstrich
#.T̄	Ṯ	(samisch) T mit Querstrich
#.z̄	ṣ	langes z, yogh
#.Z̄	Ṣ	Yogh
#.)	ʾ	Transkriptionszeichen für Alef
#.(	ʿ	Transkriptionszeichen für Ajin
#.%	ʔ	Transkriptionszeichen für Doppel-Alef

Die im folgenden aufgeführten Ligaturen sind nicht in allen Schriften verfügbar. Weitere Informationen dazu enthält die Beschreibung des Parameters SCH.

#.ˆa, #.ä	æ Ligatur æ
#.ˆo, #.ö	œ Ligatur œ
#.ˆA, #.Ä	Æ Ligatur Æ
#.ˆO, #.Ö	Œ Ligatur Œ
#.j	ij Ligatur ij
#.J	IJ Ligatur IJ
fˆˆf	Ligatur ff
fˆˆi	Ligatur fi
fˆˆl	Ligatur fl
fˆˆt	Ligatur ft

<code>f^f^i</code>	Ligatur ffi
<code>f^f^l</code>	Ligatur ffl
<code>c^h</code>	Ligatur ch
<code>c^k</code>	Ligatur ck
<code>l^l</code>	Ligatur ll (Fraktur)
<code>s^c^h</code>	Ligatur fch (Fraktur)
<code>s^i</code>	Ligatur fi (Fraktur)
<code>s^l</code>	Ligatur fl (Fraktur)
<code>s^s</code>	Ligatur ff (Fraktur)
<code>s^t</code>	Ligatur ft (Fraktur)
<code>t^t</code>	Ligatur tt (Fraktur)
<code>t^z</code>	Ligatur tz (Fraktur)

Das neben »#.« zur Codierung der Ligaturen verwendete Zeichen »^^« ist gleichzeitig das Zeichen für Trennverbot. Soll bei einer so codierten Ligatur kein Trennverbot ausgesprochen werden, so lautet die Codierung `f^\f`, `f^i` usw.

In der Fraktur-Schrift sind die Ligaturen ch, ck und tz sogenannte Zwangsligaturen; diese werden auch in gesperrten Wörtern nicht aufgelöst, sondern wie eigene Buchstaben behandelt.

Mit dem Parameter LAU (2. Wert) kann verlangt werden, dass ^^ auch nach `f` und `c` nicht als Steueranweisung für Ligaturen, sondern nur als Trennverbot gelten soll. Soll – trotz Parameterangabe zur Ausführung der Ligaturen – an bestimmten Stellen nur Trennverbot zwischen den oben genannten Zeichen verlangt werden, ohne dass die betroffenen Zeichen als Ligaturen gesetzt werden sollen, so ist die Codierung `f^^^f` usw. zu wählen.

**#.1** Alternativziffern (nur in Schriften, für die sogenannte Expert Fonts verfügbar sind; vgl. das Standard-Makro \*PSFONT).

Bei einigen Schriften stehen neben den Tabellenziffern auch Mediaevalziffern zur Verfügung. Mit dem Parameter SCH kann angegeben werden, welche dieser Ziffern als Standard-Ziffernsatz gelten sollen, die mit normalen Ziffern codiert werden. Mit Hilfe der Codierung #.1 etc. werden die entsprechenden Ziffern aus dem alternativen Ziffernsatz angesprochen (wurden beim Start des Satzprogramms die Tabellenziffern als Standard-Ziffernsatz gewählt, dann sind die Alternativziffern die Mediaevalziffern und umgekehrt). Vgl. auch die Anweisungen #X+ und #X- in Kapitel 11.1.1.

## 12.3. Hochgestellte Buchstaben und Ziffern

Die entsprechenden Zeichen werden, wenn nichts anderes angegeben, in kleinerem Schriftgrad hochgestellt. Größe: 1 Punkt mehr als die Hälfte der zuletzt verwendeten Schriftgröße; Hochstellung: ca. 3/8 der zuletzt verwendeten Schriftgröße.

### 12.3.1. Hochgestellte Einzelzeichen

- #' x** Hochgestelltes Zeichen, direkt neben dem vorhergehenden  
Für x kann jedes beliebige Zeichen stehen, das durch einen einzelnen Code angegeben werden kann (also nicht durch »#.« oder durch Makro codierte Zeichen oder Akzentbuchstaben). Bei mehreren hochgestellten Zeichen ist das »#'« zu wiederholen: 2#' n#' d. Sperrung hochgestellter Zeichen ist nicht vorgesehen.
- ^1** Wie #' 1 (für die Ziffern 0 bis 9)
- %1** Hochgestellte Ziffer als Fußnotenziffer  
Bei mehrstelligen Fußnotennummern ist das »%« vor jeder Ziffer zu wiederholen, z. B. %2%1%8.  
%0 gibt \*, %0%0 gibt \*\* als Fußnotenverweis bzw. Kennung.
- %a** Hochgestellter Kleinbuchstabe in Fußnotennummern

### 12.3.2. Hochgestellte Bereiche mit Größenänderung

In hochgestellten Bereichen können Einzelzeichen zusätzlich durch #' hochgestellt bzw. durch #; übersetzt und durch #, tiefgestellt werden; die damit verbundene Hoch- bzw. Tiefstellung bezieht sich dann auf den hochgestellten Bereich. Auch die Verschiebungen mit den Anweisungen &! (:nn) bzw. &! (;nn) (vgl. Kapitel 22) werden relativ zum bereits hochgestellten Bereich vorgenommen.

Die Hochstellung muss vor dem nächsten Leerzeichen rückgängig gemacht werden, damit Zentrierungen und Randausgleich korrekt ausgeführt werden können.

In hochgestellten Bereichen darf keine Merkstelle definiert und auf keine Merkstelle positioniert werden, es sei denn, dass sich diese Merkstellen ausschließlich auf die hochgestellte Zeichenfolge beziehen.

- &!H+** Hochstellung Anfang für alle nachfolgenden Zeichen bis &!H.  
Die hochgestellten Zeichen haben die gleiche Größe und Hochstellung wie die mit #' x codierten Zeichen.
- &!G+** Wie &!H+, aber: Größe der hochgestellten Zeichen (ab 8 Punkt Grundschrift) 75% der zuletzt verwendeten Schriftgröße, Hochstellung: ca. 1/4 der zuletzt verwendeten Schriftgröße
- &!H.** Ende von Hoch- und Tiefstellung
- &!G.** Wie &!H.

### 12.3.3. Hochgestellte Bereiche ohne Größenänderung

In hochgestellten Bereichen können Einzelzeichen zusätzlich durch #' hochgestellt bzw. durch #; übersetzt und durch #, tiefgestellt werden; die damit verbundene Hoch- bzw. Tiefstellung bezieht sich dann auf den hochgestellten Bereich. Auch die Verschiebungen mit den Anweisungen &! (:nn) bzw. &! (;nn) (vgl. Kapitel 22) werden relativ zum bereits hochgestellten Bereich vorgenommen.

- #H:** Hochstellung (ohne Größenänderung) um 1/3 Zeile für alle nachfolgenden Zeichen bis #G:
- #H(n)** Hochstellung (ohne Größenänderung) um n/2 Punkt für alle nachfolgenden Zeichen bis #G:
- #H(:n)** Hochstellung (ohne Größenänderung) für alle nachfolgenden Zeichen bis #G: um die Zahl von Halbpunktschritten, die sich aus n/32 der Gevierthöhe errechnet
- #O:** Hochstellung (ohne Größenänderung) um 1/2 Zeile für alle nachfolgenden Zeichen bis #G:
- #O(n)** Hochstellung (ohne Größenänderung) um n/2 Punkt für alle nachfolgenden Zeichen bis #G:
- #O(:n)** Hochstellung (ohne Größenänderung) für alle nachfolgenden Zeichen bis #G: um die Zahl von Halbpunktschritten, die sich aus n/32 der Gevierthöhe errechnet
- #G:** Ende von Hoch- und Tiefstellung
- #G(n)** Ende von Hoch- und Tiefstellung (n ist eine beliebige Zahl)
- #H: #O: #G: #H(n) #O(n) #G(n) beendet alle noch aktiven Hoch- bzw. Tiefstellungen, auch die Verschiebungen durch &! (:nn) bzw. &! (;nn), vgl. Kapitel 22.

### 12.4. Übergesetzte Zeichen

Das angegebene Zeichen wird in kleinerem Schriftgrad über den nachfolgenden Buchstaben gesetzt. Größe: 1 Punkt mehr als die Hälfte der zuletzt verwendeten Schriftgröße; Hochstellung: 1/2 der zuletzt verwendeten Schriftgröße (+ 1 Punkt bei Schriften, die auf dem Parameter GRO mit % gekennzeichnet sind); über Versalien und Kleinbuchstaben mit Oberlänge (außer zwischen »++« und »{«): 3/4 der zuletzt verwendeten Schriftgröße.

- #;x** Über den nachfolgenden Buchstaben gesetztes Zeichen (Beispiel: #;eu ist ũ)

Vgl. Erläuterung zu 12.1

## 12.5. Tiefgestellte Zeichen

Die entsprechenden Zeichen werden, wenn nichts anderes angegeben, in kleinerem Schriftgrad tiefgestellt. Größe: 1 Punkt mehr als die Hälfte der zuletzt verwendeten Schriftgröße; Tiefstellung:  $1/2$  Punkt +  $1/5$  der zuletzt verwendeten Schriftgröße.

### 12.5.1. Tiefgestellte Einzelzeichen

- #, x** tiefgestelltes Zeichen, direkt neben dem vorhergehenden Zeichen  
Für x kann jedes beliebige Zeichen (außer den durch »#.« oder durch Makro codierten Zeichen) stehen. #, 1 ist tiefgestellte Ziffer in gleicher Höhe und Größe wie die übrigen tiefgestellten Zeichen.
- #-x** Zeichen in gleicher Größe wie ein mit #, x tiefgestelltes Zeichen, aber auf der Schriftgrundlinie direkt neben dem vorhergehenden Zeichen  
Für x kann jedes beliebige Zeichen (außer den durch »#.« oder durch Makro codierten Zeichen) stehen.

### 12.5.2. Tiefgestellte Bereiche mit Größenänderung

In tiefgestellten Bereichen können Einzelzeichen zusätzlich durch #' hochgestellt bzw. durch #; übersetzt und durch #, tiefgestellt werden; die damit verbundene Hoch- bzw. Tiefstellung bezieht sich dann auf den tiefgestellten Bereich. Auch die Verschiebungen mit den Anweisungen &! (:nn) bzw. &! (;nn) (vgl. Kapitel 22) werden relativ zum bereits tiefgestellten Bereich vorgenommen.

Die Tiefstellung muss vor dem nächsten Leerzeichen rückgängig gemacht werden, damit Zentrierungen und Randausgleich korrekt ausgeführt werden können.

In tiefgestellten Bereichen darf keine Merkstelle definiert und auf keine Merkstelle positioniert werden, es sei denn, dass sich diese Merkstellen ausschließlich auf die tiefgestellte Zeichenfolge beziehen.

- &!H-** Tiefstellung für alle nachfolgenden Zeichen; Größe und Tiefstellung wie bei #, x
- &!G-** Wie &!H-, aber Größe der Zeichen wie bei &!G+
- &!H=** Zeichen in der gleichen Schriftgröße wie bei &!H-, aber auf der Schriftgrundlinie
- &!G=** Zeichen in der gleichen Schriftgröße wie bei &!G-, aber auf der Schriftgrundlinie
- &!H.** Ende von Hoch- und Tiefstellung
- &!G.** Wie &!H.

### 12.5.3. Tiefgestellte Bereiche ohne Größenänderung

In tiefgestellten Bereichen können Einzelzeichen zusätzlich durch #' hochgestellt bzw. durch #; übersetzt und durch #, tiefgestellt werden; die damit angegebene Hoch- bzw. Tiefstellung bezieht sich dann auf den tiefgestellten Bereich. Auch die Verschiebungen mit den Anweisungen &! (:nn) bzw. &! (;nn) (vgl. Kapitel 22) werden relativ zum bereits tiefgestellten Bereich vorgenommen.

- #T:** Tiefstellung (ohne Größenänderung) um 1/3 Zeile für alle nachfolgenden Zeichen bis #G:
- #T(n)** Tiefstellung (ohne Größenänderung) um  $n/2$  Punkt für alle nachfolgenden Zeichen bis #G:
- #T(;n)** Tiefstellung (ohne Größenänderung) für alle nachfolgenden Zeichen bis #G: um die Zahl von Halbpunktschritten, die sich aus  $n/32$  der Gevierthöhe errechnet
- #U:** Tiefstellung (ohne Größenänderung) um 1/2 Zeile für alle nachfolgenden Zeichen bis #G:
- #U(n)** Tiefstellung (ohne Größenänderung) um  $n/2$  Punkt für alle nachfolgenden Zeichen bis #G:
- #U(;n)** Tiefstellung (ohne Größenänderung) für alle nachfolgenden Zeichen bis #G: um die Zahl von Halbpunktschritten, die sich aus  $n/32$  der Gevierthöhe errechnet
- #G:** Ende von Hoch- und Tiefstellung
- #G(n)** Ende von Hoch- und Tiefstellung (n ist eine beliebige Zahl)
- #T: #U: #G: #T(n) #U(n) #G(n) beendet alle sonst noch aktiven Hoch- bzw. Tiefstellungen (auch die Verschiebungen durch &! (:nn) bzw. &! (;nn), vgl. Kapitel 22).

### 12.6. Untergesetzte Zeichen

Das angegebene Zeichen wird in kleinerem Schriftgrad unter den nachfolgenden Buchstaben gesetzt. Größe: 1 Punkt mehr als die Hälfte der zuletzt verwendeten Schriftgröße; Punktzahl Tiefstellung: 1/2 der zuletzt verwendeten Schriftgröße.

- #!x** unter den nachfolgenden Buchstaben gesetztes Zeichen (Beispiel: #!eu ist u)
- Vgl. Erläuterung zu 12.1

### 12.7. Satzzeichen, Sonderzeichen

- . . Punkt

,	,	Komma
:	:	Doppelpunkt
;	;	Strichpunkt
?	?	Fragezeichen
^?	¿	umgekehrtes Fragezeichen (spanisch)
!	!	Ausrufezeichen
^!	¡	umgekehrtes Ausrufezeichen (spanisch)
(	(	runde Klammer auf
)	)	runde Klammer zu
[	[	eckige Klammer auf
]	]	eckige Klammer zu
^<	<	spitze Klammer auf
^>	>	spitze Klammer zu
		Die Codierungen ^< und ^> müssen für die spitzen Klammern dann verwendet werden, wenn mit einem der Parameter MAC oder MAH oder MAA Makros der Form »<name>« angegeben sind. Werden keine Makros dieser Form benutzt, können spitze Klammern auch mit < und > codiert werden.
'	'	Apostroph
^'	‘	umgekehrter Apostroph
#.>	»	doppeltes französisches Anführungszeichen, Spitze nach rechts
#.<	«	doppeltes französisches Anführungszeichen, Spitze nach links
"	»...«	Anführungs- und Schlusszeichen; sie werden vom Programm selbständig unterschieden
		Standardbelegung bei Fehlen des Parameters AFZ oder bei der Angabe »0« im Parameter AFZ: sogenannte französische Anführungs- und Schlusszeichen, guillemets mit Spitze nach innen (z. B. für Deutsch) »...«
	«...»	Standardbelegung bei Angabe »1« im Parameter AFZ (z. B. für Französisch)
	»...»	Standardbelegung bei Angabe »2« im Parameter AFZ (z. B. für Finnisch und Schwedisch)
	„...“	Standardbelegung bei Angabe »-1« im Parameter AFZ (z. B. für Deutsch, Polnisch, Ungarisch)
	“...”	Standardbelegung bei der Angabe »-2« im Parameter AFZ (z. B. für Englisch, Französisch, Spanisch)



“...”	Standardbelegung bei der Angabe »-3« im Parameter AFZ (z. B. für Italienisch, Türkisch)
„...”	Standardbelegung bei der Angabe »-4« im Parameter AFZ (z. B. für Niederländisch, Polnisch, Ungarisch)
”...”	Standardbelegung bei der Angabe »-5« im Parameter AFZ (z. B. für Finnisch, Niederländisch, Schwedisch)
#.:	> einfaches französisches Anführungszeichen, Spitze nach rechts
#.;	< einfaches französisches Anführungszeichen, Spitze nach links
#.?	>...< einfache Anführungs- und Schlusszeichen; sie werden vom Programm selbständig unterschieden
	Zur Form der Anführungszeichen in Abhängigkeit von den Angaben im Parameter AFZ vgl. die Erläuterung zu ”.
#.,	„ doppeltes Anführungszeichen unten (»Doppelkomma«)
#.’	“ doppeltes Anführungszeichen oben (doppelter umgekehrter Apostroph)
#.”	” doppeltes Anführungszeichen oben (doppelter Apostroph)
#.[	⌈ Winkel links oben
#.]	⌋ Winkel rechts oben
-	- Gedankenstrich (= Halbgeviertstrich zwischen Leerzeichen), »bis« (Halbgeviertstrich) und Divis (1/4 Geviert) werden vom Programm unterschieden: <ul style="list-style-type: none"> <li>- allein (zwischen Leerzeichen): Halbgeviertstrich</li> <li>- \ Divis</li> <li>- \- Halbgeviertstrich</li> <li>-- zwei Halbgeviertstriche; arabisch: Kashida</li> <li>- vor Ziffer: Halbgeviertstrich</li> <li>- nach Buchstabe und Abführungszeichen: Divis</li> <li>- zwischen Ziffer und Buchstabe: Divis</li> <li>- nach Ziffer bzw. Komma: Halbgeviertstrich</li> <li>- nach Punkt und Auszeichnungs-Ende: wenn davor Buchstabe, Divis; sonst Halbgeviertstrich</li> <li>- in allen anderen Fällen vor Spatium: Halbgeviertstrich</li> <li>- vor Buchstabe: Divis</li> <li>- vor Komma: wenn Buchstabe folgt, Divis; sonst Halbgeviertstrich</li> <li>- vor anderen Zeichen: Halbgeviertstrich.</li> </ul>

Für PostScript-Schriften gilt: Divis = hyphen, Halbgeviertstrich = n-dash; die exakte Länge ist von der jeweiligen Schrift abhängig.

Vgl. auch die Erläuterung zu 9. (Silbentrennung)

#.\	∕	Fraktur-Divis (schräges Gleichheitszeichen) (wird bei Divis in Fraktur-Schriften automatisch gewählt)
/	/	Schrägstrich
		senkrechter Strich (vgl. 12.11.)
^		wie
#. .	→	Verweisfeil nach rechts
#./		doppelter senkrechter Strich
*	*	Sternchen, hochgestellt
^+	†	Sterbekreuz
^%	%	Prozentzeichen
^&	&	Et-Zeichen
#.!	§	Paragraphzeichen
^\$	\$	Dollarzeichen
–		Fester Ausschluss (Viertelgeviert)
		An dieser Stelle findet keine Zeilentrennung statt, wenn es nicht über Parameter verlangt wird.
		Kommt ein frei stehender fester Ausschluss beim Zeilenumbruch ans Zeilenende oder an den Zeilenanfang zu stehen, so wird er zusammen mit dem davor bzw. dahinter stehendem Leerzeichen unterdrückt.
#. _		Spatium im Wort
		An diesem Spatium findet keine Zeilentrennung statt; es wird jedoch wie eine normales Spatium zum Zweck des Zeilenausgleichs bei Bedarf in der Breite verringert oder vergrößert.
&!B!		das unmittelbar vor &!B! stehende Spatium wird fixiert; es steht zum Austreiben nicht zur Verfügung
&!B(-)		Die Spatien zwischen &!B(-) und &!B(+) werden nicht zum Austreiben bzw. Zusammenschieben der Zeilen auf die aktuelle Zeilenbreite herangezogen.
&!B(+)		Ende für die Fixierung der Spatien durch &!B(-). Am Abschnittsende wird ein nach &!B(-) noch fehlendes &!B(+) automatisch ergänzt.
		Stehen die Steueranweisungen &!B(-) oder &!B(+) zwischen Leerzeichen, so wird das Leerzeichen hinter der Steueranweisung bei der Auswertung ignoriert.
^_	_	Halbgeviertstrich auf Geviertunterkante

+	+	plus
		Sollen zwei Pluszeichen unmittelbar hintereinander gesetzt werden, so müssen diese durch »^« (als ^^ eingegeben) getrennt werden, da »++« Steueranweisung für Kapitalchen ist.
^-	-	minus
=	=	gleich
^*	×	»mal«-Kreuz
^.	.	»mal«-Punkt
#.*	°	Grad
#.-	'	Minute
#.=	"	Sekunde
^"	■	Blockade, Halbgeviert
^=	■	Blockade, Halbgeviert, mit Fehlerkommentar
^#	#	Nummernzeichen
^\	\	umgekehrter Schrägstrich
^@	@	Klammeraffe
#.{	{	geschweifte Klammer auf
#.}	}	geschweifte Klammer zu
#.^^	^	Zirkumflex, Dach (nicht als Akzent)
^&amp;	&	ET-Zeichen (bei Parameter LAU, 3. Wert=1)
		Soll die Zeichenfolge & gesetzt werden und ist als 3. Wert zu LAU eine 1 angegeben, so muss diese als ^& ; codiert werden.
^&lt;	<	(bei Parameter LAU, 3. Wert=1)
^&gt;	>	(bei Parameter LAU, 3. Wert=1)
^&apos;	'	(bei Parameter LAU, 3. Wert=1)
^&quot;	"	(senkrecht stehend / Zeichen oktal 42 aus aktuellem PS-Font; bei Parameter LAU, 3. Wert=1)

## 12.8. Sonstige Zeichen und Symbole

### 12.8.1. Mit #(name) codierte Sonderzeichen

Die in der Beschreibung »TUSTEP-Grundlagen« im Kapitel »Zeichenvorrat« beschriebenen, mit der Steueranweisung #(name) codierten Sonderzeichen stehen mit Ausnahme der als »groß« bezeichneten Zeichen, der Zeichen #(MC) #(MN) #(MQ) #(MR)

#(MZ) und der Zeichen für »Box« auch im Satzprogramm zur Verfügung. Über die dort aufgeführten Zeichen hinaus stehen im Satzprogramm folgende Metrik-Zeichen zur Verfügung:

#(mlks)	◊	Metrik: lange oder kurze Silbe
#(mklks)	≈	Metrik: zwei kurze oder eine lange Silbe
#(mlkks)	∞	Metrik: eine lange oder zwei kurze Silben
#(mas)	×	Metrik: anceps (lange oder kurze Silbe)
#(maas)	∞∞	Metrik: zwei ancipita
#(mfs)	^	Metrik: fehlendes Element

Dem mit #(E) codierten Euro-Zeichen liegt der offizielle Entwurf der Europäischen Zentralbank zugrunde.

Viele Fonts enthalten typographisch angepasste Euro-Zeichen. Auch wenn dort dem Euro-Zeichen keine Adresse im Font zugeordnet ist, kann es über die Steueranweisung &! (##/001) (siehe unten bei 12.8.3) angesprochen werden, da das Makro \*PSAUS dem Zeichen, falls es in einem Font vorhanden ist, die Zeichenummer 001 zuordnet. Das Euro-Zeichen kann außerdem über die Steueranweisung &! (#Ummmmm\zeichename\dickte) (siehe unten bei 12.8.4) direkt über seinen Namen angesteuert werden.

## 12.8.2. Römische Zahlen aus arabischen Zahlen

&! (r+)	Römische Ziffern (Gemeine) Anfang
&! (r-)	Römische Ziffern (Gemeine) Ende
	&! (r+) nnn&! (r-) gibt die in arabischen Ziffern geschriebene Zahl nnn in römischen Ziffern (Kleinbuchstaben) aus.
&! (R+)	Römische Ziffern (Versalien) Anfang
&! (R-)	Römische Ziffern (Versalien) Ende
	&! (R+) nnn&! (R-) gibt die in arabischen Ziffern geschriebene Zahl nnn in römischen Ziffern (Großbuchstaben) aus.

## 12.8.3. Zeichenadresse aus dem Encoding von Type-1-Fonts

&! (##mmmm/nnn) PostScript-Sonderzeichen über Zeichenummer

Das Zeichen mit der (dreistellig zu schreibenden) oktalen Zeichenummer nnn aus der PostScript-Schrift mit der fünfstelligen TUS-TEP-Schriftnummer mmmmm soll aufgerufen werden. Seine Dicke wird automatisch mit berücksichtigt und ggf. auf die vor der schließenden Klammer durch +d bzw. +-d angegebenen Dicktenwerte (vor der evtl. Multiplikation mit dem durch \*g angegebenen Faktor) aufaddiert; siehe dazu unten in Kapitel 22. Handelt es sich bei dem Zeichen um einen Buchstaben, so kann es zweckmäßig sein,

vor dem ## in der Klammer die Zeichenfolge @b zur Kennzeichnung dieser »hardware-nahen Codierung« als Buchstabe einzufügen. Vgl dazu die Beschreibung von &! (@b) in Kapitel 22.

**&! (##/nnn)** PostScript-Sonderzeichen über Zeichennummer

Wie &! (######/nnn), jedoch Zeichen mit der oktalen Nummer nn aus dem aktuellen Font.

## 12.8.4. Zeichenadressierung über Zeichenamen

**&! (#U#####\zeichename\dicke)** PostScript-Sonderzeichen über Zeichenname

Das Zeichen mit dem Namen *zeichename* aus der PostScript-Schrift mit der fünfstelligen TUSTEP-Schriftnummer ##### soll aufgerufen werden. Wird für ##### 0 angegeben oder wird ##### weggelassen, so wird das Zeichen *zeichename* aus dem aktuellen Font aufgerufen. Der Wert für *dicke* kann der afm-Datei des Fonts entnommen werden (vgl. auch das Standard-Makro \*PSGLYPHS) und muss hier mit angegeben werden.

## 12.8.5. Unicode-Zeichen 2000 bis 202F (»General Punctuation«)

Von #UMWANDLE werden nicht alle Unicode-Zeichen zwischen #[2000] und #[202F] in TUSTEP-Zeichencodes umcodiert, sondern mit #[XXXX] wiedergegeben. Das Satzprogramm akzeptiert aus diesem Bereich folgende Codierungen:

#[2000]	( <i>en quad</i> )	Zwischenraum 500 Bildlinien
#[2001]	( <i>em quad</i> )	Zwischenraum 1000 Bildlinien
#[2002]	( <i>en space</i> )	Zwischenraum 500 Bildlinien
#[2003]	( <i>em space</i> )	Zwischenraum 1000 Bildlinien
#[2004]	( <i>three-per-m-space</i> )	Zwischenraum 333 Bildlinien
#[2005]	( <i>four-per-m-space</i> )	Zwischenraum 250 Bildlinien
#[2006]	( <i>six-per-m-space</i> )	Zwischenraum 167 Bildlinien
#[2007]	( <i>figure space</i> )	Zwischenraum 500 Bildlinien
#[2008]	( <i>punctuation space</i> )	Zwischenraum 500 Bildlinien
#[2009]	( <i>thin space</i> )	Zwischenraum 200 Bildlinien
#[200A]	( <i>hair space</i> )	Zwischenraum 100 Bildlinien
#[2010]	( <i>hyphen</i> )	- (Divis)
#[2011]	( <i>non-breaking hyphen</i> )	- (Divis)
#[2012]	( <i>figure dash</i> )	- (Halbgeviertstrich)
#[2013]	( <i>en dash</i> )	- (Gedankenstrich, wie \-)
#[2014]	( <i>em dash</i> )	— (Geviertstrich)
#[2015]	( <i>horizontal bar</i> )	- (Halbgeviertstrich)
#[2016]	( <i>double vertical line</i> )	
#[2018]	( <i>left single quotation mark</i> )	‘
#[2019]	( <i>right single quotation mark</i> )	’
#[201A]	( <i>single low-9 quotation mark</i> )	,

# [201B]	(single high-reversed-9 quotation mark)	‘
# [201C]	(left double quotation mark)	“
# [201D]	(right double quotation mark)	”
# [201E]	(double low comma quotation mark)	„
# [201F]	(double high-reversed-9 quotation mark)	“
# [2020]	(dagger)	†
# [2021]	(double dagger)	‡
# [2022]	(bullet)	•
# [2026]	(horizontal ellipsis)	...
# [202F]	(narrow no-break space)	Zwischenraum 125 Bildlinien

Die übrigen Zeichen aus diesem Bereich werden entweder übergangen (# [200B] bis # [200F] und # [2028] bis # [202E]) oder mit Blockade zurückgewiesen.

Ist bei Parameter LAU als dritter Wert 1 angegeben, so können diese Zeichen auch als hexadezimale Character Entities  $\text{\^{\&}\^{\#x2000}}$ ; bis  $\text{\^{\&}\^{\#x202F}}$ ; codiert sein; darüber hinaus werden die Character Entities  $\text{\^{\&}\^{\#x017F}}$ ; wie #.s (l) und  $\text{\^{\&}\^{\#x0292}}$ ; wie #.z (z) interpretiert,  $\text{\^{\&}\^{\#x00A0}}$ ; (non-breaking space) wird wie #.\_ behandelt.

## 12.9. PostScript-Grafiken

### &! (#Giii) PostScript-Grafik

Bei der Ausgabe mit dem Standard-Makro \*PSAUS wird an dieser Stelle die Abbildung Nummer *iii* eingebunden. Folgen auf *iii* weitere Angaben in der Klammer, so muss die Nummer *iii* durch \ abgeschlossen werden, wenn das nächste Zeichen hinter *iii* eine Ziffer ist.

Die Abbildung muss im EPS-Format (Encapsulated PostScript) vorliegen. Sie wird vertikal relativ zur Schriftgrundlinie positioniert, auf der ein an dieser Stelle gesetztes Zeichen stehen würde.

Solche Abbildungen müssen vor der Ausgabe mit dem Standard-Makro \*GRAFIK zum Einmontieren vorbereitet und mit einer Nummer zwischen 1 und 999999 versehen in einer Datei gesammelt werden. Diese Datei muss später beim Aufruf von \*PSAUS zur Spezifikation GRAFIK angegeben werden.

### &! (#Gn/iii) PostScript-Grafik

Wie &! (#Giii); dabei ist *n* die Nummer der Grafikdatei, aus der die Abbildung ausgewählt werden soll, und gibt an, als wievielte die Grafikdatei später beim Aufruf von \*PSAUS angegeben wird.

### &! (#Gn/Hiii) PostScript-Grafik

Wie &! (#Gn/iii), jedoch Angabe der Nummer der Abbildung als 4-stellige Hexadezimalzahl.

### &! (#Siii), &! (#Sn/iii) PostScript-Grafik, skaliert

Wie &! (#Giii) bzw. &! (#Gn/iii), jedoch wird die Abbildung

der augenblicklich eingestellten Schriftgröße angepasst. Dabei wird davon ausgegangen, dass die Abbildung für die Einbindung in eine Umgebung mit 12 Punkt Kegelgröße vorbereitet wurde.

**&! (#Sn/Hiiii)** PostScript-Grafik, skaliert

Wie &! (#Sn/iii), jedoch Angabe der Nummer der Abbildung als 4-stellige Hexadezimalzahl.

## 12.10. Spiegeln und Drehen von Zeichen

Es können nur Einzelzeichen (auch solche, die mit #(name) oder mit &! (#mmmmmm/nnn) codiert sind) gespiegelt werden. Bei Buchstaben mit Akzenten wird nur der Buchstabe selbst gespiegelt; der Akzent wird unverändert ausgegeben.

Über- oder untergesetzte Zeichen können nicht gespiegelt werden. Sollen hoch- oder tiefgestellte Zeichen gespiegelt werden, so müssen diese vor dem Spiegeln mit einer der Anweisungen zum Hoch- bzw. Tiefstellen von Bereichen (siehe unter 12.3.2 und 12.3.3) hoch- bzw. tiefgestellt werden.

**&!|x** Vertikal spiegeln

Das nach dem »|« angegebene Zeichen wird spiegelverkehrt (um die vertikale Achse gespiegelt) ausgegeben.

**&!-x** Horizontal spiegeln, auf Grundlinie stellen

Das nach dem »-« angegebene Zeichen wird um die horizontale Achse gespiegelt ausgegeben. Das gespiegelte Zeichen steht auf der Schriftgrundlinie.

**&! =x** Horizontal spiegeln

Das nach dem »=« angegebene Zeichen wird um die horizontale Achse gespiegelt ausgegeben. Das gespiegelte ausgegebene Zeichen nimmt vertikal die gleiche Position ein wie das ungespiegelte Zeichen.

**&! /x** Drehen um 180 Grad, auf Grundlinie stellen

Das nach dem »/« angegebene Zeichen wird um 180 Grad gedreht. Das gedrehte Zeichen steht auf der Schriftgrundlinie.

**&! \x** Drehen um 180 Grad

Das nach dem »\« angegebene Zeichen wird um 180 Grad gedreht. Das gedrehte Zeichen nimmt vertikal die gleiche Position ein wie das ungedrehte Zeichen.

## 12.11. Linien, Punktreihen

Siehe auch @--0, @.0 usw. (10.2.), Unterstreichung, Unterpunktierung (11.2.)

| senkrechter Strich auf Gevierthöhe (Dicke:  $1/25$  Geviert, ohne Vor/Nachbreite)

Längere senkrechte Linien können auch aus | zusammengesetzt werden, wenn ohne zusätzlichen Durchschuss gesetzt wird und wenn der Spaltenhöhenausgleich ausgeschaltet ist.

^| wie |

**&! | (a, l, n)** Senkrechte Linie

Senkrechte Linie von aktueller horizontaler Position und der vertikalen Position  $a$  mit der Länge  $l$  Punkt und der Strichstärke  $n/8$  Punkt. Der Schreibstrahl wird um die angegebene Strichstärke nach rechts verschoben; seine vertikale Position bleibt unverändert.

$a$  kann sein:

- Zahlenangabe, ggf. mit Dezimalstellen für Achtelpunkt-Angabe (Dezimalpunkt, kein Komma): Abstand vom oberen Satzspiegelrand
- $*$  für die an dieser Stelle erreichte vertikale Position (Schriftgrundlinie)
- $*+n$  für  $n$  Punkt unterhalb der an dieser Stelle erreichten vertikalen Position. Es ist eine Verschiebung von maximal 31.5 Punkt möglich (Dezimalpunkt, kein Komma bei nicht ganzzahliger Punktzahl)
- $*-n$  für  $n$  Punkt oberhalb der an dieser Stelle erreichten vertikalen Position. Es ist eine Verschiebung von maximal 31.5 Punkt möglich (Dezimalpunkt, kein Komma bei nicht ganzzahliger Punktzahl)

Die Länge  $l$  der Linie wird positiv (für Linie nach unten) oder negativ (für Linie nach oben) in Punkt angegeben, ggf. mit Dezimalstellen für Achtelpunkt-Angabe.

**&! ? (a, l, n, wa, wl)** Senkrechte Wellenlinie

Wie **&! | (a, l, n)**, jedoch Wellenlinie mit der Amplitude  $w_a/8$  Punkt und der (halben) Wellenlänge  $w_l/8$  Punkt. Die Angabe  $w_a$  bezeichnet nicht die maximale Auslenkung, sondern den Abstand der Bézier-Kontrollpunkte von der von  $a$  ausgehenden senkrechten Linie. Die sichtbare Auslenkung beträgt ca. 40% des zu  $w_a$  angegebenen Wertes.

**&! \_ (a, l, n, wa, wl)** Waagerechte Wellenlinie

Wie **&! ? (a, l, n, wa, wl)**, jedoch nicht senkrecht nach unten, sondern waagrecht nach links

**&! / (a, b, n)** Linie zwischen gemerkten Punkten

Linie zwischen den (gemerkten) Punkten  $a$  und  $b$  mit der Strichstärke  $n/8$  Punkt. Die Position des Schreibstrahls wird nicht verändert.



a kann sein:

- \* für die an dieser Stelle erreichte vertikale Position (Schriftgrundlinie)
- \*+n für n Punkt unterhalb der an dieser Stelle erreichten vertikalen Position. Es ist eine Verschiebung von maximal 31.5 Punkt möglich (Dezimalpunkt, kein Komma, bei nicht ganzzahliger Punktzahl; Vielfache von Achtelpunkt angebar).
- \*-n für n Punkt oberhalb der an dieser Stelle erreichten vertikalen Position. Es ist eine Verschiebung von maximal 31.5 Punkt möglich (Dezimalpunkt, kein Komma, bei nicht ganzzahliger Punktzahl; Vielfache von Achtelpunkt angebar).
- $\forall n$  für: Angabe einer mit  $\&!M(V_n)$  gemerkten Position
- n für einen Abstand von n Punkt vom oberen Satzspiegelrand. n kann als Dezimalbruch für Vielfache von Achtelpunkt (mit Punkt statt Komma hinter der Einerstelle, z. B. 8.125) angegeben werden.

b ist immer die Angabe einer mit  $\&!M(V_m)$  gemerkten Position m in der Form  $V_m$ . Die Punkte, zwischen denen Linien gezogen werden sollen, müssen zuvor auf der selben Seite (bei mehrspaltigem Satz in der selben Spalte) definiert worden sein.

n gibt die Strichstärke in 1/8 Punkt an. Soll statt einer durchgezogenen Linie eine gestrichelte Linie gezogen werden, so kann  $n:ls:lw:loff$  statt n angegeben werden. Die Linie besteht dann aus jeweils  $ls$  schwarzen und  $lw$  weißen aufeinander folgenden Teilstücken von je 1/8 Punkt Länge; am Anfang werden  $loff$  Teilstücke von je 1/8 Punkt Länge übersprungen.

**$\&! \backslash (a, b, n, p, l, w)$**  Linie mit ausgefüllten Pfeilspitzen zwischen gemerkten Punkten

Wie  $\&! / (a, b, n)$ , aber mit ausgefüllten Pfeilspitzen der Länge 1/8 und der Breite  $w/8$  Punkt an beiden Enden (bei  $p=0$ ) bzw. am Ende, das auf die mit a (bei  $p=1$ ) bzw. auf die mit b (bei  $p=2$ ) bezeichnete Stelle zeigt.

**$\&! \backslash (a, b, n, p, l, w)$**  Linie mit offenen Pfeilspitzen zwischen gemerkten Punkten

Wie  $\&! / (a, b, n)$ , aber mit offenen Pfeilspitzen der Länge 1/8 und der Breite  $w/8$  Punkt an beiden Enden (bei  $p=3$ ) bzw. am Ende, das auf die mit a (bei  $p=4$ ) bzw. auf die mit b (bei  $p=5$ ) bezeichnete Stelle zeigt.

Für beide Pfeilspizentypen gilt: Maximale Breite der Pfeilspitzen = 127/8 Punkt, maximale Länge = 255/8 Punkt

**$\&! \backslash (a, b, n, ng, l, w)$**  Akkolade (geradlinig) zwischen gemerkten Punkten

Wie  $\&! / (a, b, n)$ , aber mit Kreisbögen von  $ng$  Grad ( $ng =$  Zahl von 6 bis 120) und mit Radius  $w/8$  Punkt an beiden Enden, nach links bei  $l=0$  bzw. nach rechts bei  $l=1$ .

- &! = (a, b, n)** Rechteck zwischen gemerkten Punkten  
 Rechteck mit den diagonal gegenüberliegenden (gemerkten) Eckpunkten a und b und mit der Strichstärke  $n/8$  Punkt zeichnen. Die Position des Schreibstrahls wird nicht verändert.  
 Für die Angaben zu a, b und n gilt das gleiche wie bei der Anweisung  $\&! / (a, b, n)$ .
- &! = (a, b, 0, g)** Graues Rechteck zwischen gemerkten Punkten  
 Rechteck mit den diagonal gegenüberliegenden (gemerkten) Eckpunkten a und b mit dem Grauwert g ausgefüllt ausgeben. Sonst wie  $\&! = (a, b, n)$
- &! = (a, b, -1, c, m, y, b)** Farbiges Rechteck zwischen gemerkten Punkten  
 Rechteck mit den diagonal gegenüberliegenden (gemerkten) Eckpunkten a und b ausgeben, ausgefüllt mit dem Farbwert, der durch die vier Grundfarben c, m, y, b (Cyan, Magenta, Yellow, Black; vgl. die Anweisung  $\&! C (c, m, y, k)$  auf Seite 1261) definiert ist. Sonst wie  $\&! = (a, b, n)$
- &! = (a, b, -2, nm, proz)** Farbiges Rechteck zwischen gemerkten Punkten  
 Rechteck mit den diagonal gegenüberliegenden (gemerkten) Eckpunkten a und b ausgeben, ausgefüllt mit der HKS-Farbe nm (n = Farbnummer, m = Material) und dem Deckungsgrad proz (vgl. die Anweisung  $\&! C (Hnm, proz)$  auf Seite 1262). Sonst wie  $\&! = (a, b, n)$
- &! - (a, b)** Unterstreichungsstrich  
 Waagerechte Linie auf der Geviertunterkante der aktuellen Zeile von horizontaler Position a bis b.  
 a und b können sein:  
 - Zahlenangabe (bis 4-stellig, maximal Satzspiegelbreite): Abstand vom linken Rand in Punkt (z. B.:  $\&! - (0, 36)$  für 36 Punkt lange Linie vom linken Rand an)  
 - Bruchteil der Satzbreite (z. B.:  $\&! - (1/2, 1/1)$  für Unterstreichen der rechten Zeilenhälfte, ab Satzspiegelmitte bis zum rechten Rand)  
 - Angabe von (zuvor definierten) Merkstellen (z. B.:  $\&! - (p1, p2)$  oder  $\&! - (p(11), p(12))$ , vgl. dazu 10.4.).  
 Die Angabe der Merkstellen durch  $B_n$  statt  $P_n$  (z. B.  $\&! - (b1, b2)$  oder  $\&! - (b(11), b(12))$ ) sollte nur benutzt werden, wenn die Zeile nicht ausgetrieben werden soll (z. B. bei Rausatz / Flattersatz) oder wenn zwischen den beiden Positionen keine Spatien vorkommen. Die Linie hat sonst nicht die nach dem Austreiben der Zeile erforderliche Länge.  
 Nach Ausführung der Anweisung steht der Schreibstrahl an der durch die Unterstreichung erreichten Position; Doppelbelichtung möglich.

- &!- (a, b, n)** Waagerechte Linie mit Strichstärke  $n$
- Die Angaben  $a$  und  $b$  entsprechen denen der Anweisung  $\&!- (a, b)$ . Zusätzlich wird hinter einem weiteren Komma die Strichstärke in Vielfachen von  $1/8$  Punkt angegeben. Maximale Strichstärke ist  $255/8$  Punkt. Die Oberkante der waagerechten Linie ist die Schriftgrundlinie.
- $n$  gibt die Strichstärke in  $1/8$  Punkt an. Soll statt einer durchgezogenen Linie eine gestrichelte Linie gezogen werden, so kann  $n:ls:lw:loff$  statt  $n$  angegeben werden. Die Linie besteht dann aus jeweils  $ls$  schwarzen und  $lw$  weißen aufeinander folgenden Teilstücken von je  $1/8$  Punkt Länge; am Anfang werden  $loff$  Teilstücke von je  $1/8$  Punkt Länge übersprungen.
- &! . (a, b)** Wie  $\&!- (a, b)$ , aber Punktreihe
- &! . (a, b, n)** Wie  $\&! . (a, b)$ , aber Sperrung der Punkte mit  $nn/50$  Geviert
- &!-- (a, b)** Fußnotenlinie
- Waagerechte Linie, um  $2/3$  Geviert (variabler Abstand; wird durch automatischen Spaltenhöhenausgleich verändert) unter Geviertunterkante der aktuellen Zeile und mit  $4/3$  Geviert (variabel) Abstand zur nachfolgenden Zeile, von horizontaler Position  $a$  bis  $b$ ; Angabe  $a$  und  $b$  wie bei  $\&!- (a, b)$ ; schließt Zeilenwechselanweisung ein.
- Vor  $\&!-- (a, b)$  und  $\&! . . (a, b)$  muss Leerzeichen oder Zeilenwechsel stehen
- &! . . (a, b)** Wie  $\&!-- (a, b)$ , aber Punktreihe
- &! . . (a, b, n)** Wie  $\&! . . (a, b)$ , aber Sperrung der Punkte mit  $nn/50$  Geviert

### 13. Griechische Schrift

Die Eingabe der griechischen Buchstaben geschieht (nach der Umschaltung auf die griechische Schrift) nach folgender Transkriptionstabelle (die Reihenfolge ist die des griechischen Alphabets):

a b g d e z h u i k l m n j o p r s t y f x c v  
 α β γ δ ε ζ η θ ι κ λ μ ν ξ ο π ρ σ τ υ φ χ ψ ω  
 A B G D E Z H U I K L M N J O P R S T Y F X C V  
 Α Β Γ Δ Ε Ζ Η Θ Ι Κ Λ Μ Ν Ξ Ο Π Ρ Σ Τ Υ Φ Χ Ψ Ω

Diese Transkriptionstabelle entspricht der Anordnung der Buchstaben auf der Tastatur griechischer Schreibmaschinen; vgl. auch »Griechische Schrift« im Kapitel »Zeichenvorrat« der Beschreibung »TUSTEP-Grundlagen«.

w	ς	(Schluss-Sigma)
^u	θ	theta (alternative Form)
^f	φ	phi (alternative Form)
W	Ϝ	Digamma
#.w	ϝ	Digamma
q	σ	(sigma in Form des lateinischen »c«)
Q	Σ	(Sigma in Form des lateinischen »C«)
^a	α	(alpha mit iota subscriptum)
^h	η	(eta mit iota subscriptum)
^v	ω	(omega mit iota subscriptum)
%;	ι	(iota subscriptum unter anderen Buchstaben)
#.s	ς	(stigma = 6)
#.k	ϙ	(koppa = 90)
#.p	Ϟ	(sampi = 900)

Satzzeichen, die vom Lateinischen abweichen:

;	;	(griechisches Fragezeichen)
?	;	(griechisches Fragezeichen)
!	·	(Semikolon, hochgestellter Punkt)

Sollen die (lateinischen) Zeichen ? und ! im griechischen Text gesetzt werden, so ist dafür ^? bzw. ^! zu schreiben.

Die griechischen Akzente:

%(	ἄ	Spiritus asper
%)	ᾶ	Spiritus lenis
%/	ᾷ	Akut
%\	Ᾱ	Gravis
;%?	ᾶ̃	Zirkumflex (Tilde)
%(/	ᾷ̃	Spiritus asper + Akut
%(\	Ᾱ̃	Spiritus asper + Gravis
%(?	ᾶ̣̃	Spiritus asper + Zirkumflex

%) /	ǎ	Spiritus lenis + Akut
%) \	ǒ	Spiritus lenis + Gravis
%) ?	ǣ	Spiritus lenis + Zirkumflex
%/ :	ĩ	Akut + Trema
%\ :	ĩ	Gravis + Trema

Zusätzlich sind die für lateinische Schriften geltenden Akzent-Codierungen %- -- % . % . . % : % : : % ' % ' ' möglich. Ein Kürzezeichen (Halbkreis, oben offen) über dem Buchstaben kann mit %, codiert werden.

Bei Verwendung der NewtonGreek Fonts (Schriftnummern 31781, 31782, 31783, 31785) können Akut + Trema bzw. Gravis + Trema mit %/ : bzw. %\ : codiert werden; Akut bzw. Gravis stehen dann nicht über dem Trema wie in ĩ, sondern zwischen den beiden Trema-Punkten wie in ĩ. Die gleiche Codierung ist in anderen Griechisch-Fonts, die diese Akzentkombination enthalten, möglich, wenn diese Zeichen mit Hilfe des Parameters BIL auf die Primäradressen 158 bzw. 159 gelegt werden.

## 14. Koptische Schrift

Die Eingabe der koptischen Buchstaben geschieht (nach der Umschaltung auf die koptische Schrift) nach folgender Transkriptionstabelle (Reihenfolge ist die des koptischen Alphabets):

a b g d e z E u i k l m n X o p r s t y F K P O

ⲁ ⲃ Ⲅ ⲅ Ⲇ ⲇ Ⲉ ⲉ Ⲋ ⲋ Ⲍ ⲍ Ⲏ ⲏ Ⲑ ⲑ Ⲓ ⲓ Ⲕ ⲕ Ⲗ ⲗ Ⲙ ⲙ Ⲛ

S f H B x h D G T W R

ϣ ϣ ϥ ϧ ϩ ϫ Ϭ ϭ Ϯ ϯ

Im koptischen Font sind außerdem folgende Zeichen definiert:

0 1 2 3 4 5 6 7 8 9 . , ; ! ? - / = ( ) [ ] | ^< ^>

0 1 2 3 4 5 6 7 8 9 . , ; · ; - / ≠ ( ) [ ] | < >

Für andere Satzzeichen und Sonderzeichen muss auf die lateinische Schrift umgeschaltet werden.

Folgende Akzentzeichen bzw. diakritische Zeichen sind im koptischen Font enthalten:

%.e	é
%:e	ë
%-e	ε
%..e	ε̇

## 15. Hebräische Schrift

Beim Hebräischen handelt es sich um eine linksläufige Schrift. Linksläufige Textteile werden in TUSTEP ebenfalls von links nach rechts (d. h. so, als würde der Text in lateinischen Buchstaben geschrieben) erfasst. Beim Satz von Texten, die solche Teile enthalten, muss in einem ersten (vorläufigen) Satzlauf zunächst der Zeilenumbruch vorgenommen werden; vor dem endgültigen Satzlauf müssen die linksläufigen Teile in den bereits umbrochenen Zeilen umgedreht werden.

Mit dem Parameter HBU kann jedoch verlangt werden, dass dies im Satzprogramm selbst (genauer: bei der PostScript-Ausgabe) geschehen soll.

Die Eingabe der hebräischen Zeichen geschieht (nach der Umschaltung auf die hebräische Schrift) nach folgender Transkriptionstabelle (Reihenfolge des hebräischen Alphabets; vgl. auch »Hebräische Schrift« im Kapitel »Zeichenvorrat« der Beschreibung »TUSTEP-Grundlagen«).

a b g d h u z x j i k l m n s y p c q r w t - : ' "

א ב ג ד ה ו ז ח ט י כ ל מ נ ס ע פ צ ק ר ש ת - : ' "

sin und schin sind in unpunktieren Texten nicht unterschieden. Die Schlussbuchstaben werden durch vorangestelltes ^ codiert:

^k = ך, ^m = ם, ^n = ן, ^p = ף, ^c = ץ.

Buchstaben mit Dageš sind als Großbuchstaben zu schreiben.

Ligaturen: # .u = ׁ, # .^u = ׂ, # .i = ׃

Vokalzeichen:

Die Vokalzeichen (einschl. Sin- und Schin-Punkt) stehen bei der Erfassung hinter dem Konsonanten, unter bzw. über den sie gesetzt werden sollen. Wenn das Satzprogramm die hebräischen Textteile nicht selbständig umdreht (wenn also nicht mit dem Parameter »HBU 0« angegeben ist, dass die hebräischen Teile automatisch umgedreht werden sollen), müssen diese Zeichen beim endgültigen Satz vor dem Konsonanten stehen, unter bzw. über den sie gesetzt werden sollen; dies ist beim externen Umdrehen der hebräischen Teile zu berücksichtigen.

#"0	◌ְ Schwa
#"1	◌ֿ Chirek
#"2	◌ֻ Zere
#"3	◌ִ Segol
#"4	◌ֶ Kamaz
#"5	◌ֶ Patach
#"6	◌ֹ Kubuz
#"7	◌ֹ Cholem
#"8	◌ׂ Schin
#"9	◌ׂ Sin
#"3#"0 oder #"0#"3	◌ְֿ Chataf Segol
#"4#"0 oder #"0#"4	◌ִֿ Chataf Kamaz
#"5#"0 oder #"0#"5	◌ֶֿ Chataf Patach

Akzentzeichen:

Die Codierung für Akzentzeichen muss links vor einer eventuell vorhandenen Codierung für Vokalzeichen stehen.

#"-	ֿ einfacher Rafe
#"=	ֿ doppelter Rafe
#"*	ֿ Kreis
#"+	ֿ Kreuz
#">	ֿ Haken
#".	ֿ Rebia'
#",	ֿ Mätäg / Silluq
#"<	ֿ Atnach
%.	ֿ Punkt über Buchstabe (Zahlzeichen)

Nicht umzudrehende Einschübe bei automatischem Umdrehen (HBU 0) :

Kommen im linksläufigen Teil in arabischen Ziffern geschriebene Zahlen vor, so werden diese Ziffernfolgen nicht umgedreht.

Stehen im linksläufigen Teil Einschübe in nicht linksläufiger Schrift, so dürfen diese ebenfalls nicht mit umgedreht werden. Dazu wird *innerhalb* des linksläufigen (hebräischen) Teils auf eine andere Schrift umgeschaltet, ohne die hebräische Schrift mit #H- auszuschalten; nach dem Beenden der Umschaltung auf die nicht linksläufige Schrift wird wieder linksläufig weitergesetzt.

Beispiel für die Codierung eines lateinischen Einschubs (in der Grundschrift) in einem hebräischen Teil:

```
#H+.....{\lateinisch{{{.....#H-
```

Andere Einschübe in hebräischen Teilen, die nicht umgedreht werden sollen, müssen durch folgende Anweisungen gekennzeichnet werden:

**#N+** Beginn eines nicht umzudrehenden Textteils

**#N-** Ende eines nicht umzudrehenden Textteils

Nicht-hebräische Einschübe in hebräischen Textteilen dürfen nicht mit #N+...#N- gekennzeichnet werden.



## 16. Arabische Schrift

Für das Arabische benutzt das Satzprogramm die Fonts Naskh Regular (Fontnamen: NaskhOneQubic, NaskhTwoQubic, NaskhThreeQubic) bzw. NaskhBold (Fontnamen: NaskhBoldOneQubic, NaskhBoldTwoQubic, NaskhBoldThreeQubic) der Firma Monotype Typography Ltd. Diese Fonts müssen auf dem benutzten Drucker bzw. Belichter installiert sein oder vor der Ausgabe in die PostScript-Datei inkludiert werden. Sie sind nicht Bestandteil des TUSTEP-Satzprogramms.

Beim Arabischen handelt es sich um eine linksläufige Schrift. Linksläufige Textteile werden in TUSTEP ebenfalls von links nach rechts (d. h. so, als würde der Text in lateinischen Buchstaben geschrieben) erfasst. Beim Satz von Texten, die solche Teile enthalten, muss in einem ersten (vorläufigen) Satzlauf zunächst der Zeilenumbruch vorgenommen werden; vor dem endgültigen Satzlauf müssen die linksläufigen Teile in den bereits umbrochenen Zeilen umgedreht werden.

Mit dem Parameter HBU kann jedoch verlangt werden, dass dies im Satzprogramm selbst (genauer: bei der PostScript-Ausgabe) geschehen soll.

Die Eingabe der arabischen Zeichen geschieht (nach der Umschaltung auf die arabische Schrift) nach folgender Transkriptionstabelle (Reihenfolge des arabischen Alphabets; vgl. auch »Arabische Schrift« im Kapitel »Zeichenvorrat« der Beschreibung »TUSTEP-Grundlagen«).

a b t o j h x d D r R s w c g p z y v f q k l m n e u i  
 ا ب ت ج ح د ذ ر ز س ش ص ط ظ ع ف ق ك ل م ن و ه ي

Die Unterscheidung von links-, beidseitig-, rechts- und unverbundenen Buchstaben muß mit dem Standard-Makro \*CASH vor dem Aufruf des Satzprogramms vorgenommen werden.

Ligaturen: Das Satzprogramm setzt bei Buchstabenkombinationen, für die im benutzten Font Ligaturen vorhanden sind, automatisch die entsprechenden Ligaturen ein. Dies sind weit mehr als die im Kapitel »Zeichenvorrat« für die arabische Schreibmaschinenschrift angeführten, mit #. zu codierenden Ligaturen.

Ziffern, Sonderzeichen:

0 1 2 3 4 5 6 7 8 9 . , : ; ? ! ' "  
 ٠ ١ ٢ ٣ ٤ ٥ ٦ ٧ ٨ ٩ . , : ; ؟ ! ' -

Vokalzeichen, Lesezeichen:

Die Vokal- bzw. Lesezeichen müssen vor dem Konsonanten stehen, über bzw. unter den sie gesetzt werden sollen.

%)	ء	Hamza
%]	ء	Hamza rechts über Ligatur
%/	َ	Fatha
%\	َ	Fatha rechts über Ligatur
%"	َ	Fatha-Tanwīn
%=	َ	Fatha-Tanwīn rechts über Ligatur
%,	ء	Damma
%'	ء	Damma rechts über Ligatur

◌:	◌̣	Ḍamma-Tanwīn
◌ [	◌̣	Ḍamma-Tanwīn rechts über Ligatur
◌ *	◌̣	Sukūn
◌ (	◌̣	Sukūn rechts über Ligatur
◌ >	◌̣	Tašdīd
◌ <	◌̣	Tašdīd rechts über Ligatur
◌ !	◌̣	Alif
◌	◌̣	Alif rechts über Ligatur
◌ ?	◌̣	Madda
◌ +	◌̣	Madda rechts über Ligatur
◌ .	◌̣	Waṣla
◌ -	◌̣	Waṣla rechts über Ligatur

## Doppelvokale:

◌ /	◌̣	Faṭḥa + Hamza
◌ /	◌̣	Faṭḥa + Hamza rechts
◌ \	◌̣	Faṭḥa + Hamza rechts
◌ / >	◌̣	Faṭḥa + Tašdīd
◌ / <	◌̣	Faṭḥa + Tašdīd rechts
◌ \ <	◌̣	Faṭḥa + Tašdīd rechts
◌ "	◌̣	Faṭḥa-Tanwīn + Hamza
◌ " ]	◌̣	Faṭḥa-Tanwīn + Hamza rechts
◌ = ]	◌̣	Faṭḥa-Tanwīn + Hamza rechts
◌ " >	◌̣	Faṭḥa-Tanwīn + Tašdīd
◌ " <	◌̣	Faṭḥa-Tanwīn + Tašdīd rechts
◌ = <	◌̣	Faṭḥa-Tanwīn + Tašdīd rechts
◌ * )	◌̣	Sukūn + Hamza
◌ * ]	◌̣	Sukūn + Hamza rechts
◌ (	◌̣	Sukūn + Hamza rechts
◌ , )	◌̣	Ḍamma + Hamza
◌ , ]	◌̣	Ḍamma + Hamza rechts
◌ ' ]	◌̣	Ḍamma + Hamza rechts
◌ , >	◌̣	Ḍamma + Tašdīd
◌ ' <	◌̣	Ḍamma + Tašdīd rechts
◌ ' <	◌̣	Ḍamma + Tašdīd rechts
◌ : )	◌̣	Ḍamma-Tanwīn + Hamza
◌ : ]	◌̣	Ḍamma-Tanwīn + Hamza rechts
◌ [ ]	◌̣	Ḍamma-Tanwīn + Hamza rechts
◌ : >	◌̣	Ḍamma-Tanwīn + Tašdīd
◌ : <	◌̣	Ḍamma-Tanwīn + Tašdīd rechts
◌ [ <	◌̣	Ḍamma-Tanwīn + Tašdīd rechts
◌ > /	◌̣	Tašdīd + Faṭḥa
◌ > \	◌̣	Tašdīd + Faṭḥa rechts
◌ < \	◌̣	Tašdīd + Faṭḥa rechts
◌ > "	◌̣	Tašdīd + Faṭḥa-Tanwīn
◌ > =	◌̣	Tašdīd + Faṭḥa-Tanwīn rechts
◌ < =	◌̣	Tašdīd + Faṭḥa-Tanwīn rechts

Unter den Buchstaben:

%) )	ء	Hamza
%) ]	ء	Hamza rechts unter Ligatur
%) /	ء	Kasra
%) \	ء	Kasra rechts unter Ligatur
%;	ء	Hamza + Kasra
%) }	ء	Hamza + Kasra rechts unter Ligatur
%) ""	ء	Kasra-Tanwīn
%) ==	ء	Kasra-Tanwīn rechts unter Ligatur
%) "	ء	Hamza + Fathā-Tanwīn
%) =	ء	Hamza + Fathā-Tanwīn rechts
%) ] =	ء	Hamza + Fathā-Tanwīn rechts
%; ;	ي	i-Punkte für Schluss-Ya'
%; ,	ي	i-Punkte + Kasra für Schluss-Ya'
%; ' '	ي	i-Punkte + Kasra-Tanwīn für Schluss-Ya'

Die Codierung für die rechts über einer Ligatur stehenden Vokalzeichen und für die Doppelvokale wird mit dem Standard-Makro \*CASH automatisch aus einer Eingabecodierung erzeugt, in der die Vokalzeichen jedem Konsonanten zugeordnet sind; so wird aus der Eingabecodierung

```
#A+a%*l%/%) asfar#A-
```

mit \*CASH die Codierung

```
#A+Ä%/%) (#. ^XSfär#A- für الأَسْفَار
```

oder aus

```
#A+%) ) a%/l%>% ; ; i#A- die Codierung
```

```
#A+%) ) Ä%\%>% ; ; #. ^L#A- für إِي.
```

## Kashida

Zum Austreiben der Zeilen werden im Arabischen nicht die Wortzwischenräume vergrößert, sondern die Verbindungsstriche zwischen bestimmten verbundenen Buchstaben durch sogenannte Kashidas verlängert. Da die Auswahl dieser Stellen kalligraphischen Regeln folgt, die nicht leicht mechanisch zu fassen sind, ist es dem Anwender überlassen, die betreffenden Stellen durch "--" zu kennzeichnen. Wird eine Zeile nicht ausgetrieben, so ist in der Satzausgabe an diesen Stellen nichts zu sehen; die betreffenden Buchstaben stehen ganz normal nebeneinander. Steht "--" jedoch zwischen Buchstaben, die sonst als Ligatur gesetzt werden, so werden die Buchstaben nebeneinander gesetzt, ohne zu einer Ligatur zusammengefasst zu werden.

Wird eine Zeile ausgetrieben, und ist in dieser Zeile mindestens eine Kashida codiert, so unterbleibt das Vergrößern der Wortzwischenräume; der zum Austreiben notwendige Platz wird gleichmäßig auf die in der Zeile codierten Kashidas verteilt.

Enthält eine Zeile keine Codierung für eine Kashida, so wird der Wortzwischenraum zum Austreiben der Zeile benutzt.

Nicht umzudrehende Einschübe bei automatischem Umdrehen (HBU 0) :

Kommen im linksläufigen Teil in arabischen Ziffern geschriebene Zahlen vor, so werden diese Ziffernfolgen nicht umgedreht.

Stehen im linksläufigen Teil Einschübe in nicht linksläufiger Schrift, so dürfen diese ebenfalls nicht mit umgedreht werden. Dazu wird *innerhalb* des linksläufigen (arabischen) Teils auf eine andere Schrift umgeschaltet, ohne die arabische Schrift mit #A- auszuschalten; nach dem Beenden der Umschaltung auf die nicht linksläufige Schrift wird wieder linksläufig weitergesetzt.

Beispiel für die Codierung eines lateinischen Einschubs (in der Grundschrift) in einem arabischen Teil:

```
#A+.....{\lateinisch{{.....#A-
```

Andere Einschübe in arabischen Teilen, die nicht umgedreht werden sollen, müssen durch folgende Anweisungen gekennzeichnet werden:

**#N+** Beginn eines nicht umzudrehenden Textteils

**#N-** Ende eines nicht umzudrehenden Textteils

Nicht-arabische Einschübe in arabischen Textteilen dürfen nicht mit #N+...#N- gekennzeichnet werden.

## 17. Russische Schrift

Für das Russische unterstützt das Satzprogramm zwei Font-Familien: die Newton Cyrillic (Schriftnummern 31751, 31752, 31753; diese Fonts sind Bestandteil der TUSTEP-Installation) der Firma Paratype, und die Times Ten Cyrillic (Schriftnummern 31851, 31852, 31853, 31855; diese Fonts müssen auf dem benutzten Drucker bzw. Belichter installiert sein oder vor der Ausgabe in die PostScript-Datei inkludiert werden; sie sind nicht Bestandteil von TUSTEP) der Firma Linotype Library GmbH.

Die Eingabe der russischen Buchstaben geschieht (nach der Umschaltung auf die russische Schrift) nach folgender Transkriptionstabelle (Reihenfolge des russischen Alphabets; vgl. auch »Russische (kyrillische) Schrift« im Kapitel »Zeichenvorrat« der Beschreibung »TUSTEP-Grundlagen«):

```
a b v g d e ^e h z i j k l m n o p r s t u f x c q w
а б в г д е ё ж з и й к л м н о п р с т у ф х ц ч ш
^w ^p y ^b ä ü ö
щ ъ ы ь э ю я
```

Darüber hinaus stehen folgende in slavischen Sprachen vorkommende Zeichen zur Verfügung:

```
^c ^d ^f ^g ^h ^i ^j ^l ^n ^s ^v ^y ^z
ц ђ ɵ є ħ i j љ њ s v ѣ ж
```

## 18. Phonetische Zeichen

Für das Setzen phonetischer Zeichen steht ein phonetischer Zeichensatz mit Namen NewtonPhoneticTu zur Verfügung (Schriftnummer 31920, siehe oben bei der Beschreibung der Parameter unter »Schriften«). Darüber hinaus unterstützt TUSTEP die TimesPhonetic (Schriftnummern 31921 und 31922) der Firma Linotype Library GmbH. Sollen die Linotype-Fonts benutzt werden, so müssen sie auf dem benutzten Drucker bzw. Belichter installiert sein; sie sind nicht Bestandteil von TUSTEP.

Die Eingabe der phonetischen Zeichen geschieht (nach der Umschaltung auf den phonetischen Zeichensatz) nach der Transkriptionstabelle, die im Kapitel »Zeichenvorrat« der Beschreibung »TUSTEP-Grundlagen« beschrieben ist. Die Codierung der Diacritica mit %% wird vom Satzprogramm derzeit jedoch nicht unterstützt.

## IV. Generelle Anweisungen

### 19. Makros

Über Makros können mit den Parametern `MAC`, `MAA`, `MAH` angegebene Texte, Sonderzeichen oder Anweisungsfolgen aufgerufen werden.

**<name>** Aufruf des Makros »name«

Der Makroname »name« darf bis zu 640 Zeichen lang sein und aus Buchstaben, Ziffern und Sonderzeichen außer spitzen Klammern bestehen. Groß- und Kleinbuchstaben werden nur unterschieden, wenn dies mit dem Parameter `MAZ` verlangt wird.

Wenn Makros dieser Form benutzt werden, so müssen die im Text ggf. vorkommenden spitzen Klammern mit `<...^>` statt mit `<...>` codiert werden; der Zirkumflex unter dem Buchstaben muss mit `%^^` statt `%<<` codiert werden.

Wenn Makros dieser Form benutzt werden, wird die Zeichenfolge `<!-- ... -->` als Kommentar interpretiert.

Makros der Form `<?:t: xxxx?>` und der Form `<?tus xxxx?>` sind XML Processing Instructions (PIs). PIs mit den Namen `:t:` und `tus` werden beim Satz durch `xxxx` aufgelöst, es sei denn, dass sie über Parameter anders aufgelöst sind.

**&.ab** Aufruf des Makros »ab«

Die mit `&.ab` codierten Makros haben folgende Form: Nach `»&.«` zwei Zeichen (Buchstaben, Ziffern oder Sonderzeichen außer `$`). Groß- und Kleinbuchstaben werden dabei nur unterschieden, wenn dies mit dem Parameter `MAZ` verlangt wird.

Es werden zwei Gruppen von Makros unterschieden:

Gruppe A: Alle Makros, die nicht zu Gruppe B gehören.

Gruppe B: Mit `»&.ab«` codierte leere Makros (das sind Makros, zu denen hinter dem Makronamen keine Auflösung angegeben ist) sowie mit `»&.ab«` codierte Makros, deren Auflösung mit `»&! («` beginnt und die nur eine Code-Folge von hardware-nahen Anweisungen (Interncode, vgl. Kapitel 22) enthalten; die Auflösung dieser Makros muss mit der entsprechenden schließenden Klammer enden.

Stehen Makros der Gruppe B zwischen Leerzeichen, so werden, auch wenn das Makro leer ist bzw. wenn dessen Auflösung keine Schreibstrahl-Bewegung verursacht, die beiden das Makro umgebenden Leerzeichen nicht zu einem Spatium zusammengefasst.

Da Mehrfach-Leerzeichen sonst zu einem Leerzeichen zusammengefasst werden, gibt die Verwendung von leeren, mit `»&.ab«` codierten Makros die Möglichkeit, doppelt breite Spatien zu erhalten, die beim Zeilenwechsel – im Unterschied zu `\` bzw. `^^` (vgl. Kapitel 8 und 9) – unterdrückt werden.

Ein leeres Makro, das ohne Leerzeichen am Anfang, innerhalb oder am Ende eines Wortes steht, ist eine einfache Möglichkeit, Wörter ohne Auswirkung auf die Belichtung zu markieren. Vgl. auch die Anweisung `&X...&X{` (Kapitel 20).

Zur Behandlung der Makros in der ZIEL- und in der PROTOKOLL-Datei siehe die Hinweise bei der Beschreibung der Parameter MAC, MAA, MAH und MAZ.

## 20. Kommentar

Der zwischen den Anweisungen `&X` und `&X{` sowie zwischen `<!--` und `-->` stehende Text wird vom Satzprogramm übergangen (wird also nicht auf die AUSGABE-Datei geschrieben), aber im Satzprotokoll und in der ZIEL-Datei mitgeführt. Auf diese Weise ist es möglich, Information für nachfolgende Programme in den zu setzenden Text einzufügen.

Damit stehen z. B. zum Zweck der Registererstellung die Zeichenfolgen `&X` bzw. `&X{` für das Kommando `#RVORBEREITE` als Markierungen für den Anfang bzw. das Ende eines Registerintrags (Parameter EA bzw. EE) zur Verfügung.

**&X**                   Kommentar Anfang

**&X{**                   Kommentar Ende

Das zwischen `&X` und `&X{` Stehende darf die Länge von 1000 Zeichen (Voreinstellung; kann über Parameter ABB geändert werden) nicht überschreiten.

Kommen im gleichen Text (kritische) Apparate vor, so darf das erste Zeichen nach `&X` kein `x` bzw. nach `&x` kein `x` sein, da sonst keine Unterscheidung zwischen Kommentaren und Apparateinträgen möglich ist (vgl. Kapitel 5). Bei der Schreibweise `&Xxenophil&X{` oder `&xXenophanes&x{` ist eine Verwechslung mit einem Apparateintrag ausgeschlossen.

Ein Kommentar im Wortinnern, der Leerzeichen enthält, bedeutet gleichzeitig Kann-Trennstelle.

**<!-- ... -->**       Kommentar

Werden Makros der Form `<name>` benutzt, so wird das zwischen `<!--` und `-->` Stehende als Kommentar behandelt.



## 21. Verknüpfungen (Verweise) für PDF-Dateien

Soll die vom Satzprogramm erzeugte Ausgabe auch elektronisch in Form von PDF-Dateien publiziert werden, so können mit den folgenden Steueranweisungen Verknüpfungen mit Seiten und Zeilen oder mit benannten Zielen im gleichen oder in einem anderen Dokument vorbereitet und in die Satzausgabe eingebunden werden. Sie werden beim Aufruf von \*PSAUS so weiterverarbeitet, dass sie bei der Druckausgabe nicht stören, aber beim Umwandeln der PostScript-Datei in eine PDF-Datei als pdfmarks eingebunden werden.

Für das Aktivieren einer Verknüpfung ist eine Schaltfläche (ein Rechteck, das den Anfang und das Ende eines als Verweis dienenden Wortes oder – in einem Register – einer Stellenangabe umschließt) vorgesehen, die so zu bemessen ist, dass sie beim Betrachten der PDF-Datei mit dem Mauszeiger bequem und eindeutig angefahren werden kann.

Soll die Schaltfläche in der PostScript- bzw. PDF-Datei hervorgehoben werden, so kann dies mit entsprechenden Steueranweisungen für den Satz der betreffenden Textteile (Fett- oder Kursivdruck, farbiger Druck, Hinterlegen durch Grau- oder Farbraster) geschehen.

Hinweis: Für die Vorbereitung von Inhaltsverzeichnissen als Lesezeichen (Bookmarks) für PDF-Dateien steht das Standard-Makro \*BOOKMARKS zur Verfügung. Bookmarks und andere pdfmarks können wie Grafiken in die Ausgabedatei eingebunden werden (vgl. dazu die Beschreibungen von \*BOOKMARKS, \*GRAFIK und \*PSAUS).

### 21.1. Verknüpfung mit Seiten

**&! (#Ly/n/z)** Anfang einer Schaltfläche für eine Verknüpfung (pdfmark /ANN, Subtype /Link).

Untere Ecke eines unsichtbaren Rechtecks, das als Schaltfläche für eine Verknüpfung mit der Seite  $n$  dienen soll.

Die Unterkante der Schaltfläche liegt um  $y$  Punkt unterhalb der aktuellen Schriftgrundlinie. Ist  $y$  gleich 0 oder fehlt  $y$ , so beginnt sie vertikal auf der aktuellen Schriftgrundlinie.

Die Verknüpfung erfolgt mit der Seite mit der laufenden Nummer  $n$  in der selben Datei. Diese Seite wird, wenn die Verknüpfung aktiviert wird, um  $z$  Punkt gegenüber dem Seitenanfang nach oben verschoben.

Hinweis: Zur Berechnung des Wertes, der für  $z$  eingesetzt werden muss, damit beim Betrachten der PDF-Datei das Sprungziel in der ersten Zeile des Fensters angezeigt wird, können die Angaben in der zweiten zu PROTOKOLL angegebenen Datei zugrunde gelegt werden.

**&! (#L/n)** Wie &! (#Ly/n/z). Werden  $y$  und/oder  $z$  nicht angegeben, so wird jeweils der Wert 0 angenommen.

**&! (#Ly/d:n/z)** Wie &! (#Ly/n/z), jedoch Verknüpfung mit der Seite mit der laufenden Nummer  $n$  in der  $d$ -ten externen Datei.

$d$  ist eine einstellige Zahl und gibt an, als wievielter der Name der externen PDF-Datei beim Aufruf von \*PSAUS zur Spezifikation EXTERN angegeben wird.

**&! (#L/d:n)** Wie &! (#Ly/d:n/z). Werden  $y$  und/oder  $z$  nicht angegeben, so wird jeweils der Wert 0 angenommen.

**&! (#Ly)** Ende der Schaltfläche

Endpunkt (dem Anfangspunkt diagonal gegenüberliegende obere Ecke eines unsichtbaren Rechtecks) der mit einer der vorstehenden Anweisungen begonnenen Schaltfläche. Die Oberkante des Rechtecks liegt um  $y$  Punkt oberhalb der aktuellen Schriftgrundlinie.

**&! (#L)** Wie &! (#Ly); die Oberkante des Rechtecks liegt 11 Punkt oberhalb der aktuellen Schriftgrundlinie.

Beispiel für eine auf der Schriftgrundlinie beginnende Schaltfläche mit 11 Punkt Höhe für eine Verknüpfung mit der Seite 201 in der selben Datei:

`&! (#L/201)Stichwort&! (#L) .`

Die Schaltfläche kann auch von rechts nach links aufgebaut werden; Beispiel (gleichwertig mit dem soeben angeführten Beispiel):

`&! (#L)Stichwort&! (#L/201) .`

## 21.2. Verknüpfung mit benannten Zielen (named destinations)

### 21.2.1. Definition eines benannten Zieles

**&! (#Dn/z)** Definition eines benannten Zieles in der selben Datei.

Erzeugt eine pdfmark-Anweisung /DEST als benanntes Ziel (named destination) mit der Nummer  $n$  für Verknüpfungen, auf das mit der Anweisung &! (#Ly/n) verwiesen werden kann. Die Nummer  $n$  muß größer als 0 sein.

Beim Aktivieren der Verknüpfung mit diesem Ziel wird die angezeigte Seite um  $z$  Punkt gegenüber dem Seitenanfang nach oben verschoben.

Fehlt  $z$ , so wird die Seite nicht verschoben.

Benannte Ziele (named destinations), die mit einer der beiden folgenden Anweisungen definiert werden, sind in jedem Fall unabhängig vom Seitenumbruch. Die Definition dieser Ziele und die Verknüpfungen damit können deshalb schon vor dem Aufruf des Satzprogramms mit allen notwendigen Angaben festgelegt werden. (Dies kann in Sonderfällen auch für die Anweisung &! (#Dn/z) gelten.)

**&! (#Dn/\*+y)** Wie &! (#Dn/z), jedoch wird die betreffende Seite beim Anzeigen so verschoben, dass die Schriftgrundlinie, auf der ein hier gesetztes Zeichen stehen würde, um  $y$  Punkt unterhalb der Oberkante des Fensters liegt.

**&! (#Dn)** Wie &! (#Dn/z) bzw. &! (#Dn/\*+y), jedoch wird die Seite beim Anzeigen nicht verschoben, sondern von oben an gezeigt.

## 21.2.2. Verknüpfung mit einem benannten Ziel

**&! (#LDy/n)** Anfang einer Schaltfläche für eine Verknüpfung mit dem benannten Ziel Nummer *n* in der selben Datei.

Untere Ecke eines unsichtbaren Rechtecks, das als Schaltfläche für eine Verknüpfung mit dem benannten Ziel Nummer *n* dienen soll.

Die Unterkante der Schaltfläche liegt um *y* Punkt unterhalb der aktuellen Schriftgrundlinie.

**&! (#LD/n)** Wie &! (#LDy/n). Wird *y* nicht angegeben, so wird der Wert 0 angenommen.

**&! (#LDy/d:n)** Anfang einer Schaltfläche für eine Verknüpfung mit dem benannten Ziel Nummer *n* in der *d*-ten externen Datei. Sonst wie &! (#LDy/n).

*d* ist eine einstellige Zahl und gibt an, als wievielter der Name dieser externen PDF-Datei beim Aufruf von \*PSAUS zur Spezifikation EXTERN angegeben wird.

**&! (#LD/d:n)** Wie &! (#LDy/d:n). Wird *y* nicht angegeben, so wird der Wert 0 angenommen.

**&! (#Ly)** und

**&! (#L)** Ende der Schaltfläche

Beschreibung siehe oben unter 21.1

Die Schaltfläche kann auch von rechts nach links aufgebaut werden; die Angaben &! (#LD/n) *Stichwort* &! (#L) und &! (#L) *Stichwort* &! (#LD/n) sind also gleichwertig.

## 21.3. Verknüpfung zu einem Dokument im WWW

**&! (#PUy/uri)** Anfang einer Schaltfläche für eine Verknüpfung (pdfmark /ANN, Subtype /Link, Action /Launch /URI) mit dem Dokument, das im WWW über den mit *uri* angegebenen URL (Universal Resource Locator; in PDF immer als URI = Universal Resource Identifier bezeichnet) erreichbar ist.

Die Unterkante des als Schaltfläche dienenden unsichtbaren Rechtecks liegt um *y* Punkt unterhalb der aktuellen Schriftgrundlinie.

**&! (#PU/uri)** Wie &! (#PUy/uri). Wird *y* nicht angegeben, so wird der Wert 0 angenommen.

**&! (#Ly)** und

**&! (#L)** Ende der Schaltfläche

Beschreibung siehe oben unter 21.1

Die Schaltfläche kann auch von rechts nach links aufgebaut werden; die Angaben `&!(#PUy/uri)Stichwort&!(#L)` und `&!(#L)Stichwort&!(#PUy/uri)` sind also gleichwertig.

## 21.4. Aufruf eines anderen Dokuments oder eines Programms

**&!(#PLY/pfad)** Anfang einer Schaltfläche für eine Verknüpfung (`pdfmark /ANN, Subtype /Link Action /Launch /File`) mit dem Dokument bzw. Programm, das über die Angabe zu `pfad` (z. B. `c:\briefe\muster.doc` für ein Word-Dokument oder `c:\windows\system32\mspaint.exe` für das Windows-Programm Paint).

Die Unterkante des als Schaltfläche dienenden unsichtbaren Rechtecks liegt um `y` Punkt unterhalb der aktuellen Schriftgrundlinie.

**&!(#PL/pfad)** Wie `&!(#PLY/pfad)`. Wird `y` nicht angegeben, so wird der Wert 0 angenommen.

**&!(#Ly)** und

**&!(#L)** Ende der Schaltfläche

Beschreibung siehe oben unter 21.1

Die Schaltfläche kann auch von rechts nach links aufgebaut werden; die Angaben `&!(#PLY/pfad)Stichwort&!(#L)` und `&!(#L)Stichwort&!(#PLY/pfad)` sind also gleichwertig.

## 21.5. Einfügen von Notizen / Kommentaren

**&!(#Nxy/farbe::titel::text)** Einfügen einer Notiz (eines Kommentars) (`pdfmark /ANN, Subtype /Text`)

Beim Öffnen des Dokuments ist an dieser Stelle ein Symbol sichtbar; ein PopUp-Fenster mit dem Inhalt der Notiz kann durch Doppelclick auf das Symbol geöffnet werden

**&!(#Nxy/farbe;;;titel;;;text)** Einfügen einer Notiz (eines Kommentars) (`pdfmark /ANN, Subtype /Text`)

Beim Öffnen des Dokuments ist außer dem Symbol für die Notiz auch der Text der Notiz in einem PopUp-Fenster geöffnet.

Notizen werden mit Symbolen bzw. Markierungen angezeigt, die an dieser Stelle im Text erscheinen. Die Symbole werden über einen für `x` einzusetzenden Buchstaben ausgewählt. Folgende Symbole stehen zur Verfügung:

- B** Büroklammer (`/Name /Paperclip`)
- C** Circle, Kreis, Ellipse (`/Name /Circle`)
- E** Einfügung, Dreieck (`/Name /Insert`)
- G** Grafik, Balkengrafik (`/Name /Graph`)

<b>H</b>	Hilfe, Fragezeichen im Kreis (/Name /Help)
<b>K</b>	Kommentar, Sprechblase (/Name /Comment)
<b>L</b>	Linie (unsichtbar) (/Name /Line)
<b>N</b>	Nadel, Stecknadel (/Name /PushPin)
<b>P</b>	Paragraph (/Name /Paragraph)
<b>Q</b>	Quadrat, Rechteck (/Name /Square)
<b>S</b>	Schlüssel (/Name /Key)
<b>T</b>	Tag (/Name /Tag)
<b>Z</b>	Zettel, Notizzettel (/Name /Note)

Die Symbole (außer C und Q, die nicht als Fläche, sondern als Linien ausgegeben werden) überdecken ggf. den Text, der sich unter ihnen befindet.

Die Angabe zu  $y$  wird nur bei  $x = C$  und  $x = Q$  ausgewertet; sie gibt an, wie weit sich das Rechteck (»Quadrat«) bzw. die Ellipse (»Kreis«) unterhalb der aktuellen Schriftgrundlinie erstrecken soll.

Mit Angaben zu »farbe« kann die Farbe des Symbols und des Rahmens des PopUp-Fensters festgelegt werden. Die Angaben bestehen aus drei Zahlenwerten zwischen 0 und 1 (mit Dezimalpunkt statt Komma) im RGB-Farbsystem (rot, grün, blau).

Für »titel« und »text« wird Text eingesetzt. Der zu »titel« angegebene Text erscheint als Titelzeile des PopUp-Fensters, der zu »text« angegebene Text stellt den Inhalt des PopUp-Fensters dar.

Im Text vorkommende Wortzwischenräume sind mit »\_« statt Leerzeichen zu codieren, runde Klammern mit \ ( bzw. \), Umlaute, Akzentbuchstaben und andere Nicht-Ascii-Zeichen mit \nnn, wobei nnn eine Oktalzahl ist und den Code der Zeichen gemäß dem PDF DocEncoding angibt; für die Spalten 20–70 und A0 bis F0 entspricht dies (außer dem Zeichen A0, auf dem das Euro-Zeichen liegt) dem WindowsWestern-Encoding (CP1252, vgl. S. 664.)

**&! (#Ly)**

und

**&! (#L)**

Position des Symbols für die Notiz

Die Angabe zu  $y$  gibt an, wie weit oberhalb der Schriftgrundlinie das Symbol erscheinen bzw. (bei  $x = C$  und  $x = Q$ ), wie weit sich die Ellipse (»Kreis«) bzw. das Rechteck (»Quadrat«) über die Schriftgrundlinie hinaus erstrecken soll (Angabe in Punkt). Fehlt  $y$ , so wird ein Wert von 11 Punkt angenommen. Die Position dieser Anweisung wird ebenfalls nur bei  $x = C$  und  $x = Q$  ausgewertet und bestimmt deren horizontale Erstreckung.

## 22. Hardware-nahe Anweisungen (Interncode)

Die Anweisungen dieser Gruppe sollten im allgemeinen nicht in den Eingabedaten enthalten sein. Sie werden insbesondere für die Auflösung von Makros benötigt, wenn damit nicht über andere Steueranweisungen ansprechbare Leistungen der Satzausgabe verlangt werden sollen (z. B. Übereinander-Belichten von Zeichen).

Der dabei verwendete Code ist der (auf 8-bit-Werte ausgedehnte und um die am Schluss dieses Kapitels aufgeführten Codes erweiterte) Interncode des Digiset 50T1 (siehe Tabelle S. 1308); er wird, wenn nichts anderes angegeben ist, vom Programm an der entsprechenden Stelle ohne weitere Prüfung in die AUSGABE-Datei übernommen. Es kann also aus der Tatsache, dass das Satzprogramm bei der Verwendung dieser Anweisungen keinen Fehler meldet, nicht geschlossen werden, dass eine fehlerfreie Weiterverarbeitung dieser AUSGABE-Datei mit \*PSAUS möglich ist.

Bei einigen Anweisungen ist es jedoch sinnvoll, dem Satzprogramm eine Buchführung über die entsprechenden Schritte zu ermöglichen (so bei allen Verschiebe-Anweisungen) bzw. die Dicktenzählung für den Zeilenausgleich mitzuführen oder nach anderen Operationen den Grundzustand wieder herzustellen. Deshalb ist für einige Funktionen eine abweichende Form der Codierung vorgesehen.

Die hier einzeln zwischen &! ( und ) aufgeführten Codes sollten, wenn nichts anderes angegeben ist, zu einer einzigen Code-Folge zwischen &! ( und ) zusammengefasst werden. Dabei ist die Code-Folge in der Klammer mit der Angabe des Dicktenwertes, den der Klammersausdruck hat, abzuschließen (vgl. unten die Anweisungen &! (nn+d), &! (nn-d), &! (nn+d\*g)), falls dieser Dicktenwert nicht 0 ist oder falls er nicht lt. Beschreibung automatisch mit berücksichtigt wird.

- &! (+nn)** Verschieben des Schreibstrahls um nn Bildlinien nach rechts, mit Berücksichtigung bei der Zeilenlängen-Berechnung  
Für nn kann ein beliebiger Wert angegeben werden. In der Klammer darf sonst nichts stehen.
- &! (+nn=)** wie &! (+nn), jedoch nur auf linken Seiten bzw. in Spalten mit gerader Spaltennummer.
- &! (+nn/)** wie &! (+nn), jedoch nur auf rechten Seiten bzw. in Spalten mit ungerader Spaltennummer.
- &! (-nn)** Verschieben des Schreibstrahls um nn Bildlinien nach links, mit Berücksichtigung bei der Zeilenlängen-Berechnung  
Für nn kann ein beliebiger Wert angegeben werden. In der Klammer darf sonst nichts stehen.
- &! (-nn=)** wie &! (-nn), jedoch nur auf linken Seiten bzw. in Spalten mit gerader Spaltennummer.
- &! (-nn/)** wie &! (-nn), jedoch nur auf rechten Seiten bzw. in Spalten mit ungerader Spaltennummer.
- &! (++)** Verschieben des Schreibstrahls um nn Bildlinien nach rechts, ohne Berücksichtigung bei der Zeilenlängen-Berechnung

Für `nn` kann ein beliebiger Wert angegeben werden. Diese Angabe wird nach rechts abgeschlossen durch das nächste von einer Ziffer verschiedene Zeichen der zwischen `&! ( und )` vorgesehenen Code-Folge oder durch ein `\`.

**`&! (++nn=)`** wie `&! (++nn)`, jedoch nur auf linken Seiten bzw. in Spalten mit gerader Spaltennummer.

**`&! (++nn/)`** wie `&! (++nn)`, jedoch nur auf rechten Seiten bzw. in Spalten mit ungerader Spaltennummer.

**`&! (--nn)`** Verschieben des Schreibstrahls um `nn` Bildlinien nach links, ohne Berücksichtigung bei der Zeilenlängen-Berechnung

Für `nn` kann ein beliebiger Wert angegeben werden. Dieser Wert wird nach hinten abgeschlossen durch das nächste von einer Ziffer verschiedene Zeichen der zwischen `&! ( und )` vorgesehenen Code-Folge oder durch ein `\`.

**`&! (--nn=)`** wie `&! (--nn)`, jedoch nur auf linken Seiten bzw. in Spalten mit gerader Spaltennummer.

**`&! (--nn/)`** wie `&! (--nn)`, jedoch nur auf rechten Seiten bzw. in Spalten mit ungerader Spaltennummer.

**`&! (nn)`** Hardware-nahe Anweisung (Interncode) bzw. Primäradressenaufruf `nn`

`nn` ist eine zweistellige Dezimalzahl und muss einem legalen Interncode entsprechen.

**`&! (Dnnn)`** Wie `&! (nn)`, jedoch ist `nnn` eine dreistellige Dezimalzahl von 000 bis 255 (je einschließlich), bei `&! (00Dnnn)` und `&! (2836Dnnn)` von 000 bis 999

**`&! (Hnn)`** Wie `&! (nn)`, jedoch ist `nn` eine zweistellige Hexadezimalzahl von 00 bis FF (je einschließlich)

**`&! (nn+d)`** Dickenwert `d` (in Bildlinien), der der gesamten Anweisungsfolge `nn` außer den in der Klammer enthaltenen, mit `#nnnn` codierten Sonderzeichen, entspricht; `d` ist eine beliebige Dezimalzahl

Aus Gründen der Kompatibilität mit bereits vorhandenen älteren Daten können Werte, die größer sind als 99, durch Wiederholung angegeben werden, z. B. `+99+06` für »Dicke von 105 Bildlinien«.

**`&! (nn-d)`, `&! (nn+-d)`** Negativer Dickenwert `d` (in Bildlinien), sonst wie `&! (nn+d)`

Aus Gründen der Kompatibilität mit bereits vorhandenen älteren Daten können Werte, die kleiner sind als -99, durch Wiederholung angegeben werden, z. B. `+ -99+ -6` für »negative Dicke von 105 Bildlinien«.

**`&! (nn+d*g)`** Angabe der Schriftgröße `g`, auf die sich die unmittelbar davor in `+d` (bzw. bei `&! (nn-d*g)` in `-d`) genannte Bildlinienzahl bezieht

\*g muss angegeben werden, wenn in der Anweisungsklammer die Schriftgröße oder die Schrift-Dicke geändert wurde.

Für die Berechnung der Zeilenlänge werden die unmittelbar davor stehenden Angaben der Bildlinienzahl mit dem Faktor g multipliziert und durch die vor der Klammer zuletzt durch Steueranweisungen eingestellte Schriftgröße dividiert. Beispiel: In der Klammer wurde auf 5-Punkt-Schrift umgeschaltet, um einige verkleinerte Zeichen zu setzen. Diese Zeichen haben zusammen eine Dicke von 137 Bildlinien. Die zur Berechnung der Zeilenlänge notwendige Angabe lautet dann: +137\*05

Die Angaben zur Dickenberechnung müssen als letzte in der Klammer stehen und auch für Bildlinienzahl 00 gemacht werden. Ist eine Multiplikation wie eben beschrieben notwendig, so muss die entsprechende Angabe die letzte in der Klammer sein.

- &! ( . . )**      Rückschalten der Schriftgröße auf den vor der Klammer zuletzt durch Steueranweisungen eingestellten Wert
- &! ( { { )**      Rückschalten des Auszeichnungszustandes auf den zuletzt vor der Klammer durch Steueranweisungen erreichten Wert
- &! ( : nn )**      Verschieben des Schreibstrahls nach oben um nn (zweistellig) Halbpunktschritte (absolut, nicht addierend mit vorhergehenden, auf diese Weise angegebenen Werten)
- &! ( : )**          wie &! ( : 00 ) = Verschieben des Schreibstrahls zurück auf die Schriftgrundlinie
- &! ( : Dnnn )**    Wie &! ( : nn ) , jedoch ist nnn eine dreistellige Dezimalzahl von 000 bis 255
- &! ( : : nn )**      Verschieben des Schreibstrahls nach oben um die Zahl von Halbpunktschritten, die sich aus nn/32 der Gevierthöhe errechnet; sonst wie &! ( : nn )
- &! ( : : Dnnn )**    Wie &! ( : : nn ) , jedoch ist nnn eine dreistellige Dezimalzahl von 000 bis 255
- &! ( ; nn )**        Verschieben des Schreibstrahls nach unten um nn+1 Halbpunktschritte (absolut, nicht addierend mit vorhergehenden, auf diese Weise angegebenen Werten; nn ist zweistellig zu schreiben)
- &! ( ; Dnnn )**     Wie &! ( ; nn ) , jedoch ist nnn eine dreistellige Dezimalzahl von 000 bis 255
- &! ( ; ; nn )**      Verschieben des Schreibstrahls nach unten um einen halben Punkt plus die Zahl von Halbpunktschritten, die sich aus nn/32 der Gevierthöhe errechnet; sonst wie &! ( ; ; nn )
- &! ( ; ; Dnnn )**    Wie &! ( ; ; nn ) , jedoch ist nnn eine dreistellige Dezimalzahl von 000 bis 255



Die Grundstellung wird wieder erreicht durch `&! (:00);` oder `&! (:)`; dieser Zustand sollte vor Verlassen der Klammer wieder hergestellt werden.

**&! (@b)**

Kennzeichnung einer hardware-nahen Codierung als Buchstabe

Als erste innerhalb der Klammer verwendet, macht die Zeichenfolge `@b` (`b` ist ein beliebiger Kleinbuchstabe) den Klammersausdruck für das Silbentrennprogramm und für evtl. Sperrungen und Akzente mit dem angegebenen Buchstaben gleichwertig.

Dies sollte z. B. verwendet werden, wenn innerhalb von längeren Wörtern Sonderformen von Buchstaben vorkommen, die durch Makro codiert und dann in Form von Hardware-Anweisungen angegeben werden, da diese Wörter sonst in der Umgebung dieses Makros weder getrennt noch gesperrt werden können.

## Erweiterungen des 50T1-Codes:

<code>&amp;! (2800)</code>	Umschaltung auf Primäradressenbereich 192–255
<code>&amp;! (2801)</code>	Video aus für das nächste Zeichen
<code>&amp;! (2802)</code>	Bild nachtragen
<code>&amp;! (2803uullnn)</code>	PostScript-Bild ( <code>schru schrl nr</code> ) einblenden
<code>&amp;! (2803uullD255)</code>	für <code>glyphshow</code> auf Font ( <code>schru schrl nr</code> ) umschalten; Font davor merken für automatische Rückschaltung nach <code>glyphshow</code>
<code>&amp;! (2804nn)</code>	Unterlegung <code>nn</code> Anfang bei <code>nn &gt; 10</code> : <code>ashow</code> -Anfang mit $(nn-10)/50$ Geviert bei <code>nn = 10</code> : <code>ashow</code> -Ende (für gesperrte Punkte-Linie <code>&amp;! . (a,b,n)</code> genutzt)
<code>&amp;! (2805nn)</code>	Unterlegung <code>nn</code> Ende
<code>&amp;! (2806ccmmyykk)</code>	Color <code>cmyk</code> Anfang (Farbwerte <code>cc mm yy kk</code> )
<code>&amp;! (2806ccmmyykkppiinn)</code>	bei <code>kk &gt; 128</code> : Color <code>cmyk</code> Anfang für Farbseparation: <code>pp</code> = Prozent, <code>ii</code> = HKS-Nummer, <code>nn</code> = Material
<code>&amp;! (2807)</code>	Color Ende
<code>&amp;! (2808)</code>	Video ein
<code>&amp;! (2809)</code>	Video aus
<code>&amp;! (2810000000)</code>	für PostScript [ ] 0 <code>setdash</code>
<code>&amp;! (28100000sswwoo)</code>	für PostScript [ <code>ss ww</code> ] <code>oo setdash</code>
<code>&amp;! (2810ppqq00gg)</code>	Rechteck grau (Grauwert <code>gg</code> ) zwischen Merkpositionen <code>pp</code> und <code>qq</code> ( <code>pp=00</code> : aktuelle Position)
<code>&amp;! (2810ppqqd25500ccmmyykk)</code>	Rechteck farbig (Farbwerte <code>cc mm yy kk</code> )
<code>&amp;! (2810ppqqd25500ccmmyykk %nnmm)</code>	Rechteck farbig (Farbwerte <code>cc mm yy kk</code> mit $kk \geq 128$ und <code>%%</code> für Deckungsgrad, <code>nn</code> für HKS-Farbnummer, <code>mm</code> für Material)
<code>&amp;! (2811yyyyl111nn)</code>	Vertikale Linie von vertikaler Position <code>yyyy</code> (Viertelpunkt von Satzspiegeloberkante), <code>l111/8</code> Punkt lang und <code>nn/8</code> Punkt breit
<code>&amp;! (2812ppl111nn)</code>	Vertikale Linie von Merkposition <code>pp</code> , <code>l111/8</code> Punkt lang und <code>nn/8</code> Punkt breit

- &! (2812D2531111aaaawwwnn) Horizontale Wellenlinie von aktueller Position, 1111/8 Punkt lang und nn/8 Punkt breit, mit Auslenkung aaaa und halber Wellenlänge www
- &! (2812D254yyyy1111aaaawwwnn) Vertikale Wellenlinie von vertikaler Position yyyy (Viertelpunkt von Satzspiegeloberkante), 1111/8 Punkt lang und nn/8 Punkt breit, mit Auslenkung aaaa und halber Wellenlänge www
- &! (2812D2551111aaaawwwnn) Vertikale Wellenlinie von aktueller Position, 1111/8 Punkt lang und nn/8 Punkt breit, mit Auslenkung aaaa und halber Wellenlänge www
- &! (2813D254nnnn) Satzspiegel um nnnn Punkt nach rechts verschieben
- &! (2813D255nnnn) Satzspiegel um nnnn Punkt nach links verschieben
- &! (2813ppqqnn) Linie nn/8 Punkt breit, zwischen Merkpositionen pp und qq (pp=00: aktuelle Position)
- &! (28130000gg) Definiton von XYgrad für Akkoladen (gg = Gradangabe für die Kreisbögen am Ende der Linie, 6–120 Grad)
- &! (2813ppqqnn11ww) Linie, nn/8 Punkt breit, zwischen Merkpositionen pp-128 und qq-128, mit Pfeilspitzen der Länge 11/8 Punkt und der Breite  $\text{mod}(ww, 128) / 8$  Punkt. Ist pp oder qq kleiner als 128, so wird an diesem Ende der Linie keine Pfeilspitze gezeichnet. Ist ww größer als 128, ist die Pfeilspitze nicht ausgefüllt, sondern offen.
- &! (2813D128qqnn11ww) Linie, nn/8 Punkt breit, zwischen aktueller Position und Merkposition qq, mit Pfeilspitzen der Länge 11/8 Punkt und der Breite  $\text{mod}(ww, 128) / 8$  Punkt. Ist statt D128 00 angegeben oder ist qq kleiner als 128, so wird an diesem Ende der Linie keine Pfeilspitze gezeichnet. Ist ww größer als 128, ist die Pfeilspitze nicht ausgefüllt, sondern offen.
- &! (281300qqnn11ww) bei ll = 0 oder 1: Akkolade, nn/8 Punkt breit, zwischen aktueller Position und Merkposition qq, an beiden Enden Kreisbogen mit Radius  $\text{mod}(ww, 128) / 8$  Punkt. Der Kreisbogen wird durch vorangehendes &! (28130000gr) mit gr von 6–120 Grad definiert. Der Kreisbogen zeigt bei ll = 0 nach links, bei ll = 1 nach rechts.
- &! (2813pp0000) Positionieren auf vertikale Position pp
- &! (2814pp) aktuelle Position als pp merken
- &! (2815nn) Größe nn/4 Punkt
- &! (2819nn) für &! Tn+ = Textfeld-Anfang
- &! (2820nn) für &! Tn- = Textfeld-Ende
- &! (2821nn) für &! Tn. = Textfeld-Ausgabe
- &! (2822) für &! | = Vertikal spiegeln Anfang

&! (2823)	für &!  = Spiegeln/Drehen Ende
&! (2824001111nn)	waagerechte Linie von aktueller Position, 1111/8 Punkt lang und nn/8 Punkt dick
&! (2825nn)	Schriftumschaltung auf Bereich nn
&! (2826)	Horizontal spiegeln, Unterkante = Grundlinie
&! (2827)	Horizontal spiegeln, Lage wie Original
&! (2828)	um 180 Grad drehen, Unterkante = Grundlinie
&! (2829)	um 180 Grad drehen, Lage wie Original
&! (2831nn)	Negativer Zeilenvorschub
&! (2832nnnn)	nnnn/8 p Abstand von oben für nachfolgende Linie bzw. Rechteck
&! (2833nn)	Dickenänderung auf nn/4 Punkt
&! (2842nn)	Schrägstellung um nn Grad nn von 00 bis 45; bei negativem nn (Neigung nach links): 256+nn, also z. B. &! (2842D226) für -30 Grad
&! (2846mmnn)	Schreibstrahl rückwärts um mm + nn (mm und nn < 256)
&! (284701)	Linksläufige Ausgabe Anfang
&! (284700)	Linksläufige Ausgabe Ende
&! (284801)	Linksläufigkeit unterbrechen Anfang
&! (284800)	Linksläufigkeit unterbrechen Ende
&! (2853uu11)	Kashida, Länge = uu*256+11 .
&! (2854)	Unterstreichung Anfang
&! (2855nn)	Unterstreichung nn Ende
&! (2863)	Wiederholen
&! (2865yyynnzzzz)	pdfmark /Launch; yy = Oberkante d. Schaltfläche, nnzzzz = nn Zeichen zz für Pfadname
&! (2866yyynnzzzz)	pdfmark /Launch; yy = Oberkante d. Schaltfläche, nnzzzz = nn Zeichen zz für Uri-Name
&! (28xxyyynnzzzz)	pdfmark /Note; xx = 66 + b, wobei b = laufende Nummer (im Alphabet) des Buchstabens, der die Art der Notiz kennzeichnet (B, C, E, G, H, K, L, N, P, Q, S, T, Z), und nnzzzz = nn Zeichen zz für r,g,b:::titel:::text bzw. r,g,b;;;titel;;;text
&! (28D201)	Beginn Zeichenname für glyphshow
&! (28D202)	Ende Zeichenname für glyphshow
&! (D25536)	kürzere Form für &! (00D2500036)
&! (D255D25536)	kürzere Form für &! (286300020036)

### Zeichencodierung in der Ausgabedatei

Für die Zeichencodierung in der Ausgabedatei gilt die folgende Tabelle.

Sie folgt im Wesentlichen den Konventionen, die für den von #SATZ ursprünglich benutzten Digiset 50T1 (s. o. S. 1170) galten.

TUSTEP Satz: Zeichencodierung in der Ausgabedatei (nach DIGISET 50T1)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
00	SP				‰	§	"	"	"	<				Ä		1	00
01	1	ü	j	a	;	†	J	A		>	-	˘	'	Ö		2	01
02	2	s	k	b	?	S	K	B		ö	.	˘	"	Ü		3	02
03	3	t	l	c	!	T	L	C		ƶ	˘	-	.	£	đ	4	03
04	4	u	m	d	`	U	M	D		ƒ	,	.		\$	ł	5	04
05	5	v	n	e	'	V	N	E		^	.	˘		ø		6	05
06	6	w	o	f	>	W	O	F		'	,			þ		7	06
07	7	x	p	g	<	X	P	G		`	=		-	ð		8	07
08	8	y	q	h	«	Y	Q	H		"	×			Ł		9	08
09	9	z	r	i	»	Z	R	I		^	°	+	-	∅		0	09
0A	0				(	→				˘	˘		-	Ɔ			0A
0B	:	,	æ	.	)	<	Æ	↔	@	˘	˘	"	-	}	ß		0B
0C	-		œ	-	[		€	ı	{	˘	'	˘					0C
0D	ä		ß	-	]	›	&	i	}	˘	˘	˘		ff			0D
0E	ö				/	ƒ	≅	3	\	˘	˘	˘		fi			0E
0F				*	‰	˘		¿	#	˘	˘	˘		fl			0F
	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	

Akzentzeichen AB, AC, AD, B1-BF, D6, D8, FD: Versal-Akzente  
 0C = Divis, 3C = Halbgeviert, 3D = Geviert, B9 = Minus,  
 D9 = #.) DA = #.( F0-F9 = Alternativziffern

## Anhang

Nach Codes geordnete Übersicht über die Steueranweisungen  
(Eingabecodierung)

(Sortierfolge: !"#\$%&'()\*+,-./:;<=>?@[\\]^\_`{|}0..9A..Z)

!	! Ausrufezeichen (12.7.)
!!(n)	n-te Schrift Anfang (11.1.2.)
!!(n){	n-te Schrift Ende, Rückschalten auf gemerkte Schrift (11.1.2.)
"	» « Anführungs- und Schlusszeichen (12.7.)
""	sechste Schrift Anfang (11.1.2.)
""{	sechste Schrift Ende, Rückschalten auf gemerkte Schrift (11.1.2.)
#!x	$\underset{x}{u}$ untergesetztes kleineres Zeichen x (12.6.)
#"*	(hebräisch:) Kreis über Buchstabe (15.)
#+	(hebräisch:) Kreuz über Buchstabe (15.)
#,	(hebräisch:) Mätäg / Silluq (15.)
#"–	(hebräisch:) Rafe (15.)
#".	(hebräisch:) Rebia' (15.)
#"<	(hebräisch:) Atnach (15.)
#"=	(hebräisch:) doppelter Rafe (15.)
#">	(hebräisch:) Haken über Buchstabe (15.)
#"0	(hebräisch:) Schwa (15.)
#"0#"3	(hebräisch:) Chataf Segol (15.)
#"0#"4	(hebräisch:) Chataf Kamaz (15.)
#"0#"5	(hebräisch:) Chataf Patach (15.)
#"1	(hebräisch:) Chirek (15.)
#"2	(hebräisch:) Zere (15.)
#"3	(hebräisch:) Segol (15.)
#"3#"0	(hebräisch:) Chataf Segol (15.)
#"4	(hebräisch:) Kamaz (15.)
#"4#"0	(hebräisch:) Chataf Kamaz (15.)
#"5	(hebräisch:) Patach (15.)
#"5#"0	(hebräisch:) Chataf Patach (15.)
#"6	(hebräisch:) Kubuz (15.)
#"7	(hebräisch:) Cholem (15.)
#"8	(hebräisch:) Schin (15.)
#"9	(hebräisch:) Sin (15.)
##	fünfte Schrift Anfang (11.1.2.)
##{	fünfte Schrift Ende, Rückschalten auf gemerkte Schrift (11.1.2.)
#'x	$\overset{x}{x}$ hochgestelltes kleineres Zeichen x (12.3.)
#(E)	Euro-Zeichen (12.8.1.)
#(name)	mit #(name) codierte Sonderzeichen (12.8.1.)
#,1	1 tiefgestellte Ziffer 1 (12.5.)
#,x	$\overset{x}{x}$ tiefgestelltes kleineres Zeichen x (12.5.)
#-x	$\underset{x}{x}$ kleineres Zeichen auf Schriftgrundlinie (12.5.)
#.!	§ Paragraphzeichen (12.7.)

#. ”	”	Anführungszeichen »99« oben (12.7.)
#. %	›	Doppel-Alef (12.2.)
#. ’	“	Anführungszeichen »66« oben (12.7.)
#. (	˘	Ajin (12.2.)
#. )	›	Alef (12.2.)
#. *	◦	Grad (12.7.)
#. +	+	plus (veraltet für +) (12.7.)
#. ,	„	Anführungszeichen unten (12.7.)
#. –	’	Minute (12.7.)
#. .	→	Verweispeil (12.7.)
#. /		doppelter senkrechter Strich (12.7.)
#. :	›	Einfaches Anführungszeichen links (12.7.)
#. ;	˘	Einfaches Anführungszeichen rechts (12.7.)
#. <	“	französisches Anführungszeichen rechts (12.7.)
#. =	”	Sekunde (12.7.)
#. >	»	französisches Anführungszeichen links (12.7.)
#. ?	›˘	einfaches Anführungszeichen (12.7.)
#. [	⌈	Winkel oben links (12.7.)
#. \	⋈	Fraktur-Divis (12.7.)
#. ]	⌋	Winkel oben rechts (12.7.)
#. ^ ^	^	Zirkumflex, Dach (nicht als Akzent) (12.2.)
#. _		Spatium im Wort (12.7.)
#. {	{	geschweifte Klammer auf (12.7.)
#. }	}	geschweifte Klammer zu (12.7.)
#. ^a	æ	Ligatur ae (12.2.)
#. ^A	Æ	Ligatur AE (12.2.)
#. ^d	ð	eth (12.2.)
#. ^D	Ð	Eth (12.2.)
#. ^o	œ	Ligatur oe (12.2.)
#. ^O	Œ	Ligatur OE (12.2.)
#. ^S	ß	scharfes S (12.2.)
#. ^u	ױ	(hebräisch:) Ligatur Waw Jud (14.)
#. 1	ı	Alternativziffern (12.2.)
#. b	b	b mit Querstrich (12.2.)
#. d	đ	(serbokroat.) d mit Querstrich (12.2.)
#. D	Đ	(serbokroat.) D mit Querstrich (12.2.)
#. h	ħ	(maltesisch) h mit Querstrich (12.2.)
#. H	Ħ	(maltesisch) H mit Querstrich (12.2.)
#. i	ı	ohne Punkt (12.2.);
	ײ	(hebräisch:) Ligatur Jud Jud (14.)
#. j	ij	Ligatur ij (12.2.)
#. J	IJ	Ligatur IJ (14.)
#. k	ϙ	(griechisch:) koppa (13.)
#. L	Ł	(polnisch) L mit Querstrich (12.2.)
#. l	ł	(polnisch) l mit Querstrich (12.2.)
#. O	Ø	(dän.-norw.) O mit Schrägstrich (12.2.)
#. o	ø	(dän.-norw.) o mit Schrägstrich (12.2.)
#. p	þ	(isländ.) Thorn (12.2.); Ϡ (griechisch:) sampi (13.)

#.P	Þ (isländ.) Thorn (12.2.)
#.s	ſ langes s; Fraktur: rundes s (12.2.);
#.ß	ß scharfes s; (12.2.); ς (griechisch:) stigma (13.)
#.T	Ƨ (samisch) T mit Querstrich (12.2.)
#.t	Ƨ (samisch) t mit Querstrich (12.2.)
#.u	𐤀 (hebräisch:) Ligatur Waw Waw (14.)
#.w	Ϝ (griechisch:) Digamma (13.)
#.z	ꝛ langes z, yogh (12.2.)
#.Z	Ꝝ Yogh (12.2.)
#/( ) -	Schrägstellung Ende (11.1.1.)
#/(nn) +	Schrägstellung Anfang (11.1.1.)
#/(nn) -	Schrägstellung Ende (11.1.1.)
#/+	kursiv Anfang (11.1.1.)
#/-	kursiv Ende (11.1.1.)
#;x	ũ übergesetztes kleineres Zeichen x (12.4.)
#?+	Auszeichnungen bei Abschnittsgrenze nicht aufheben (11.2.)
#?-	Unterstreichug etc. Ende (11.2.)
#[2000]	( <i>en quad</i> ) (12.8.5.)
#[2001]	( <i>em quad</i> ) (12.8.5.)
#[2002]	( <i>en space</i> ) (12.8.5.)
#[2003]	( <i>em space</i> ) (12.8.5.)
#[2004]	( <i>three-per-m-space</i> ) (12.8.5.)
#[2005]	( <i>four-per-m-space</i> ) (12.8.5.)
#[2006]	( <i>six-per-m-space</i> ) (12.8.5.)
#[2007]	( <i>figure space</i> ) (12.8.)
#[2008]	( <i>punctuation space</i> ) (12.8.5.)
#[2009]	( <i>thin space</i> ) (12.8.5.)
#[200A]	( <i>hair space</i> ) (12.8.5.)
#[2010]	( <i>hyphen</i> ) (12.8.5.)
#[2011]	( <i>non-breaking hyphen</i> ) (12.8.5.)
#[2012]	( <i>figure dash</i> ) (12.8.5.)
#[2013]	( <i>en dash</i> ) (12.8.5.)
#[2014]	( <i>em dash</i> ) (12.8.5.)
#[2016]	( <i>double vertical line</i> ) (12.8.5.)
#[2018]	( <i>left single quotation mark</i> ) (12.8.5.)
#[2019]	( <i>right single quotation mark</i> ) (12.8.5.)
#[201A]	( <i>single low-9 quotation mark</i> ) (12.8.5.)
#[201B]	( <i>single high-reversed-9 quotation mark</i> ) (12.8.5.)
#[2021]	( <i>double dagger</i> ) (12.8.5.)
#[202F]	( <i>narrow no-break space</i> ) (12.8.5.)
#^0+	Unterstreichug mit beliebigen Zeichen Anfang (11.2.)
#^0-	Unterstreichug mit beliebigen Zeichen Ende (11.2.)
#^1+	Unterstreichug mit beliebigen Zeichen Anfang (11.2.)
#^1-	Unterstreichug mit beliebigen Zeichen Ende (11.2.)
#^2+	Unterstreichug mit beliebigen Zeichen Anfang (11.2.)
#^2-	Unterstreichug mit beliebigen Zeichen Ende (11.2.)
#^3+	Unterstreichug mit beliebigen Zeichen Anfang (11.2.)

---

#^3-	Unterstreichung mit beliebigen Zeichen Ende (11.2.)
#^4+	Unterstreichung mit beliebigen Zeichen Anfang (11.2.)
#^4-	Unterstreichung mit beliebigen Zeichen Ende (11.2.)
#^5+	Unterstreichung mit beliebigen Zeichen Anfang (11.2.)
#^5-	Unterstreichung mit beliebigen Zeichen Ende (11.2.)
#^6+	Unterstreichung mit beliebigen Zeichen Anfang (11.2.)
#^6-	Unterstreichung mit beliebigen Zeichen Ende (11.2.)
#^7+	Unterstreichung mit beliebigen Zeichen Anfang (11.2.)
#^7-	Unterstreichung mit beliebigen Zeichen Ende (11.2.)
#^8+	Unterstreichung mit beliebigen Zeichen Anfang (11.2.)
#^8-	Unterstreichung mit beliebigen Zeichen Ende (11.2.)
#^9+	Unterstreichung mit beliebigen Zeichen Anfang (11.2.)
#^9-	Unterstreichung mit beliebigen Zeichen Ende (11.2.)
#0+	Durchstreichung Anfang (11.2.)
#0-	Durchstreichung Ende (11.2.)
#1+	einfache Unterstreichung Anfang (11.2.)
#1-	einfache Unterstreichung Ende (11.2.)
#10+...#10-	Kombination von #1+...#1- und #0+...#0- *107 (11.2.)
#2+	doppelte Unterstreichung Anfang (11.2.)
#2-	doppelte Unterstreichung Ende (11.2.)
#20+...#20-	Kombination von #2+...#2- und #0+...#0- *107 (11.2.)
#3+	halbfette Unterstreichung Anfang (11.2.)
#3-	halbfette Unterstreichung Ende (11.2.)
#30+...#30-	Kombination von #3+...#3- und #0+...#0- *107 (11.2.)
#4+	Unterpunktierung Anfang (11.2.)
#4-	Unterpunktierung Ende (11.2.)
#49+...#49- -	Kombination von #4+...#4- und #9+...#9- *107 (11.2.)
#5+	tiefe Überstreichung Anfang (11.2.)
#5-	tiefe Überstreichung Ende (11.2.)
#50+...#50-	Kombination von #5+...#5- und #0+...#0- *107 (11.2.)
#51+...#51-	Kombination von #5+...#5- und #1+...#1- *107 (11.2.)
#52+...#52-	Kombination von #5+...#5- und #2+...#2- *107 (11.2.)
#53+...#53-	Kombination von #5+...#5- und #3+...#3- *107 (11.2.)
#6+	hohe Überstreichung Anfang (11.2.)
#6-	hohe Überstreichung Ende (11.2.)
#60+...#60-	Kombination von #6+...#6- und #0+...#0- *107 (11.2.)
#61+...#61-	Kombination von #6+...#6- und #1+...#1- *107 (11.2.)
#62+...#62-	Kombination von #6+...#6- und #2+...#2- *107 (11.2.)
#63+...#63-	Kombination von #6+...#6- und #3+...#3- *107 (11.2.)
#7+	helle Unterlegung Anfang (11.2.)
#7-	helle Unterlegung Ende (11.2.)
#8+	dunkle Unterlegung Anfang (11.2.)
#8-	dunkle Unterlegung Ende (11.2.)
#9+	tiefe Unterstreichung Anfang (11.2.)
#9-	tiefe Unterstreichung Ende (11.2.)
#A+	Arabisch Anfang (11.1.1.)
#A-	Arabisch Ende (11.1.1.)
#B+	Basis-Schnitt Anfang (11.1.1.)



#B-	Basis-Schnitt Ende (11.1.1.)
#D+	Fraktur (»Deutsche Schrift«) Anfang (11.1.1.)
#D-	Fraktur (»Deutsche Schrift«) Ende (11.1.1.)
#F+	halbfett Anfang (11.1.1.)
#F-	halbfett Ende (11.1.1.)
#G (n)	Aufheben von einfacher Hoch- und Tiefstellung (12.3.)
#G+	Griechisch Anfang (11.1.1.)
#G-	Griechisch Ende (11.1.1.)
#G:	Aufheben von einfacher Hoch- und Tiefstellung (12.3.)
#H (n)	einfache Hochstellung um $n/2$ Punkt (12.3.)
#H (:n)	einfache Hochstellung um $n/32$ Geviert (12.3.)
#H+	Hebräisch Anfang (11.1.1.)
#H-	Hebräisch Ende Ende (11.1.1.)
#H:	einfache Hochstellung um $1/3$ Zeile (12.3.)
#i+nnn#i-	Ausgabe der Zahl nnn als römische Zahl (Kleinbuchstaben)
#I+nnn#I-	Ausgabe der Zahl nnn als römische Zahl (Großbuchstaben)
#K+	Kapitälchen Anfang (11.1.1.)
#K-	Kapitälchen Ende (11.1.1.)
#N+	Beginn eines nicht umzudrehenden Textteils (15.)
#N-	Ende eines nicht umzudrehenden Textteils (15.)
#O:	einfache Hochstellung um $1/2$ Zeile (12.3.)
#O (n)	einfache Hochstellung um $n/2$ Punkt (12.3.)
#O (:n)	einfache Hochstellung um $n/32$ Geviert (12.3.)
#P+	Phonetisches Alphabet (IPA) (11.1.1.)
#P-	Phonetisches Alphabet (IPA) Ende (11.1.1.)
#Q+	Kapitälchen Anfang (11.1.1.)
#Q-	Kapitälchen Ende (11.1.1.)
#R+	Russisch Anfang (11.1.1.)
#R-	Russisch Ende (11.1.1.)
#S+	Sperrung Anfang (11.1.1.)
#S*	Sperrung Anfang (11.1.1.)
#S-	Sperrung Ende (11.1.1.)
#S=	Sperrung Ende (11.1.1.)
#T (n)	einfache Tiefstellung um $n/2$ Punkt (12.5.)
#T (;n)	einfache Tiefstellung um $n/32$ Geviert (12.5.)
#T+	Koptisch Anfang (11.1.1.)
#T-	Koptisch Ende (11.1.1.)
#T:	einfache Tiefstellung um $1/3$ Zeile (12.5.)
#U (n)	einfache Tiefstellung um $n/2$ Punkt (12.5.)
#U (;n)	einfache Tiefstellung um $n/32$ Geviert (12.5.)
#U:	einfache Tiefstellung um $1/2$ Zeile (12.5.)
#V+	Versalien Anfang (11.1.1.)
#V-	Versalien Ende (11.1.1.)
#W+	Gemeine Anfang (11.1.1.)
#W-	Gemeine Ende (11.1.1.)
#X+	Austausch Standardziffern / Alternativziffern (11.1.1.)
#X-	Ende Austausch Standardziffern / Alternativziffern (11.1.1.)
#Z (nn) +	Sperrung Anfang (11.1.1.)

#Z (nn) *	Sperrung Anfang (11.1.1.)
#Z+	Sperrung Anfang (11.1.1.)
#Z*	Sperrung Anfang (11.1.1.)
#Z-	Sperrung Ende (11.1.1.)
#Z=	Sperrung Ende (11.1.1.)
\$	Absatz (7.1.)
\$\$	Beginn einer Einschaltung (»Petit-Satz«) (4.)
\$\$!\$\$ {	Freiraum für spaltenhohe Abbildung (7.3.)
\$\$!#iii\$\$ {	Freiraum für spaltenhohe Abbildung (7.3.)
\$\$!-nnn\$\$ {	Freiraum auf nächster Spalte (7.3.)
\$\$!nnn\$\$ {	Freiraum für spaltenbreite Abbildung (7.3.)
\$\$#iii\$\$ {	Einmontieren der Abbildung Nr. iii (7.3.)
\$\$\$	Neue Zeile stumpf (8.)
\$\$\$\$	neue Zeile stumpf, Zeile davor auf Satzbreite austreiben (8.)
\$\$\$\$+n\$\$\$	Zeilenwechsel mit Vorschub von insgesamt n Punkt, keine Abschnittsgrenze; Zeile davor auf Satzbreite austreiben (7.2.)
\$\$\$\$-n\$\$\$	neue Zeile stumpf, bedingter Spaltenwechsel, die Zeile davor auf Satzbreite austreiben (7.1.)
\$\$\$\$n\$\$\$	Zeilenwechsel und zusätzlicher Vorschub von n Punkt, keine Abschnittsgrenze; Zeile davor auf Satzbreite austreiben (7.2.)
\$\$\$\$+n\$\$\$	Zeilenwechsel mit Vorschub von insgesamt n Punkt (7.2.)
\$\$\$\$-n\$\$\$	Neue Zeile stumpf, Abschnittsgrenze, bedingter Spaltenwechsel (7.1.)
\$\$\$/\$\$\$	Zeilenwechsel und 1/2 Leerzeile, Abschnittsgrenze (7.2.)
\$\$\$=\$\$\$	Zeilenwechsel und Leerzeile, Abschnittsgrenze (7.2.)
\$\$\$=n\$\$\$	Zeilenwechsel und n Leerzeilen, Abschnittsgrenze (7.2.)
\$\$\$=n. \$\$\$	Zeilenwechsel und n Punkt Freiraum, Abschnittsgrenze (7.2.)
\$\$\$\\ \$\$\$	Nachfolgende Zeilenwechselanweisung ignorieren (8.)
\$\$\$0\$\$\$	Neue Zeile stumpf, Abschnittsgrenze (7.1.)
\$\$\$n\$\$\$	Zeilenwechsel und zusätzlicher Vorschub von n Punkt, Abschnittsgrenze (7.2.)
\$\$+-0\$\$\$ {	Zeilenwechsel, vertikaler Tabulator für mehr als eine Spalte (7.4.)
\$\$+-0+n\$\$\$ {	Zeilenwechsel, vertikaler Tabulator für mehr als eine Spalte (7.4.)
\$\$+-0-n\$\$\$ {	Zeilenwechsel, vertikaler Tabulator für mehr als eine Spalte (7.4.)
\$\$+-!nnn\$\$\$ {	Zeilenwechsel, vertikaler Tabulator für mehr als eine Spalte (7.4.)
\$\$+-nnn\$\$\$ {	Zeilenwechsel, vertikaler Tabulator für mehr als eine Spalte (7.4.)
\$\$+m: !nnn\$\$\$ {	Spaltenwechsel, vertikaler Tabulator (7.4.)
\$\$+m: nnn\$\$\$ {	Spaltenwechsel, vertikaler Tabulator (7.4.)
\$\$+!nnn\$\$\$ {	Zeilenwechsel, vertikaler Tabulator (7.4.)
\$\$+nnn\$\$\$ {	Zeilenwechsel, vertikaler Tabulator (7.4.)
\$\$-0\$\$\$ {	Vormerkungen für Freiraum löschen (7.3.)
\$\$-mmm/ !nnn\$\$\$ {	Aussparung auf nächster Spalte (7.3.)
\$\$-mmm/ nnn\$\$\$ {	Aussparung auf nächster Spalte (7.3.)

\$\$-mmm/!nnn#iii\$\$ {	Aussparung auf nächster Spalte (7.3.)
\$\$-mmm/nnn#iii\$\$ {	Aussparung auf nächster Spalte (7.3.)
\$\$-mmm/!nnn#iii#jjj\$\$ {	Aussparung auf nächster Spalte (7.3.)
\$\$-mmm/nnn#iii#jjj\$\$ {	Aussparung auf nächster Spalte (7.3.)
\$\$-nnn\$\$ {	Freiraum auf nächster Spalte (7.3.)
\$\$-nnn#iii\$\$ {	Freiraum auf nächster Spalte (7.3.)
\$\$-nnn#iii#jjj\$\$ {	Freiraum auf nächster Spalte (7.3.)
\$\$-s\$\$ {	Freiraum von s Seiten (7.3.)
\$\$/!nnn\$\$ {	Spaltenhöhe verringern für aktuelle Spalte (7.4.)
\$\$/+!nnn\$\$ {	Spaltenhöhe vergrößern für aktuelle Spalte (7.4.)
\$\$/+nnn\$\$ {	Spaltenhöhe vergrößern für aktuelle Spalte (7.4.)
\$\$/-!nnn\$\$ {	Spaltenhöhe verringern für aktuelle und folgende Spalte(n) (7.4.)
\$\$/-nnn\$\$ {	Spaltenhöhe verringern für aktuelle und folgende Spalte(n) (7.4.)
\$\$//nnn\$\$ {	Spaltenhöhe verringern für aktuelle Spalte (7.4.)
\$\$//!nnn\$\$ {	Spaltenhöhe verringern für aktuelle Spalte (7.4.)
\$\$/=nnn\$\$ {	Spaltenhöhe verringern für den Rest des Textes (7.3.)
\$\$/nnn\$\$ {	Spaltenhöhe verringern für aktuelle Spalte (7.4.)
\$\$\!nnn\$\$ {	Freiraum unten für spaltenbreite Abbildung (7.3.)
\$\$\!nnn\$\$ {	Freiraum unten für spaltenbreite Abbildung (7.3.)
\$\$\nnn\$\$ {	Freiraum unten für spaltenbreite Abbildung (7.3.)
\$\$\nnn\$\$ {	Freiraum unten für spaltenbreite Abbildung (7.3.)
\$\$ {	Einschaltung Ende (4.)
\$\$0\$\$ {	Vormerkungen für Freiraum löschen (7.3.)
\$\$0/0#iii\$\$ {	Abbildung Nr. iii in jeder Spalte einmontieren (7.3.)
\$\$0/0#0\$\$ {	Aufheben von \$\$0/0#iii\$\$ { (7.3.)
\$\$0/1#iii\$\$ {	Abbildung Nr. iii in jeder ungeraden Spalte einmontieren (7.3.)
\$\$0/1#0\$\$ {	Aufheben von \$\$0/1#iii\$\$ { (7.3.)
\$\$0/2#iii\$\$ {	Abbildung Nr. iii in jeder geraden Spalte einmontieren (7.3.)
\$\$0/2#0\$\$ {	Aufheben von \$\$0/2#iii\$\$ { (7.3.)
\$\$0/nnn\$\$ {	Freiraum für spaltenbreite Abbildungen (7.3.)
\$\$mmm/!nnn\$\$ {	Aussparung für Abbildungen (7.3.)
\$\$mmm/nnn\$\$ {	Aussparung für Abbildungen (7.3.)
\$\$mmm/!nnn#iii\$\$ {	Aussparung für Abbildungen (7.3.)
\$\$mmm/nnn#iii\$\$ {	Aussparung für Abbildungen (7.3.)
\$\$mmm/!nnn#iii+eee\$\$ {	Aussparung für Abbildungen (7.3.)
\$\$mmm/nnn#iii+eee\$\$ {	Aussparung für Abbildungen (7.3.)
\$\$mmm/nnn#iii+-eee\$\$ {	Aussparung für Abbildungen (7.3.)
\$\$mmm/nnn#iii+-eee/aaa\$\$ {	Aussparung für Abbildungen (7.3.)
\$\$mmm/!nnn#iii+-eee\$\$ {	Aussparung für Abbildungen (7.3.)
\$\$mmm/nnn#iii+-eee/aaa\$\$ {	Aussparung für Abbildungen (7.3.)
\$\$mmm/!nnn#iii#jjj\$\$ {	Aussparung für Abbildungen (7.3.)
\$\$mmm/nnn#iii#jjj\$\$ {	Aussparung für Abbildungen (7.3.)
\$\$mmm/!nnn+\$\$ {	rechtsbündige Aussparung für Abbildungen (7.3.)
\$\$mmm/nnn+\$\$ {	rechtsbündige Aussparung für Abbildungen (7.3.)
\$\$mmm/!nnn+-\$ {	abwechselnd (spaltenweise) rechts- und linksbündige Aus-

	sparung für Abbildungen (7.3.)
\$\$mmm/nnn+-\$\${	abwechselnd (spaltenweise) rechts- und linksbündige Aussparung für Abbildungen (7.3.)
\$\$mmm/!nnn+\$\$\${	abwechselnd (spaltenweise) links- und rechtsbündige
\$\$mmm/nnn+\$\$\${	abwechselnd (spaltenweise) links- und rechtsbündige Aussparung für Abbildungen (7.3.)
\$\$mmm/nnn+=-\$\${	abwechselnd (seitenweise) rechts- und linksbündige Aussparung für Abbildungen (7.3.)
\$\$mmm/nnn=-+\$\$\${	abwechselnd (seitenweise) links- und rechtsbündige Aussparung für Abbildungen (7.3.)
\$\$nnn\$\$\${	Freiraum für spaltenbreite Abbildung (7.3.)
\$\$!nnn#iii\$\$\${	Freiraum für spaltenbreite Abbildung (7.3.)
\$\$nnn#iii+eee\$\$\${	Freiraum für spaltenbreite Abbildung (7.3.)
\$\$nnn#iii+-eee\$\$\${	Freiraum für spaltenbreite Abbildung (7.3.)
\$\$nnn#iii+-eee/aaa\$\$\${	Freiraum für spaltenbreite Abbildung (7.3.)
\$\$nnn#iii-+eee\$\$\${	Freiraum für spaltenbreite Abbildung (7.3.)
\$\$nnn#iii-+eee/aaa\$\$\${	Freiraum für spaltenbreite Abbildung (7.3.)
\$\$nnn#iii=+-eee\$\$\${	Freiraum für spaltenbreite Abbildung (7.3.)
\$\$nnn#iii#jjj\$\$\${	Freiraum für spaltenbreite Abbildungen (7.3.)
\$\$nnn#iii+-eee#jjj+-eee\$\$\${	Freiraum für spaltenbreite Abbildungen (7.3.)
\$\$nnn#iii=+-eee#jjj+-eee\$\$\${	Freiraum für spaltenbreite Abbildungen (7.3.)
%!	(arabisch:) Alif (Akzent) (16.)
%"	Doppelakut (Akzent) (12.1.1.); (arabisch:) Fatha-Tanwīn (16.)
%""	Doppelakut (Akzent) unter dem Buchstaben (12.1.2.); (arabisch:) Kasra-Tanwīn (16.)
%")	(arabisch:) Fatha-Tanwīn + Hamza (16.)
%"<	(arabisch:) Fatha-Tanwīn + Taṣdīd (16.)
%">	(arabisch:) Fatha-Tanwīn + Taṣdīd (16.)
%"]	(arabisch:) Fatha-Tanwīn + Hamza (16.)
%'	Betonungszeichen (Akzent) (12.1.2.); (arabisch:) Ḍamma (16.)
%''	Betonungszeichen (Akzent) unter dem Buchstaben (12.1.2.); (arabisch:) i-Punkte + Kasra-Tanwīn (16.)
%'<	(arabisch:) Ḍamma + Taṣdīd (16.)
%']	(arabisch:) Ḍamma + Hamza (16.)
% (	Bogen (Akzent) (12.1.1.); Spiritus asper (13.); (arabisch:) Sukūn (16.)
% ((	Bogen (Akzent) unter dem Buchstaben (12.1.2.)
% (/	Spiritus asper + Akut (13.)
% (?	Spiritus asper + Zirkumflex (13.)
% (\	Spiritus asper + Gravis (13.)
% ]	(arabisch:) Sukūn + Hamza (16.)
% )	Halbkreis (Akzent) (12.1.1.); Spiritus lenis (13.); (arabisch:) Hamza (16.)
% )"'	(arabisch:) Hamza + Fatha-Tanwīn (16.)
% )=	(arabisch:) Hamza + Fatha-Tanwīn (16.)
% ) )	Halbkreis (Akzent) unter dem Buchstaben (12.1.2.); (arabisch:) Hamza (16.)

◌ /	Spiritus lenis + Akut (13.)
◌ ?	Spiritus lenis + Zirkumflex (13.)
◌ \	Spiritus lenis + Gravis (13.)
◌ *	Kringel (Akzent) (12.1.1.); (arabisch:) Sukūn (16.)
◌ *	(arabisch:) Sukūn + Hamza (16.)
◌ **	Kringel (Akzent) unter dem Buchstaben (12.1.2.)
◌ * ]	(arabisch:) Sukūn + Hamza (16.)
◌ +	Plus (Akzent) (12.1.1.); (arabisch:) Madda (16.)
◌ ++	Plus (Akzent) unter dem Buchstaben (12.1.2.)
◌ ,	Komma (Akzent) (12.1.1.); (arabisch:) Ḍamma (16.)
◌ , )	(arabisch:) Ḍamma + Hamza (16.)
◌ , ,	Komma (Akzent) unter dem Buchstaben (12.1.2.) ; (arabisch:) i-Punkte + Kasra (16.)
◌ , <	(arabisch:) Ḍamma + Tašdīd (16.)
◌ , >	(arabisch:) Ḍamma + Tašdīd (16.)
◌ , ]	(arabisch:) Ḍamma + Hamza (16.)
◌ -	Querstrich, Balken (Akzent) (12.1.1.); (arabisch:) Waṣla (16.)
◌ --	Querstrich, Balken (Akzent) unter dem Buchstaben (12.1.2.)
◌ - ]	back sign, subscript (12.1.2.)
◌ -	raising sign, subscript (12.1.2.)
◌ .	Punkt (Akzent) (12.1.1.); (hebräisch:) Punkt über Buchstabe (Zahlzeichen) (15.); (arabisch:) Waṣla (16.)
◌ . .	Punkt (Akzent) unter dem Buchstaben (12.1.2.)
◌ /	Akut (Akzent) (12.1.1.); (arabisch:) Faṭḥa (16.)
◌ / )	(arabisch:) Faṭḥa + Hamza (16.)
◌ / :	(griechisch:) Akut + Trema (13.)
◌ / >	(arabisch:) Faṭḥa + Tašdīd (16.)
◌ / <	(arabisch:) Faṭḥa + Tašdīd (16.)
◌ / /	Akut (Akzent) unter dem Buchstaben (12.1.2.); (arabisch:) Kasra (16.)
◌ / ]	(arabisch:) Faṭḥa + Hamza (16.)
◌ :	Trema (Akzent) (12.1.1.); (arabisch:) Ḍamma-Tanwīn (16.)
◌ : )	(arabisch:) Ḍamma-Tanwīn + Hamza (16.)
◌ ::	Trema (Akzent) unter dem Buchstaben (12.1.2.)
◌ : <	(arabisch:) Ḍamma-Tanwīn + Tašdīd (16.)
◌ : >	(arabisch:) Ḍamma-Tanwīn + Tašdīd (16.)
◌ : ]	(arabisch:) Ḍamma-Tanwīn + Hamza (16.)
◌ ;	Cedille (12.1.2.); (griechisch:) iota subscriptum (13.); (arabisch:) Hamza + Kasra (16.)
◌ ; ;	Krummhaken (12.1.2.) Arabisch: i-Punkte (16.)
◌ <	Zirkumflex, Dach (Akzent) (12.1.1.); (arabisch:) Tašdīd (16.)
◌ < \	(arabisch:) Tašdīd + Faṭḥa (16.)
◌ < <	Zirkumflex, Dach (Akzent) unter dem Buchstaben (12.1.2.)
◌ < =	(arabisch:) Tašdīd + Faṭḥa-Tanwīn (16.)
◌ =	Kreuz (Akzent) (12.1.1.); (arabisch:) Faṭḥa-Tanwīn (16.)
◌ ==	Kreuz (Akzent) unter dem Buchstaben (12.1.2.); (arabisch:) Kasra-Tanwīn (16.)

◌=<	(arabisch:) Fatḥa-Tanwīn + Tašdīd (16.)
◌=]	(arabisch:) Fatḥa-Tanwīn + Hamza (16.)
◌>	Haček, Haken (Akzent) (12.1.1.); (arabisch:) Tašdīd (16.)
◌>/	(arabisch:) Tašdīd + Fatḥa (16.)
◌>“	(arabisch:) Tašdīd + Fatḥa-Tanwīn (16.)
◌>=	(arabisch:) Tašdīd + Fatḥa-Tanwīn (16.)
◌>>	Haček, Haken (Akzent) unter dem Buchstaben (12.1.2.)
◌>\	(arabisch:) Tašdīd + Fatḥa (16.)
◌?	Tilde (Akzent) (12.1.1.); (arabisch:) Madda (16.)
◌???	Tilde (Akzent) unter dem Buchstaben (12.1.2.)
◌[	Brücke (Akzent) (12.1.1.); (arabisch:) Damma-Tanwīn (16.)
◌[-	front sign, subscript (12.1.2.)
◌[<	(arabisch:) Damma-Tanwīn + Tašdīd (16.)
◌[[	Brücke (Akzent) unter dem Buchstaben (12.1.2.)
◌[[	laminal sign, subscript (12.1.2.); (arabisch:) Damma-Tanwīn + Hamza (16.)
◌\	Gravis (Akzent) (12.1.1.); (arabisch:) Fatḥa (16.)
◌\:	(griechisch:) Gravis + Trema (13.)
◌\<	(arabisch:) Fatḥa + Tašdīd (16.)
◌\\	Gravis (Akzent) unter dem Buchstaben (12.1.2.); (arabisch:) Kasra (16.)
◌\]	(arabisch:) Fatḥa + Hamza (16.)
◌]	umgekehrte Brücke (Akzent) (12.1.1.); (arabisch:) Hamza (16.)
◌]=	(arabisch:) Hamza + Fatḥa-Tanwīn (16.)
◌]]	umgekehrte Brücke (Akzent) unter dem Buchstaben (12.1.2.); (arabisch:) Hamza unter dem Buchstaben (16.)
◌^^	Zirkumflex, Dach (Akzent) unter dem Buchstaben (12.1.2.)
◌{	Halbkreis rechts offen (Akzent) (12.1.1.)
◌{{	Halbkreis rechts offen (Akzent) unter dem Buchstaben (12.1.2.)
◌	(arabisch:) Alif (Akzent) (16.)
◌ -	lowering sign, subscript (12.1.2.)
◌}	Halbkreis links offen (Akzent) (12.1.1.); (arabisch:) Hamza + Kasra unter Ligatur (16.)
◌}}}	Halbkreis links offen (Akzent) unter dem Buchstaben (12.1.2.)
◌1	<sup>1</sup> (hochgestellte) Fußnotenziffer (12.3.)
◌a	<sup>a</sup> hochgestellter Buchstabe in Fußnotennummern (12.3.)
&	Titelzeile Stufe 1 (3.)
&!%(nnn)	Dicktenwert für Verlängerung des Querstrichs über/unter Buchstaben (12.1)
&! (# # m m m m m / n n n)	PostScript-Zeichen Nummer nnn aus Font m m m m m (12.8.4.)
&! (# # / n n n)	PostScript-Zeichen Nummer nnn aus aktuellem Font (12.8.3.)

&! (#Dn)	Definieren eines benannten Zieles (21.2.1.)
&! (#Dn/**y)	Definieren eines benannten Zieles (21.2.1.)
&! (#Dn/z)	Definieren eines benannten Zieles (21.2.1.)
&! (#Giii)	PostScript-Grafik (12.9.)
&! (#Gn/iii)	PostScript-Grafik (12.9.)
&! (#Gn/Hiiii)	PostScript-Grafik (12.9.)
&! (#L)	Ende einer Schaltfläche (21.1.–21.5.)
&! (#L/d:n)	Anfang einer Schaltfläche, Verknüpfung mit Seite n der Datei d (21.1.)
&! (#L/n)	Anfang einer Schaltfläche, Verknüpfung mit Seite n (21.1.)
&! (#Ly)	Anfang einer Schaltfläche für eine Verknüpfung (21.1.)
&! (#Ly/d:n/z)	Anfang einer Schaltfläche, Verknüpfung mit Seite n der Datei d (21.1.)
&! (#Ly/n/z)	Anfang einer Schaltfläche, Verknüpfung mit Seite n (21.1.)
&! (#LD/d:n)	Anfang einer Schaltfläche, Verknüpfung mit Seite n der Datei d (21.2.2.)
&! (#LD/n)	Anfang einer Schaltfläche, Verknüpfung mit dem benannten Ziel n (21.2.2.)
&! (#LDy/d:n)	Anfang einer Schaltfläche, Verknüpfung mit dem benannten Ziel n in der Datei d (21.2.2.)
&! (#LDy/n)	Anfang einer Schaltfläche, Verknüpfung mit dem benannten Ziel n (21.2.2.)
&! (#Nxy/farbe::titel::text)	Einfügen einer Notiz (21.5.)
&! (#Nxy/farbe;;;titel;;;text)	Einfügen einer Notiz (21.5.)
&! (#PL/pfad)	Anfang einer Schaltfläche, Aufruf eines anderen
&! (#PLy/pfad)	Anfang einer Schaltfläche, Aufruf eines anderen Dokuments oder eines Programms (21.4.)
&! (#PU/uri)	Anfang einer Schaltfläche, Verknüpfung mit einem WWW-Dokument (21.3.)
&! (#PUy/uri)	Anfang einer Schaltfläche, Verknüpfung mit einem WWW-Dokument (21.3.)
&! (#Siii)	PostScript-Grafik (12.9.)
&! (#Sn/iii)	PostScript-Grafik (12.9.)
&! (#Sn/Hiiii)	PostScript-Grafik (12.9.)
&! (#Ummmm\zeichename\dicke)	PostScript-Sonderzeichen zeichename (12.8.4.)
&! (+nn)	Verschieben des Schreibstrahls nach rechts (22.)
&! (+nn=)	wie &! (+nn), jedoch nur auf linken Seiten (22.)
&! (+nn/)	wie &! (+nn), jedoch nur auf rechten Seiten (22.)
&! (++nn)	Verschieben des Schreibstrahls nach rechts (22.)
&! (++nn=)	wie &! (++nn), jedoch nur auf linken Seiten (22.)
&! (++nn/)	wie &! (++nn), jedoch nur auf rechten Seiten (22.)
&! (-nn)	Verschieben des Schreibstrahls nach links (22.)
&! (--nn)	Verschieben des Schreibstrahls nach links (22.)
&! (..)	Rückschalten der Schriftgröße (22.)
&! (:)	Verschieben des Schreibstrahls zurück (22.)
&! (:nn)	Verschieben des Schreibstrahls nach oben (22.)
&! (;nn)	Verschieben des Schreibstrahls nach unten (22.)

&! (r+)	Römische Ziffern (Gemeine) Anfang (12.8.2.)
&! (r-)	Römische Ziffern Ende (12.8.2.)
&! (R+)	Römische Ziffern (Versalien) Anfang (12.8.2.)
&! (R-)	Römische Ziffern Ende (12.8.2.)
&!- (a, b)	Unterstreichungsstrich (12.11.)
&!- (a, b, n)	Waagerechte Linie (12.11.)
&!- (a, b, n:ls:lw:l <sub>off</sub> )	Gestrichelte waagerechte Linie (12.11.)
&!-- (a, b)	Fußnotenlinie (12.11.)
&!-x	horizontales Spiegeln des Zeichens x (12.10.)
&! . (a, b)	Unterpunktierung (12.11.)
&! . (a, b, n)	Unterpunktierung (12.11.)
&! . . (a, b)	Punktreihe in eigener Zeile (12.11.)
&! . . (a, b, n)	Punktreihe in eigener Zeile (12.11.)
&! / (a, b, n)	Linie zwischen gemerkten Punkten (12.11.)
&! / (a, b, n:ls:lw:l <sub>off</sub> )	Gestrichelte Linie zwischen gemerkten Punkten (12.11.)
&! = (a, b, -1, c, m, y, b)	farbiges Rechteck zwischen gemerkten Punkten (12.11.)
&! = (a, b, -2, nm, p <sub>roz</sub> )	farbiges Rechteck zwischen gemerkten Punkten (12.11.)
&! = (a, b, 0, g)	graues Rechteck zwischen gemerkten Punkten (12.11.)
&! = (a, b, n)	Rechteck zwischen gemerkten Punkten (12.11.)
&! = (a, b, n:ls:lw:l <sub>off</sub> )	gestricheltes Rechteck zwischen gemerkten Punkten (12.11.)
&! \ (a, b, n:ls:lw:l <sub>off</sub> , p, l, w)	gestrichelte Linie mit Pfeilspitzen zwischen gemerkten Punkten (12.11.)
&! \ (a, b, n, ng, l, w)	Akkolade (geradlinig) zwischen gemerkten Punkten (12.11.)
&! \ (a, b, n, p, l, w)	Linie mit Pfeilspitzen zwischen gemerkten Punkten (12.11.)
&!   (a, l, n)	Senkrechte Linie (12.11.)
&! ? (a, l, n, wa, w <sub>l</sub> )	Senkrechte Wellenlinie (12.11.)
&! _ (a, l, n, wa, w <sub>l</sub> )	Waagerechte Wellenlinie (12.11.)
&! /x	Drehen des Zeichens x (12.10.)
&! =x	horizontales Spiegeln des Zeichens x (12.10.)
&! \x	Drehen des Zeichens x (12.10.)
&!  x	vertikales Spiegeln des Zeichens x (12.10.)
&! A (0)	Aufheben der Anweisung &! A (nn) (1.)
&! A (n)	Begrenzung für das Austreiben zu kurzer Spalten (1.)
&! A	Ausgleichsstelle für zu kurze Spalten (1.)
&! A+	Ausgleichsstelle für zu kurze Spalten (1.)
&! A-	Einzug für \$ erzwingen (7.1.)
&! B!	Spatium fixieren (12.7)
&! B (+)	Spatien fixieren Anfang (12.7)
&! B (-)	Spatien fixieren Ende (12.7)
&! B (n)	Positionieren auf Merkstelle (10.4.)
&! B+	Blocksatz einschalten (8.)
&! B-	Blocksatz ausschalten (8.)
&! Bn	Positionieren auf Merkstelle (10.4.)
&! C (c, m, y, k)	Umschalten auf Farbdruck (CMYK) (11.4.)
&! C (g)	Umschalten auf Graudruck (11.4.)



&!C (Hnm)	Umschalten auf Farbdruck (HKS) (11.4.)
&!C (Hnm, proz)	Umschalten auf Farbdruck (HKS) (11.4.)
&!C{	Rückschalten auf Schwarz-Weiß-Druck (11.4.)
&!D!!	vertikale Position fixieren (7.4.)
&!D- (nn)	Verringerter Durchschuss nach der Zeile (7.4.)
&!D-n	Verringerter Durchschuss nach der Zeile (7.4.)
&!D/n	Verringerter Durchschuss nach der Zeile (7.4.)
&!D/ (nn)	Verringerter Durchschuss nach der Zeile (7.4.)
&!D= (nn)	Zusätzlicher Durchschuss vor der Zeile (7.4.)
&!D= (nn!)	Zusätzlicher Durchschuss vor der Zeile (7.4.)
&!D=n	Zusätzlicher Durchschuss vor der Zeile (7.4.)
&!D?0	variablen Vorschub fixieren (7.4.)
&!D?n	festen Vorschub variabel machen (7.4.)
&!D\n	verringertes Durchschuss nach der Zeile (7.4.)
&!D\ (nn)	verringertes Durchschuss nach der Zeile (7.4.)
&!D\ (nn!)	verringertes Durchschuss nach der Zeile (7.4.)
&!D00	Zeile registerhaltig nach oben verschieben (7.4.)
&!D88	Zeile registerhaltig verschieben (7.4.)
&!D99	Zeile registerhaltig nach unten verschieben (7.3.)
&!Dnn	zusätzlicher Durchschuss nach der Zeile (7.4.)
&!E	erreichte Position ausdrucken (10.4.)
&!F (nnn)	fester Ausschluss (8.)
&!F-	Fußnotenlinie für aktuelle Spalte ausschalten (6.)
&!FN	Fußnoten mit Original-Nummern ausgeben (6.)
&!FS	Fußnoten seitenweise nummerieren (6.)
&!FU	Fußnotennummer unterdrücken (6.)
&!G+	kleinerer Schriftgrad, Hochstellung Anfang (12.3.)
&!G-	kleinerer Schriftgrad, Tiefstellung Anfang (12.5.)
&!G.	kleinerer Schriftgrad Ende, Schriftgrundlinie (12.3.)
&!G=	kleinerer Schriftgrad in der Zeile Anfang (12.5.)
&!H+	kleinerer Schriftgrad, Hochstellung Anfang (12.3.)
&!H-	kleinerer Schriftgrad, Tiefstellung Anfang (12.5.)
&!H.	kleinerer Schriftgrad Ende, Schriftgrundlinie (12.3.)
&!H=	kleinerer Schriftgrad in der Zeile Anfang (12.5.)
&!I	Belichtung wieder ein (8.)
&!J (+n)	zusätzlicher Raum für (überlaufende) Fußnoten (6.)
&!J (-n)	Raum für Fußnoten verringern (6.)
&!K (+m: n)	Größenänderung um m Punkt (11.3.)
&!K (+m: n/1)	Größenänderung um m Punkt, Zeilenabstandsänderung auf 1 Punkt (11.3.)
&!K (+m: n/+1)	Größenänderung um m Punkt, Zeilenabstandsänderung um 1 Punkt (11.3.)
&!K (+m: n/-1)	Größenänderung um m Punkt, Zeilenabstandsänderung um -1 Punkt (11.3.)
&!K (+n)	Größenänderung um n Punkt (11.3.)
&!K (+n/1)	Größenänderung um n Punkt, Zeilenabstandsänderung auf 1 Punkt (11.3.)
&!K (+n/+1)	Größenänderung um n Punkt, Zeilenabstandsänderung um 1

	Punkt (11.3.)
&!K(+n/-1)	Größenänderung um n Punkt, Zeilenabstandsänderung um 1 Punkt (11.3.)
&!K(-m:n)	Größenänderung um -m Punkt (11.3.)
&!K(-m:n/1)	Größenänderung um -m Punkt, Zeilenabstandsänderung auf 1 Punkt (11.3.)
&!K(-m:n/+1)	Größenänderung um -m Punkt, Zeilenabstandsänderung um 1 Punkt (11.3.)
&!K(-m:n/-1)	Größenänderung um -m Punkt, Zeilenabstandsänderung um -1 Punkt (11.3.)
&!K(-n)	Größenänderung um -n Punkt (11.3.)
&!K(-n/1)	Größenänderung um -n Punkt, Zeilenabstandsänderung auf 1 Punkt (11.3.)
&!K(-n/+1)	Größenänderung um -n Punkt, Zeilenabstandsänderung um 1 Punkt (11.3.)
&!K(-n/-1)	Größenänderung um -n Punkt, Zeilenabstandsänderung um -1 Punkt (11.3.)
&!K(mm:nn)	Größenänderung auf die angegebene Schriftgröße (11.3.)
&!K(mm:nn/1)	Größenänderung auf die angegebene Schriftgröße, Zeilenabstandsänderung auf 1 Punkt (11.3.)
&!K(mm:nn/+1)	Größenänderung auf die angegebene Schriftgröße, Zeilenabstandsänderung um 1 Punkt (11.3.)
&!K(mm:nn/-1)	Größenänderung auf die angegebene Schriftgröße, Zeilenabstandsänderung um -1 Punkt (11.3.)
&!K(nn)	Größenänderung auf nn Punkt (11.3.)
&!K(nn/1)	Größenänderung auf die angegebene Schriftgröße, Zeilenabstandsänderung auf 1 Punkt (11.3.)
&!K(nn/+1)	Größenänderung auf die angegebene Schriftgröße, Zeilenabstandsänderung um 1 Punkt (11.3.)
&!K(nn/-1)	Größenänderung auf die angegebene Schriftgröße, Zeilenabstandsänderung um -1 Punkt (11.3.)
&!K(0)	Rückschalten auf Original-Schriftgröße (11.3.)
&!K+n	veraltet für &!K(+n)
&!K-n	veraltet für &!K(-n)
&!K{	Rückschalten auf Original-Schriftgröße (11.3.)
&!K0	veraltet für &!K(0)
&!K00	veraltet für &!K(00) und &!K(0)
&!Kmm:nn	veraltet für &!K(mm:nn)
&!Knn	veraltet für &!K(nn)
&!L+	Linie unter Kolumnentitel wieder einschalten (2.)
&!L-	Linie unter Kolumnentitel ausschalten (2.)
&!L-(n)	Linie unter Kolumnentitel ausschalten (2.)
&!L.	Nächste Linie unter Kolumnentitel unterdrücken (2.)
&!L.(n)	Nächste n Linien unter Kolumnentitel unterdrücken (2.)
&!M(!n)	horizontale Merkstelle definieren und ausdrucken (10.4.)
&!M(n)	horizontale Merkstelle definieren (10.4.)
&!M(n1-n2)	Merkstellen definieren (10.4.)
&!M(Vn)	vertikale Merkstelle definieren (10.4.)

&!M (Wn)	vertikale Merkstelle definieren (10.4.)
&!M (Vn1–n2)	vertikale Merkstellen definieren (10.4.)
&!M–	alle horizontale Merkstellen auf 0 setzen (10.4.)
&!Mn	horizontale Merkstelle definieren (10.4.)
&!MO	horizontale Merkstelle definieren (10.4.)
&!MZ	horizontale Merkstelle definieren (10.4.)
&!N	normaler Zeilenumbruch (8.)
&!N+	Seitennummer wieder einschalten (2.)
&!N (n)	ausgebende Seitennummer auf n setzen (1.)
&!N (_)	Ausgabe der Seitennummer unterdrücken (1.)
&!N (A)	Seitennummer in arabischen Ziffern ausgeben (1.)
&!N (a)	Seitennummer in arabischen Ziffern ausgeben (1.)
&!N (R)	Seitennummer in römischen Ziffern (Großbuchstaben) ausgeben (1.)
&!N (r)	Seitennummer in römischen Ziffern (Kleinbuchstaben) ausgeben (1.)
&!N–	Seitennummer ausschalten (2.)
&!N– (n)	Seitennummer ausschalten (2.)
&!N.	Nächste Seitennummer unterdrücken (2.)
&!N. (n)	Nächste n Seitennummern unterdrücken (2.)
&!O	Belichtung aus (8.)
&!P (+–0)	Satzspiegel-Verschiebung aufheben (10.4.)
&!P (+n)	Satzspiegel nach rechts verschieben (10.4.)
&!P (–n)	Satzspiegel nach links verschieben (10.4.)
&!P (n)	Positionieren auf Merkstelle (10.4.)
&!P (Hn)	Positionieren auf Mitte zwischen vertikale Merkstelle n und aktueller Position (10.4.)
&!P (Vn)	Positionieren auf vertikale Merkstelle (10.4.)
&!P (Wn)	Positionieren auf vertikale Merkstelle (10.4.)
&!Pn	Positionieren auf Merkstelle (10.4.)
&!Q+	Kolumnentitel wieder einschalten (2.)
&!Q–	Kolumnentitel ausschalten (2.)
&!Q– (n)	Kolumnentitel ausschalten (2.)
&!Q.	Nächsten Kolumnentitel unterdrücken (2.)
&!Q. (n)	Nächste n Kolumnentitel unterdrücken (2.)
&!R	Blocksatz (»Randausgleich«) erzwingen (10.2.)
&!R (–1)	Zeilennummerierung seitenweise (10.5.)
&!R (: 3)	Zeilennummernfeld ab hier dreistellig (10.5.)
&!R (: 2)	Zeilennummernfeld wieder zweistellig (10.5.)
&!R (nn)	Zeilennummerierung fortlaufend (10.5.)
&!R (U)	Zeilennummerierung unterbrechen (10.5.)
&!R (W)	Zeilennummerierung wieder aufnehmen (10.5.)
&!R+	Einschalten der Zeilennummerierung (10.5.)
&!R–	Ausschalten der Zeilennummerierung (10.5.)
&!R.	Ausschalten der Zeilennummerierung (10.5.)
&!S (+0)	Veränderung der Spatienbreite aufheben (8.)
&!S (+n)	Spatienbreite vergrößern (8.)
&!S (–n)	Spatienbreite verkleinern (8.)

&!S (1)	Weiterschalten der Spaltennummern um 1 (1)
&!S (1, m/n)	Verändern der Satzbreite (4.3.)
&!S (1, m/n; n)	wie &!S (1, m/n), für max. n Zeilen (4.3.)
&!S (1, mmm)	Verändern der Satzbreite (4.3.)
&!S (1, mmm; n)	wie &!S (1, mmm), für max. n Zeilen (4.3.)
&!S (1, Pnn)	Verändern der Satzbreite (4.3.)
&!S (1, Pnn; n)	wie &!S (1, Pnn), für max. n Zeilen (4.3.)
&!S (1, Pnn+m)	Verändern der Satzbreite (4.3.)
&!S (1, Pnn+m; n)	wie &!S (1, Pnn+m), für max. n Zeilen (4.3.)
&!S (1, Pnn-m)	Verändern der Satzbreite (4.3.)
&!S (1, Pnn-m; n)	wie &!S (1, Pnn-m), für max. n Zeilen (4.3.)
&!S (2)	Weiterschalten der Spaltennummern um 2 (1.)
&!S (n, m/n)	Beginn einer n-spaltigen Einschaltung (4.2.)
&!S (n, mmm)	Beginn einer n-spaltigen Einschaltung (4.2.)
&!S+	Silbentrennung ein (9.)
&!S-	Silbentrennung aus (9.)
&!S:n	n = Zahl der in aufeinander folgenden Zeilen erlaubten Trennungen (9.)
&!S::	n = Rückstellen der Zahl der in aufeinander folgenden Zeilen erlaubten Trennungen (9.)
&!S{	Rückkehr zu einspaltigem normalbreitem Satz (4.2.)
&!SN	Umschalten auf Silbentrennregeln nach der Rechtschreibreform (9.)
&!SN(itrenw, itrnv, itrnh)	Umschalten auf Silbentrennregeln nach der Rechtschreibreform; Trennparameter ändern (9.)
&!SV	Umschalten auf Silbentrennregeln vor der Rechtschreibreform (9.)
&!SV(itrenw, itrnv, itrnh)	Umschalten auf Silbentrennregeln vor der Rechtschreibreform; Trennparameter ändern (9.)
&!T	Trennung aus (9.)
&!T()	Mindestzahl von Textzeilen zurücksetzen (6.)
&!T(n)	Mindestzahl von Textzeilen neu setzen (6.)
&!T.	Spaltenkopftext löschen (2.2.)
&!T:	Spaltenkopftext merken und ausgeben (2.2.)
&!T/...&!T{	Spaltenkopftext für rechte Seiten (2.2.)
&!T=...&!T{	Spaltenkopftext für nachfolgende Seiten (2.2.)
&!T\...&!T{	Spaltenkopftext für nachfolgende Spalten (2.2.)
&!Tn+	Anfang eines Textfeldes (10.4.)
&!Tn-	Ende eines Textfeldes (10.4.)
&!Tn.	erneute Ausgabe eines Textfeldes (10.4.)
&!U	unveränderte Zeileneinteilung (8.)
&!V	volle Zeile (8.)
&!W	Warten mit Spaltenwechsel (1.)
&!W&!W	Warten mit Spaltenwechsel (1.)
&!X	Aufheben von &!V (8.)
&!Y	Aufheben von &!W und &!W&!W (1.)
&!Y(n)	bedingter Spaltenwechsel (1.)
&!Y(n.)	bedingter Spaltenwechsel (1.)

&!Z (nnn)	bedingter Zeilenwechsel (8.)
&!Z (m/n)	bedingter Zeilenwechsel (8.)
&!Z- (nnn)	bedingter Zeilenwechsel (8.)
&!Z- (m/n)	bedingter Zeilenwechsel (8.)
&#xXXXX;	character entities mit dem Hex-Code XXXX (12.8.5.)
&&	Titelzeile Stufe 2 (3.)
&&&&	Logisches Dateieinde (1.)
&&&*n&&&{	Spaltenwechsel, neue Spaltennummer um n höher (1.)
&&&+n&&&{	Seitenwechsel, neue Seitennummer um n höher (1.)
&&&-n&&&{	Seitenwechsel auf die Seitennummer n (1.)
&&&-0&&&{	(am Dateieinde): ggf. leere linke Seite ausgeben (1.)
&&&=n&&&{	Spaltenwechsel auf die Spaltennummer n (1.)
&&&{	Titelzeile Stufe 3 Ende (3.)
&&&L&&&{	Neue linke Seite (1.)
&&&N&&&{	Neue rechte oder linke Seite (1.)
&&&R&&&{	Neue rechte Seite (1.)
&&&S&&&{	Neue Spalte (1.)
&&*n&&{	Spaltenwechsel, neue Spaltennummer um n höher
&&+n&&{	Seitenwechsel, neue Seitennummer um n höher
&&-n&&{	Seitenwechsel auf die Seitennummer n (1.)
&&=n&&{	Spaltenwechsel auf die Spaltennummer n (1.)
&&{	Titelzeile Stufe 2 Ende (3.)
&&L&&{	Neue linke Seite (1.)
&&N&&{	Neue rechte oder linke Seite (1.)
&&R&&{	Neue rechte Seite (1.)
&&S&&{	Neue Spalte (1.)
&.ab	Makroaufruf (19.)
&:	Bibliographie-Eintrag (hängender Einzug) (10.1.2.)
&= (nnn)	Rest der Zeile und Folgezeilen einrücken (10.1.2.)
&= (nnn; z)	Aussparung für Initiale (10.1.2.)
&= (nnn; Vn)	Aussparung für Initiale (10.1.2.)
&= (0; 1)	Aussparung für Initiale aufheben (10.1.2.)
&= (m/n)	Rest der Zeile und Folgezeilen einrücken (10.1.2.)
&= (m/n; z)	Aussparung für Initiale (10.1.2.)
&= (Pm+)	Rest der Zeile und Folgezeilen einrücken (10.1.2.)
&= (Pm+nnn)	Rest der Zeile und Folgezeilen einrücken (10.1.2.)
&= (Pm+nnn; z)	Rest der Zeile und Folgezeilen einrücken (10.1.2.)
&= (Pm-nnn)	Rest der Zeile und Folgezeilen einrücken (10.1.2.)
&= (Pm-nnn; z)	Rest der Zeile und Folgezeilen einrücken (10.1.2.)
&=- (nnn)	Folgezeilen einrücken (10.1.2.)
&=- (m/n)	Folgezeilen einrücken (10.1.2.)
&=- (Pm+nnn)	Folgezeilen einrücken (10.1.2.)
&=- (Pm-nnn)	Folgezeilen einrücken (10.1.2.)
&=- (R)	Folgezeilen nach rechts ausschließen (10.2.)
&=-nnn	Folgezeilen einrücken (10.1.2.)
&=nnn	Rest der Zeile und Folgezeilen einrücken (10.1.2.)
&=999	Folgezeilen auf aktuelle Position einrücken (10.1.2.)
&=\	Nächste mit &= beginnende Anweisung ignorieren (10.1.2.)

&{	Titelzeile Stufe 1 Ende (3.)
&n&	Titelzeile Stufe 1.n (1.1 bis 1.9 und 1.a bis 1.w) (3.)
&n&{	Titelzeile Stufe 1.n (1.1 bis 1.9 und 1.a bis 1.w) Ende (3.)
&X	Kommentar Anfang (20.)
&X{	Kommentar Ende (20.)
&XXn	Anfang eines Eintrags zum Apparat Nr. n (5.)
&XX{	Ende eines Apparateintrags (5.)
'	' Apostroph (12.7.)
(	( runde Klammer auf (12.7.)
)	) runde Klammer zu (12.7.)
*	* Sternchen, hochgestellt (12.7.)
+	+ Pluszeichen (12.7.)
++	dritte Schrift Anfang (11.1.2.)
++{	dritte Schrift Ende, Rückschalten auf gemerkte Schrift (11.1.2.)
,	, Komma (12.7.)
-	- Divis, Gedankenstrich, »bis« (12.7.)
--	-- 2 Halbgeviertstriche (12.7.); (arabisch:) Kashida (16.)
-\	- Divis (12.7.)
.	. Punkt (12.7.)
/	/ Schrägstrich (12.7.)
:	: Doppelpunkt (12.7.)
;	; Strichpunkt (12.7.)
<	< spitze Klammer auf (12.7.)
<<	achte Schrift Anfang (11.1.2.)
<<{	achte Schrift Ende, Rückschalten auf gemerkte Schrift (11.1.2.)
<!-- ... -->	Kommentar (19.)
<?tus xxx?>	PI (XML: Processing Instruction) (19.)
<name>	Makroaufruf (19.)
=	= Gleichheitszeichen (12.7.)
>	> spitze Klammer zu (12.7.)
>>	siebte Schrift Anfang (11.1.2.)
>>{	siebte Schrift Ende, Rückschalten auf gemerkte Schrift (11.1.2.)
?	? Fragezeichen (12.7.)
@" ... @{	Kolumnentitel unten für linke Seiten (2.)
@' ... @{	Kolumnentitel unten für rechte Seiten (2.)
@ (m/n)	Ende einer Tabellenspalte (10.3.)
@+	davor stehendes und/oder nachfolgendes Spatium unterdrücken (8.)
@--0	Zeilenteilung, mit Linie aufgefüllt (10.2.)
@- (nnn)	Einzug in gleicher Zeile vom linken Rand an (10.1.1.)
@- (m/n)	Einzug in gleicher Zeile vom linken Rand an (10.1.1.)
@-0	Zeilenteilung (10.2.)
@-A	austreiben (10.2.)
@-G	Zeilenteilung in Spalten mit gerader Nummer (10.2.)

@-L	Zeilenteilung auf linken Seiten (10.2.)
@-n	n*10 Punkt Einzug in gleicher Zeile (10.1.1.)
@-M (na1, ne1)	Zentrierung in einem Feld (10.2.)
@-M (na1, ne1, na2)	Zentrierung im kürzeren von zwei Feldern (10.2.)
@-N	nicht austreiben trotz Blocksatz (10.2.)
@-R	Zeilenteilung auf rechten Seiten (10.2.)
@-U	Zeilenteilung in Spalten mit ungerader Nummer (10.2.)
@-Z	Zeilenteilung, Rest der Zeile auf Mitte zentrieren (10.2.)
@.0 @. .0 @. . .0	Zeilenteilung mit Auspunktieren (10.2.)
@.0 @. .0 @. . .0	Zeilenteilung mit Auspunktieren, mindestens 2 Punkte (10.2.)
@/ . . .@{	Kolumnentitel für rechte Seiten (2.)
@:0 @::0 @:::0	Auffüllen mit Punkten (10.2.)
@= . . .@{	Kolumnentitel für linke Seiten (2.)
@@	vierte Schrift Anfang (11.1.2.)
@@{	vierte Schrift Ende, Rückschalten auf gemerkte Schrift (11.1.2.)
@A. . .@{	Marginalie außen (10.5.)
@I. . .@{	Marginalie innen (10.5.)
@L. . .@{	Marginalie links (10.5.)
@n	Ende einer Tabellenspalte (10.3.)
@R. . .@{	Marginalie rechts (10.5.)
@S. . .@{	Marginalie links im Satzspiegel (10.5.)
@T. . .@{	Marginalie rechts im Satzspiegel (10.5.)
[	[ eckige Klammer auf (12.7.)
\	am Wortanfang: Trennverbot (9.)
\	Kann-Trennstelle für Silbentrennung (9.)
\-	- Gedankenstrich, »bis« (12.7.)
\\	Trennverbot (9.)
\\\	Auszeichnungen nicht automatisch aufheben (11.1.1.)
\{	letzte Schriftumschaltung aufheben (11.1.1.)
]	] eckige Klammer zu (12.7.)
^!	¡ spanisches Ausrufezeichen (12.7.)
^"	■ Blockade (12.7.)
^#	# Nummernzeichen (12.7.)
^\$	\$ Dollarzeichen (12.7.)
^%	% Prozentzeichen (12.7.)
^&	& Et-Zeichen (12.7.)
^&^#XXXX;	& Character-Entities (wenn 3. Wert von LAU = 1) für feste Zwischenräume (12.8.5.)
^&amp;	& Et-Zeichen (wenn 3. Wert von LAU = 1) (12.7.)
^&apos;	' Apostroph (wenn 3. Wert von LAU = 1) (12.7.)
^&gt;	> größer als (wenn 3. Wert von LAU = 1) (12.7.)
^&lt;	< kleiner als (wenn 3. Wert von LAU = 1) (12.7.)
^&quot;	" Anführungszeichen (senkrecht stehend; wenn 3. Wert von LAU = 1) (12.7.)
^'	‘ umgekehrter Apostroph (12.7.)
^*	× »mal« (Kreuz) (12.7.)

^+	† Sterbekreuz (12.7.)
^-	- Minus (12.7.)
^.	· »mal« (Punkt) (12.7.)
^<	< spitze Klammer auf (12.7.)
^=	■ Blockade mit Fehlerkommentar im Protokoll (12.7.)
^>	> spitze Klammer zu (12.7.)
^?	¿ spanisches Fragezeichen (12.7.)
^@	@ Klammeraffe (12.7.)
^\	\ umgekehrter Schrägstrich (12.7.)
^^	Trennverbot (9); nach f und c (in Fraktur-Schriften auch nach l, s und t): Ligatur (12.2.)
^_	_ Unterstreichungsstrich (12.7.)
^	senkrechter Strich (12.7.)
^1	¹ hochgestellte Ziffer (12.3.)
^a	ä a Umlaut (12.2.)
^A	Ä A Umlaut (12.2.)
^o	ö o Umlaut (12.2.)
^O	Ö O Umlaut (12.2.)
^s	ß scharfes s (12.2.)
^S	ß scharfes S (12.2.)
^u	ü u Umlaut (12.2.)
^U	Ü U Umlaut (12.2.)
^x	(Weitere, mit ^ codierte Buchstaben kommen in nicht-lateinischen Schriften vor; vgl. Kapitel 13 bis 17)
_	fester Ausschluss (12.7.)
{	Auszeichnung Ende (11.1.2.)
{{	Rückschalten auf Grundschrift, Merken der zuvor benutzten Schrift (11.1.2.)
	senkrechter Strich (12.7.)
}	Schrägstellung der Schrift (11.1.2.)
}{	Schrägstellung Ende (11.1.2.)
}}	zweite Schrift Anfang (11.1.2.)
}}{	zweite Schrift Ende, Rückschalten auf gemerkte Schrift (11.1.2.)
0-9	Ziffern 0-9
a-z	Kleinbuchstaben a-z
A-Z	Großbuchstaben A-Z



# **Makros für die Satzumgebung**

##*AUMBRUCH	Umbrechen von Texten mit Apparaten . . . . .	1331
##*BOOKMARKS	Lesezeichen (bookmarks) für PDF-Dateien erzeugen . .	1339
##*FPROT	Fehlerprotokoll aus SATZ-Protokoll extrahieren . . . .	1342
##*FUNO	Fußnoten aus dem Text extrahieren . . . . .	1345
##*GRAFIK	EPS-Grafiken für *PSAUS aufbereiten . . . . .	1348
##*HIEROGR	Hieroglyphe in Grafikdatei einfügen . . . . .	1353
##*KERNPAR	Kerningtabellen in KOPIERE-Parameter unwandeln . .	1355
##*MASKE	Masken für die SATZ-Ausgabe definieren . . . . .	1356
##*MONT	SATZ-Ausgabedateien zusammenmontieren . . . . .	1358
##*PRECOMPOSED	PS-Dateien nach OpenType-Konventionen umcodieren	1359
##*PROT2QU	Quelldaten und Parameter aus Satzprotokoll extrahieren	1361
##*PS4A4	PostScript-Datei ausdrucken: 4 Seiten/A4-Blatt . . . .	1362
##*PSAUS	SATZ-Ausgabe auf PostScript-Geräten . . . . .	1364
##*PSAUSWAHL	Seiten aus PostScript-Datei auswählen . . . . .	1376
##*PSDICKTEN	Dicktenliste für PostScript-Font drucken . . . . .	1377
##*PSDR	PostScript-Datei ausdrucken . . . . .	1379
##*PSFONT	Dicktenwerte von PostScript-Fonts bereitstellen . . . .	1381
##*PSFONTVOR	Vorbereiten einer AFM-Datei für *PSFONT . . . . .	1384
##*PSGLYPHS	Zeichenliste für PostScript-Font drucken . . . . .	1385
##*PSKERNFILE	Kerning-Information bereitstellen . . . . .	1387
##*PSMONT	PostScript-Dateien zusammenmontieren . . . . .	1388
##*REKLAM	mit Klammern codierte Registereinträge umwandeln .	1390
##*SILARCH	Silbentrennungs-Archiv erstellen . . . . .	1393
##*SILKOR	Silbentrennungen korrigieren . . . . .	1395
##*SILLIST	Silbentrennungs-Liste erstellen . . . . .	1397
##*SILMARKE	Silbentrennungs-Markierungen einfügen . . . . .	1400
##*SILMARKO	Silbentrennungs-Markierungen optimieren . . . . .	1403
##*SUMBRUCH	Umbrechen von mehrspaltigen Einschüben . . . . .	1406
##*TAGS	SGML/XML-Tags in Makros für SATZ umwandeln . . . .	1408
##*TAGUEB	Makro-Auflösungen in *TAGS-Ergebnis übertragen . . .	1412
##*TEXGRAF	TeX-Seiten (PostScript) in Grafikdatei einfügen . . . .	1413
##*XMLATTSOR	Attribute in XML-Tags sortieren . . . . .	1416
##*XMLZIEL	XML-Tags in Satz-Zieldatei wiederherstellen . . . . .	1417

# Umbrechen von Texten mit Apparaten

#\*AUMBRUCH

## Makro:

#\*AUMBRUCH

QUELLE	=	datei	Name der Datei mit dem Protokoll eines Satzprogrammlaufs.
ZIEL	=	datei	Name der Datei für den Text mit den ans Seitenende umgestellten Apparateinträgen.
MODUS	=	-STD-	* Normalfall: Fußnoten bleiben vor den Apparaten stehen.
	=	-STD-' 2	wie -STD-, aber für zweispaltig gesetzte Fußnoten
	=	U	Fußnoten hinter die Apparateinträge umstellen.
	=	U' 2	wie U, aber für zweispaltig gesetzte Fußnoten
LOESCHEN	=	-	* Daten in der PROTOKOLL-Datei nicht löschen. Die ZIEL-Datei muss in diesem Fall leer sein.
	=	+	Daten in der ZIEL- und in der PROTOKOLL-Datei zuerst löschen.
PARAMETER	=	-	* keine Parameterangaben.
	=	datei	Name der Datei mit den Parametern.
	=	*	Die Parameter folgen auf den Makroaufruf und sind mit *EOF abgeschlossen.
PROTOKOLL	=	+	* Fehlerprotokoll ins Ablaufprotokoll ausgeben.
	=	-STD-	Fehlerprotokoll in die Standard-Protokoll-Datei ausgeben.
	=	datei	Fehlerprotokoll in die angegebene Datei ausgeben.

## Leistung

Mit diesem Makro wird der Umbruch von Texten vorbereitet, die kritische Apparate enthalten. Das Makro muss zu diesem Zweck nach einem Lauf des Satzprogramms (mit MODUS=T), bei dem der Umbruch des Textteils festgelegt wird, aufgerufen werden. Die ZIEL-Datei von #\*AUMBRUCH wird zur QUELL-Datei eines nachfolgenden Aufrufs des Satzprogramms (mit MODUS=A).

## Erläuterungen

### Satz von Texten mit kritischen Apparaten

Zusätzlich zu den normalen Fußnoten können im Satzprogramm bis zu 9 weitere Fußnoten-Apparate verwaltet werden, wie sie zum Satz kritischer Text-Editionen benötigt werden.

Die entsprechenden Apparateinträge können beim Satz zu fortlaufenden Textblöcken zusammengefasst (besonders bei kurzen Apparateinträgen aus Platzgründen zu empfehlen) oder, ähnlich wie die gewohnten Fußnoten, jeweils mit einer neuen Zeile beginnend, gesetzt werden.

Die Zuordnung der einzelnen Apparateinträge zum Text erkennt der Leser entweder anhand von Symbolen (z. B. Index-Buchstaben oder -Zahlen), die im Text auf den Apparat hinweisen und die vor den entsprechenden Apparateinträgen wiederholt sind, oder anhand von Verweisen auf Textzeilen vor jedem Apparateintrag. Die für diese Zeilenverweise benutzten Zahlen stehen entweder schon von vorneherein fest (z. B. bei Gedichten mit einer festen, als Referenz benutzten Versnummerierung), oder sie ergeben sich erst beim Umbruch, wenn die einzelnen Textzeilen jeder Seite oder der einzelnen Kapitel durchgezählt und durch eine Zeilennummer am Rand markiert werden; die Apparateinträge verweisen dann auf diese laufende Zeilennummer (häufigster Fall bei Prosa-Texten). Nur im letzteren Fall muss die Verweisnummer der Apparateinträge vom Programm selbst generiert werden.

Die einzelnen Apparateinträge sollten in der Regel relativ kurz sein (einzelne Wörter bis wenige Zeilen). Im Unterschied zu Fußnoten ist insbesondere nicht vorgesehen, dass lange Apparateinträge bei Bedarf automatisch auf Folgeseiten fortgesetzt werden. Überlange Apparateinträge zur letzten Zeile einer Seite können daher zu überlangen Seiten führen. Das Satzprogramm meldet in diesen Fällen, um wieviel eine Seite zu hoch geworden ist, und lässt auf der Folgeseite ebensoviel Platz frei. So können Apparateteile, die zu überlangen Seiten geführt haben, vor einem weiteren Satzprogrammmlauf mit anderen Mitteln (z. B. mit Hilfe des Editors) entsprechend umgestellt werden.

### Codierung der Apparateinträge

Jeder Apparateintrag beginnt mit

`&xxn`

wobei `n` die Nummer des Apparates ist, zu dem der Eintrag gehört (es sind max. 9 verschiedene Apparate möglich), und endet mit

`&xx{`

Vor `&xxn` und nach `&xx{` steht je ein Leerzeichen; nach `&xxn` und vor `&xx{` steht kein Leerzeichen.

Ein Apparateintrag steht im Text hinter dem Wort, auf das er sich bezieht. Soll im Apparateintrag später automatisch die Nummer der Textzeile eingefügt werden, aus der der Apparateintrag stammt, so ist an der Stelle, an der im Apparateintrag diese

Nummer eingefügt werden soll (normalerweise am Anfang des Eintrags, also unmittelbar hinter dem Code »&xxn« für Apparatanfang), eine doppelte Null (»00«) zu schreiben. Beispiel:

```
Dies ist eine Stelle &xx100_Stelle] Variante }a{&xx{ aus
einem Text, der ediert werden soll.
```

Die jeweils eingesetzte Nummer ist die laufende Nummer der den Apparateintrag enthaltenden Zeile auf der betreffenden Seite, es sei denn, hinter einer solchen Zeile steht in der zu QUELLE angegebenen Protokoll-Datei von #satz ein Kommentar mit einer anderen Zeilennummer, die – z. B. bei einer kapitelweise vorzunehmenden Zählung der Zeilen – mit der Steueranweisung &!r(nn) erzeugt wird. Diese Kommentare werden von #satz nur dann in die Protokolldatei ausgegeben, wenn dabei auch der Parameter ZLN benutzt und dort als 7. Wert (IZPRN) 2 oder 3 angegeben wird.

Steht ein Apparateintrag hinter einem Wort, das vom Satzprogramm am Zeilenende getrennt wird, so setzt das Makro \*AUMBRUCH die Nummern der beiden Zeilen, in denen die Teile des Wortes stehen, durch einen »bis«-Strich (z. B.: »9-10«) getrennt als Zeilenverweis in den Apparat ein.

Bezieht sich ein Apparateintrag auf mehr als ein Wort im Text, so genügt eine einfache Zeilenangabe im allgemeinen nicht zur Identifikation der Stelle im Text; es ist vielmehr erforderlich anzugeben, über welche Zeilen sich der Apparateintrag erstreckt (z. B.: »9-11«). Zu diesem Zweck wird der Apparateintrag ebenfalls hinter dem ersten Wort, auf das er sich bezieht, in den Text eingefügt. Anstelle von »00« wird jedoch »0-0« als Steueranweisung für die Ergänzung der Zeilennummern geschrieben. Hinter dem letzten Bezugswort für diesen Apparateintrag wird ein leerer Apparateintrag (»&xxn&xx{«) mit der gleichen Apparatnummer als Bezugs-Ende-Code eingefügt. Beispiel:

```
Dies ist ein Abschnitt &xx10-0_Abschnitt ... edierenden]
zu edierender #/+b#/-&xx{ aus einem zu edierenden &xx1&xx{
Text mit Apparateinträgen.
```

Falls Apparateintrag und Bezugs-Ende-Code beim Umbruch in der gleichen Zeile zu stehen kommen, interpretiert das Programm »0-0« wie »00«; es setzt also z. B. statt »17-17« automatisch »17« ein.

Apparateinträge, die »0-0« enthalten, und Bezugs-Ende-Codes zu ein und demselben Apparat dürfen sich weder überschneiden noch ineinander geschachtelt sein, jedoch dürfen sowohl einfache Apparateinträge (also mit »00« als Ort für die Zeilennummer) zum gleichen Apparat als auch Apparateinträge mit »0-0« und zugehörige Bezugs-Ende-Codes zu anderen Apparaten zwischen einem Apparateintrag und dem zugehörigen Bezugs-Ende-Code stehen.

Der Bezugs-Ende-Code muss beim Satz spätestens auf der nächsten Seite nach der Steueranweisung für den betreffenden Apparat stehen. Weiter hinten stehende Bezugs-Ende-Codes kann das Makro nicht automatisch verarbeiten.

Eine ggf. erforderliche Schachtelung von Apparateinträgen, die »0-0« enthalten,

kann in beschränktem Umfang (d. h. max. bis zur Tiefe von 9 ineinander geschachtelten Apparaten) erreicht werden, indem zur Codierung nicht benötigte Apparatennummern herangezogen werden, die über Parameter anschließend gleichgesetzt werden.

Angaben zur typographischen Gestalt der Apparate, insbesondere Zeilenbreite und Zeileneinteilung (fortlaufender Apparat oder Zeilenwechsel bei jedem Apparateintrag) sowie Abstände und evtl. Linien zwischen den einzelnen Apparaten, werden über Parameter dem Satzprogramm direkt mitgeteilt.

Bei manchen Editionen muss in fortlaufend zu setzenden Apparaten an bestimmten Stellen (z. B. beim ersten Apparateintrag in einem auf dieser Seite beginnenden neuen Kapitel) Zeilenwechsel vorgenommen werden. Zu diesem Zweck können Stellen in Text durch `&xxn\&xx{` markiert werden, um anzuzeigen, dass der erste Apparateintrag danach mit einer neuen Zeile beginnen soll.

## Ablauf des Satzvorganges

Der Satz von Texten, die Apparate enthalten, erfordert drei Arbeitsschritte.

In einem ersten Satzprogrammmlauf (mit `MODUS=T`; Angabe zum Parameter `LAU`: 1 oder 2) wird der endgültige Zeilenumbruch für die Textzeilen festgelegt und anhand der Angaben mit den Parametern `AP1 . . . AP9` der Platzbedarf für die Apparate (einschließlich der in den Fußnoten enthaltenen Apparateinträge) berechnet. Die Apparateinträge selbst bleiben dabei noch an den Stelle im Text stehen, an denen sie angegeben wurden.

Dieser Schritt wird ggf. so lange (nach entsprechender Korrektur) wiederholt, bis keine Fehlerkommentare zu den Apparateinträgen mehr auftreten.

Anschließend werden mit dem Makro `*AUMBRUCH` die Apparateinträge aus dem Text (und ggf. den Fußnoten) herausgezogen, wobei ggf. die Zeilenverweise eingesetzt werden, und mit den für den Satz benötigten Steueranweisungen ans Seitenende (ggf. unter die normalen Fußnoten) gestellt. Eingabe für dieses Makro ist die `PROTOKOLL`-Datei des vorhergehenden Satzprogrammmlaufs.

Das Ergebnis von `*AUMBRUCH` ist eine Datei, die als `QUELL`-Datei für einen anschließenden Satzprogrammmlauf (mit `MODUS=A`, Angabe zum Parameter `LAU`: 1 oder 3 oder 4) dient. Erst dieser Lauf stellt den endgültigen seitenumbrochenen Satz einschließlich der Apparate her.

Aus diesem Ablauf ergibt sich, dass den Umbruch verändernde Korrekturen nur möglich sind, solange in der Textfassung korrigiert wird, die als Eingabe für den Satzprogrammmlauf vor der Festlegung des Umbruchs und dem Herausziehen der Apparateinträge mit dem Makro `*AUMBRUCH` dient. Lassen sich den Umbruch verändernde Korrekturen in der Datei, in der die Apparateinträge schon aus dem Text herausgezogen und ans Seitenende gestellt sind, nicht vermeiden, so muss die Umstellung der Text- und Apparateile und die Umnummerierung der Zeilenverweise in den Apparateinträgen in die Korrektur mit einbezogen werden.

## Parameter

Die Angaben in [ ] geben die Parameterarten an; diese sind in der Beschreibung »TUSTEP-Grundlagen« im Kapitel »Parameter« definiert.

Die Parameter müssen in der gleichen Reihenfolge wie in dieser Beschreibung angegeben werden.

### Fixierung des Umbruchs

Sind keine Parameter angegeben, so werden in den Text zusätzliche Steueranweisungen eingefügt, die den Umbruch für Text und Fußnoten fixieren. Die Fußnoten werden aus der QUELL-Datei mit entsprechenden Steueranweisungen in die ZIEL-Datei mit übernommen.

Soll nur der Umbruch für den Textteil fixiert und sollen die Fußnoten nicht mit in die ZIEL-Datei übernommen werden, so kann folgender Parameter (ohne Angaben im Informationsfeld) angegeben werden:

**FIX**           Fixiert den Umbruch des Textteils, ohne evtl. vorhandene Fußnoten mit zu übernehmen

### Parameter für die Behandlung der Fußnoten

**FNA**           Zeichenfolgen, die am Anfang des Fußnotenteils eingesetzt werden sollen. [ II ]

Es werden zwei (bzw., bei zweispaltig gesetzten Fußnoten, vier) Zeichenfolgen erwartet; die erste der angegebenen Zeichenfolgen wird eingesetzt, wenn der Fußnotenteil einer Seite mit einer neuen Fußnote beginnt; die zweite Zeichenfolge wird eingesetzt, wenn der Fußnotenteil einer Seite mit einer Fortsetzungszeile einer übergelaufenen Fußnote beginnt.

Bei zweispaltig gesetzten Fußnoten wird die dritte bzw. vierte Zeichenfolge beim Übergang auf die zweite Spalte eingesetzt, und zwar die dritte, wenn die zweite Spalte mit einer neuen Fußnote beginnt, bzw. die vierte, wenn die zweite Spalte innerhalb einer Fußnote beginnt.

**FNE**           Zeichenfolgen, die am Ende des Fußnotenteils eingesetzt werden sollen. [ II ]

Es werden zwei Zeichenfolgen erwartet. Die erste der angegebenen Zeichenfolgen wird eingesetzt, wenn der Fußnotenteil mit dem Ende einer Fußnote endet; die zweite Zeichenfolge wird eingesetzt, wenn die letzte Fußnote der Seite nicht abgeschlossen ist, sondern auf die

nächste Seite überläuft, ohne dass zu Beginn der ersten überlaufenden Zeile eine Zeilenwechselanweisung steht.

**FN** Zeichenfolgen, die vor bzw. nach der Fußnotenziffer eingesetzt werden sollen. [ II ]

Vor der Fußnotenziffer in der ersten Fußnotenzeile einer Seite wird nichts eingesetzt. Die dort notwendige Zeichenfolge sollte in der ersten zum Parameter **FNA** angegebenen Zeichenfolge enthalten sein.

**FNZ** Angabe, ob die Fußnotenziffern als normale oder als hochgestellte Ziffern gesetzt werden sollen. [ I ]

Es wird genau ein Zahlenwert erwartet:

1 für »hochgestellte Ziffern«,  
0 für »normale Ziffern« (in der Größe der für die Fußnoten verwendeten Schrift, auf der Grundlinie stehend).

**FNF** Zeichenfolge, die eingesetzt werden soll, wenn auf einer Seite keine Fußnoten vorkommen (ersetzt auch die unter **FNA** und **FNE** angegebenen Codes). [ II ]

## Parameter für die Behandlung der Apparate

Die folgenden beiden Parameter müssen immer angegeben werden, wenn andere Parameter für die Behandlung der Apparate angegeben werden.

**APA** Zeichenfolge, die am Anfang des Apparateils eingesetzt werden soll. [ II ]

**APE** Zeichenfolge, die am Ende des Apparateils eingesetzt werden soll. [ II ]

Die folgenden Parameter zu den einzelnen Apparaten müssen in der Reihenfolge **AP1**, **AZ1**, **AF1**, **AP2**, **AZ2**, **AF2**, . . . , **APF** lückenlos aufsteigend angegeben werden.

**APn** Zeichenfolgen, die eingesetzt werden sollen vor dem ersten Apparateintrag des n-ten Apparates einer Seite, zwischen zwei Apparateinträgen des n-ten Apparates, nach dem letzten Apparateintrag des n-ten Apparates einer Seite und vor dem ersten Apparateintrag des n-ten Apparates, der nach der Codierung  $\&xxn \setminus \&xx\{$  steht und nicht gleichzeitig der erste Apparateintrag des n-ten Apparates auf der Seite ist. [II]

Für diese Angaben werden genau vier Zeichenfolgen erwartet.

Ist im Informationsfeld nichts angegeben, so wird geprüft, ob der Apparat mit der Nummer *n* im Text vorkommt, und ggf. eine Fehlermeldung ausgegeben.

Ist im Informationsfeld die Ziffer 0 angegeben, so werden die Apparateinträge zu diesem Apparat nicht aus dem Text herausgeholt, sondern bleiben als Kommentar im Text stehen.



Ist im Informationsfeld eine der Ziffern von 1 bis 8 angegeben (nur bei den Parametern AP2 bis AP9), so bedeutet dies, dass dieser Apparat in den Apparat mit der angegebenen (niedrigeren) Nummer integriert werden soll.

Stehen im Informationsfeld des Parameters AP<sub>n</sub> keine Zeichenfolgen, sondern nichts oder eine Zahl, so dürfen die entsprechenden Parameter AZ<sub>n</sub> und AF<sub>n</sub> für den n-ten Apparat nicht angegeben werden.

**AZn**

Angaben, die das Einsetzen von Zeilenverweisnummern bzw. deren Behandlung betreffen. [ II ]

Ist im Informationsfeld nichts angegeben, so setzt das Programm keine Zeilenverweisnummern ein (evtl. im Apparateintrag stehende Zeichenfolgen »00« bzw. »0-0« werden ignoriert, d. h. sie werden wie Text behandelt; leere Apparateinträge »&xxn&xx{« sind dann sinnlos).

Andernfalls werden genau 6 Zeichenfolgen erwartet.

Beginnt die 1. Zeichenfolge mit dem Buchstaben »h« (also z. B.: »/H/. . .«), so werden hochgestellte Ziffern für die Zeilenverweisnummern benutzt; ist die erste Zeichenfolge leer oder beginnt sie mit einem anderen Zeichen, so werden normal große, auf der Schriftgrundlinie stehende Ziffern benutzt.

Die 2. Zeichenfolge wird als Ersatz für die zweite und die weiteren von aufeinander folgenden gleichen Verweisnummern (wenn mehr als ein Apparateintrag des betreffenden Apparates sich auf die gleiche Textzeile bezieht) eingesetzt. Zum Löschen dieser Verweisnummern kann die Zeichenfolge »/ /« angegeben werden. Ist die zweite Zeichenfolge leer, so bleiben mehrere aufeinander folgende gleiche Zeilenverweisnummern unverändert stehen.

Ist die 3. Zeichenfolge nicht leer, so wird bei Apparateinträgen, in denen »00« zum Einsetzen der Verweisnummern steht, so verfahren, als stünde dort »0-0« und als stünde hinter dem nächsten Textwort ein leerer Apparateintrag (»&xxn&xx{«); dies hat zur Folge, dass in einem Apparateintrag, der hinter dem letzten Wort einer Zeile steht, die Zeichenfolge »00« durch ein Verweisnummern-Paar ersetzt wird.

Ist die 4. Zeichenfolge nicht leer, so wird bei Verweisnummern-Paaren, bei denen die zweite Verweisnummer um genau 1 höher ist als die erste Verweisnummer, der »bis«-Strich und die zweite Verweisnummer durch diese Zeichenfolge ersetzt (also z. B. bei der Angabe ».../\_f./...« das Verweisnummern-Paar »8-9« ersetzt durch »8\_f.«).

Die 5. Zeichenfolge gibt an, wie Verweisnummern-Paare behandelt werden sollen, deren zweite Nummer sich auf eine Zeile der folgenden Seite bezieht. Diese 5. Zeichenfolge muss die Zeichenfolgen »xxxxxx« und »xx« in dieser Reihenfolge enthalten, wobei für »xxxxxx« die Seitennummer und für »xx« die Zeilennummer eingesetzt wird. Evtl.

müssen Steueranweisungen für die Verwendung von Hochziffern mit angegeben werden.

Ist die 5. Zeichenfolge leer, so wird die folgende Voreinstellung benutzt:

```
/\-S._xxxxxx.xx/
```

Die 6. Zeichenfolge gibt an, wie Verweise auf getrennte Wörter behandelt werden sollen. Ist die 6. Zeichenfolge leer, so wird ein Verweisnummern-Paar erzeugt, das auf beide Zeilen verweist, in denen die Teile des getrennten Wortes stehen. Ist die 6. Zeichenfolge nicht leer, so wird als Verweis die Nummer der Zeile eingesetzt, in der das getrennte Wort beginnt.

**AFn** Zeichenfolge, die eingesetzt wird, falls der Apparat n fehlt. [ II ]

Diese Zeichenfolge wird nur eingesetzt, wenn Apparate zur aktuellen Seite vorhanden sind. Voreinstellung: Es wird nichts eingesetzt.

**APF** Zeichenfolge, die eingesetzt wird, wenn zur aktuellen Seite keine Apparate vorhanden sind. Voreinstellung: Es wird nichts eingesetzt. [ II ]

# Lesezeichen (bookmarks) für PDF-Dateien erzeugen

#\*BOOKMARKS

## Makro:

#\*BOOKMARKS

QUELLE	=	datei	Name der ZIEL-Datei eines Satzprogramm-Laufes, aus der die Daten für die Bookmarks entnommen werden sollen.
ZIEL	=	datei	Name der Datei für die Bookmarks.
LOESCHEN	=	-	* Daten in der ZIEL-Datei nicht löschen.
	=	+	Daten in der ZIEL-Datei zuerst löschen.
STUFEN	=	*	* Angaben zu den Hierarchiestufen und ihrer Codierung folgen auf den Makroaufruf und sind durch *eof abgeschlossen.
ZEILENINFO	=	-	* Keine Angaben zur Position der einzelnen Zeilen. Die erzeugten Verweise gehen auf den Anfang der entsprechenden Seiten des Dokuments.
	=	name	Name der Datei mit den Angaben zur vertikalen Position der einzelnen Zeilen jeder Seite (= Name der Datei, die als zweite Datei zu PROTOKOLL beim Aufruf des Satzprogramms angegeben war). Die Informationen aus dieser Datei werden benötigt, wenn beim Aktivieren eines Verweises die jeweilige Seite so positioniert werden soll, dass die betreffende Zeile oben im Fenster zu stehen kommt.
SEITENHOEHE	=	-	* Keine Angabe zur Seitenhöhe.
	=	nnn	Seitenhöhe (in Punkt) als Bezugspunkt für die vertikale Positionierung der Verweisziele. Sind die bei *PSAUS zu RAHMEN angegebenen Werte so gewählt, dass die Oberkante des Rahmens mit der Satzspiegeloberkante zusammenfällt, so kann der zu diesem Zweck angegebene Y-Wert für die rechte untere Ecke des Rahmens als nnn übernommen werden. Damit würde aber die Seite so weit nach oben geschoben, dass die Grundlinie der Zeile, auf die sich der Verweis bezieht, mit der Oberkante des sichtbaren Fensters zusammenfällt. Deshalb ist für nnn mindestens die Höhe einer Textzeile auf diesen Wert aufzuaddieren.
STARTSEITE	=	nnn	Nummer der ersten Seite, die mit *PSAUS ausgegeben werden soll.

	= -STD-	* absolute und relative Seitennummern in der mit *PSAUS erzeugten Datei sind identisch.
SPALTEN	= n	Die Bookmarks werden für eine Datei mit n-spaltigem Satz erzeugt.
	= 1	* Die Bookmarks werden für eine Datei mit 1-spaltigem Satz erzeugt.
FAKTOR	= 100	* Bei *PSAUS für den Text ist FAKTOR = 100 oder kein Faktor angegeben.
	= n	Bei *PSAUS ist zu Faktor (für die y-Richtung) der Wert n angegeben.

## Leistung

Mit diesem Makro können aus der ZIEL-Datei eines Satzprogramm-Laufes Lesezeichen (Bookmarks) für PDF-Dateien generiert werden, die ein hierarchisch gegliedertes Inhaltsverzeichnis enthalten. Die hierarchische Gliederung wird über Angaben zur Spezifikation STUFEN angegeben. Die einzelnen Lesezeichen sind mit den Seiten im Dokument verknüpft, aus denen die Lesezeichen entnommen wurden.

## Angaben zu STUFEN

Spalte 1:	Zahl, einstellig, die die Hierarchiestufe angibt (1 = höchste, 9 = niedrigste Stufe)
Spalte 2:	Doppelpunkt
Spalte 3ff:	je eine Such- bzw. Ausnahmezeichenfolge, mit der die Überschriften der angegebenen Hierarchiestufe beginnen; diese Zeichenfolgen müssen in der zu QUELLE angegebenen Datei jeweils am Zeilenanfang stehen. Die Suchzeichenfolgen sind durch "/" vor und hinter der Zeichenfolge einzuschließen, die Ausnahmezeichenfolgen durch "/" (jeweils ohne Gänsefüßchen). Die Ausnahmezeichenfolgen können bei einer beliebigen Stufe angegeben werden, jedoch empfiehlt es sich der Übersichtlichkeit halber, bei Ausnahmezeichenfolgen in Spalte 1 eine 0 anzugeben. Für die Such- bzw. Ausnahmezeichenfolgen gelten im übrigen die Regeln für die Such- bzw. Ausnahmezeichenfolgen von Parameterart IX.

Beispiel:

```

1: /&&&/
2: /&&/
3: /&/
0: //&://
0: //&{\0}&//
4: /&{0} #/
4: /${0} #/

```

## Vorbereitung der Daten

Soll nicht der gesamte Inhalt der Zeilen, die mit den zu STUFEN angegebenen Kennungen beginnen, in die bookmarks übernommen werden, so sollte die Datei vor dem Aufruf des Makros \*BOOKMARKS entsprechend abgeändert werden. Dabei ist darauf zu achten, dass die Zeilen- und ggf. die Seitennummern der betreffenden Zeilen erhalten bleiben.

## Weiterverarbeitung

Bookmarks müssen beim Aufruf von \*PSAUS wie Abbildungen in die PostScript-Datei eingebunden werden. Dazu muss die beim Aufruf von \*BOOKMARKS zu ZIEL angegebene Datei mit dem Makro \*GRAFIK so weiterbearbeitet werden, als wäre es eine Grafik, wobei zur Spezifikation FAKTOR der Wert "-" anzugeben ist. Zuvor muss sie in der Regel jedoch noch nachbearbeitet werden, z. B. indem die zu STUFEN angegebenen Zeichenfolgen eliminiert und ggf. Anweisungen über die gewünschte Schriftart und Farbe hinzugefügt werden.

Für die Lesezeichen kann zwischen den Schriftarten 0 (normal; Voreinstellung), 1 (kursiv), 2 (fett) und 3 (fett kursiv) gewählt werden, indem vor die Anweisung »/OUT« die Zeichenfolge »/F n« (zwischen blanks; n = Zahl zwischen 0 und 3) eingesetzt wird. Soll ein Lesezeichen in einer anderen Farbe als schwarz dargestellt werden, so kann dies durch Einfügen der Anweisung »/C [r g b]« (zwischen blanks; r, g und b sind Zahlen zwischen 0.0 und 1.0 und geben den Farbanteil von rot, grün und blau an) an der gleichen Stelle verlangt werden.

# Fehlerprotokoll aus Satzprotokoll extrahieren

#\*FPROT

## Makro:

#\*FPROT

QUELLE	=	datei	Name der Datei mit dem Protokoll eines Satzprogrammmlaufs.
LOESCHEN	=	-	* Daten in der PROTOKOLL-Datei nicht löschen.
	=	+	Daten in der PROTOKOLL-Datei zuerst löschen.
PARAMETER	=	-	* Keine Parameterangaben.
	=	datei	Name der Datei mit den Parametern.
	=	*	Die Parameter folgen auf den Makroaufruf und sind mit *EOF abgeschlossen.
PROTOKOLL	=	-STD-	* Das Fehlerprotokoll des Satzprogrammmlaufs in die Standard-Protokoll-Datei ausgeben.
	=	+	Das Fehlerprotokoll des Satzprogrammmlaufs ins Ablaufprotokoll ausgeben.
	=	datei	Das Fehlerprotokoll des Satzprogrammmlaufs in die angegebene Datei ausgeben.

## Leistung

Mit diesem Makro kann man das Protokoll eines Satzprogrammmlaufs so reduzieren, dass es nur noch die Liste der Parameter und die vom Satzprogramm ausgegebenen Fehlerkommentare mit knapper Textumgebung enthält. Dabei kann zusätzlich über Parameter angegeben werden, welche Fehlerkommentare unberücksichtigt bleiben bzw. ausschließlich berücksichtigt werden sollen.

## Parameter

Die Angaben in [ ] geben die Parameterarten an; diese sind in der Beschreibung »TUSTEP-Grundlagen« im Kapitel »Parameter« definiert.

### Auswählen der Fehlerkommentare

Sollen im Fehlerprotokoll alle vom Satzprogramm erzeugten Fehlerkommentare berücksichtigt werden, so brauchen keine Parameter angegeben zu werden.

### Fehlerkommentare zur Fußnotennummerierung

Sprünge in der Nummerierung der Fußnoten oder Wiederholung einer Fußnotennummer werden vom Satzprogramm mit einem entsprechenden Fehlerkommentar im Protokoll festgehalten. Dies ist auch der Fall, wenn die Fußnotennummerierung wieder (z. B. am Anfang eines neuen Kapitels) mit Nummer 1 begonnen wird. Es kann daher sinnvoll sein, z. B. wenn alle unbeabsichtigten Sprünge in der Fußnotennummerierung bereits korrigiert sind, diese Art von Fehlerkommentaren zu übergehen.

**FN-** (keine Angabe im Informationsfeld) Fehlerkommentare, die Sprünge in der Fußnotennummerierung betreffen, sollen übergangen werden.

### Fehlerkommentare zur Spaltenbreite

Das Austreiben der Zeilen wird vom Satzprogramm durch Verändern der Spaltenbreiten gegenüber dem dort angegebenen Soll-Wert vorgenommen. Bewegen sich die dabei entstehenden Spaltenbreiten außerhalb der vom Satzprogramm vorgegebenen oder über Parameter angegebenen Schranken, so wird ein entsprechender Fehlerkommentar erzeugt. Um die entsprechenden Fehlerkommentare auf das einem Satzauftrag angemessene Maß zu reduzieren, empfiehlt es sich, schon beim Aufruf des Satzprogramms mit Hilfe des Parameters **AUS** die Toleranzgrenzen entsprechend zu modifizieren. Mit Hilfe des folgenden Parameters kann darüber hinaus die Anzahl der im Fehlerprotokoll zu berücksichtigenden Fehlerkommentare reduziert werden.

**SP-** Grenzen für die Spaltenbreiten, die beim Erstellen des Fehlerprotokolls übergangen werden sollen [ 1 ]

**MIN** Untergrenze für Spaltenbreiten, bis zu der ein entsprechender Fehlerkommentar ins Fehlerprotokoll übernommen werden soll

**MAX** Obergrenze für Spaltenbreiten, ab der ein entsprechender Fehlerkommentar ins Fehlerprotokoll übernommen werden soll

Fehlerkommentare, die Spaltenbreiten zwischen **MIN** und **MAX** (jeweils einschließlich) Bildlinien betreffen, werden übergangen.

## Übrige Fehlerkommentare

Sollen beim Erstellen des Fehlerprotokolls weitere Fehlerkommentare übergangen werden, so können diese über die darin enthaltenen Zeichenfolgen mit dem folgenden Parameter ausgewählt werden:

**K-** Zeichenfolgen, die in Fehlerkommentaren enthalten sein müssen, damit diese beim Erstellen des Fehlerprotokolls übergangen werden [ IX ]

Sollen nur bestimmte Fehlerkommentare beim Erstellen des Fehlerprotokolls berücksichtigt werden, so können diese über die darin enthaltenen Zeichenfolgen mit dem folgenden Parameter ausgewählt werden. Dieser Parameter schließt alle anderen Parameter aus.

**K+** Zeichenfolgen, die in Fehlerkommentaren enthalten sein müssen, damit diese beim Erstellen des Fehlerprotokolls berücksichtigt werden [ IX ]



# Fußnoten aus Text extrahieren

#\*FUNO

## Makro:

#\*FUNO

QUELLE	=	datei	Name der Datei, die den Text mit den zu extrahierenden Fußnoten enthält.
ZIEL	=	datei	Name der Datei für den Text ohne die Fußnoten, aber mit Fußnotenverweisnummern.
MODUS	=	-	(wird z. Zt. nicht ausgewertet)
LOESCHEN	=	-	* Daten in der ZIEL- und in der FUSSNOTEN-Datei nicht löschen.
	=	+	Daten in der ZIEL- und in der FUSSNOTEN-Datei zuerst löschen.
ABSCHNITTE	=	*	Parameter, mit denen die im Text verwendeten Kennzeichnungen der Abschnittsanfänge angegeben sind, folgen auf den Makroaufruf und sind mit *EOF abgeschlossen.
	=	-STD-	* Es wird folgende Parameterangabe benutzt: AA            /\$/&/&./&!/&x/\$\$\$//
FUSSNOTEN	=	datei	Name der Datei für die aus dem Text extrahierten Fußnoten.
ANFANG	=	-	* Die erste Fußnote erhält die Nummer 1.
	=	n	Die erste Fußnote erhält die Nummer n.
KENNUNGEN	=	-STD-	* Normalfall: Die Fußnoten sind mit @F+ und @F- bzw. %0%0 und %0%0{ markiert.
	=	*	Nach dem Makroaufruf (ggf. nach Angabe der Parameter für die Kennzeichnung der Abschnitte und dem zugehörigen *EOF) werden, durch *EOF abgeschlossen, in je einer Zeile die Zeichenfolgen angegeben, mit denen im Text der Anfang einer Fußnote, das Ende einer Fußnote sowie gegebenenfalls die Rückschaltung der Fußnotennummer auf 1 oder auf 0 markiert sind. Die hier angegebenen Kennungen werden in die Codes umgesetzt, die vom Satzprogramm für den Satz der Fußnoten und deren Einbindung in den Text benötigt werden. Handelt es sich bei den zu QUELLE angegebene Daten um XML-Daten und bei den Ken-

nungen um XML-Tags (z. B. <fussnote>...</fussnote>), so ist es häufig sinnvoll, diese Tags in den Fußnoten oder (wenn z. B. bei jedem als <kapitel> getaggten Abschnitt wieder mit 1 beginnend gezählt werden sollen) im Text stehen zu lassen. Ein »+« oder «+« unmittelbar hinter einer Kennung verhindert, dass sie nach der Umsetzung aus den Daten entfernt wird.

TRENnzeichen = -STD- \* Normalfall: Folgen zwei oder mehr Fußnoten unmittelbar aufeinander, so sollen die Fußnotenverweise durch hochgestelltes Komma getrennt werden.  
 = - Folgen zwei oder mehr Fußnoten unmittelbar aufeinander, so sollen die Fußnotenverweise nicht sichtbar getrennt werden.

## Leistung

Mit diesem Makro können die im Text enthaltenen Fußnoten aus dem Text extrahiert, durchnummeriert und nach den Konventionen des Satzprogramms in die FUSSNOTEN-Datei ausgegeben werden. Der Text wird ohne Fußnoten in die ZIEL-Datei ausgegeben; dabei werden an den Stellen, an denen die Fußnoten standen, Fußnotenverweise eingesetzt.

## Zum Satz von Texten mit Fußnoten

Das Satzprogramm erwartet, wie unten beschrieben, Text und Fußnoten in getrennten Dateien. Bei der Texterfassung ist es jedoch oft bequemer, die Fußnoten im Text an der Stelle mit zu erfassen, zu der sie gehören. Dabei müssen der Anfang und das Ende der Fußnoten durch eindeutige Zeichenfolgen gekennzeichnet werden.

Mit dem Makro \*FUNO können Fußnoten, die im Text zwischen den Zeichenfolgen »@F+« und »@F-« mitgeschrieben wurden, aus dem Text extrahiert, durchnummeriert und nach den Konventionen des Satzprogramms in eine eigene Datei ausgegeben werden. Im Text werden an den Stellen, an denen die Fußnoten extrahiert wurden, Fußnotenverweise eingesetzt.

Soll (z. B. nach Beginn eines neuen Kapitels) die Fußnotenzählung wieder mit 1 beginnen, so kann dies an den entsprechenden Stellen im Text durch die Zeichenfolge »@F« angegeben werden. Die nächste danach zwischen »@F+« und »@F-« stehende Fußnote erhält dann die Nummer 1. Soll die Fußnotenzählung mit 0 statt mit 1 neu beginnen, so kann dies durch die Zeichenfolge »@F@F« angegeben werden.

Zur Kennzeichnung der Fußnoten können statt der Zeichenfolgen »@F+« und »@F-« auch die Zeichenfolgen »%0%0« (für den Fußnotenanzfang) und »%0%0{« (für das Fußnotenende) benutzt werden. Die Rückschaltung der Fußnotennummer auf 1 geschieht dann dadurch, dass der Anfang der Fußnote, die wieder die Nummer 1 erhalten soll, mit »%0%0%0« statt »%0%0« gekennzeichnet wird. Die Rückschaltung der Fußnotennummer auf 0 statt auf 1 kann mit der Anweisung »%0%0%0%0« angegeben werden.

Mit dem TUSTEP-Programm SATZ werden Texte, die Fußnoten enthalten, in zwei Schritten bearbeitet. Zunächst werden die Fußnoten gesetzt und dann beim Satz des Textes auf den entsprechenden Seiten eingesteuert.

Jede Fußnotennummer und jeder Fußnotenverweis muss beim Setzen aus 1–4 Ziffern, von denen jede durch vorangestelltes »%« gekennzeichnet ist, und evtl. zusätzlich einem ebenfalls durch »%« gekennzeichneten Kleinbuchstaben (z. B.: %1%2%3%a) bestehen.

In der Datei, die die Fußnoten enthält, muss jede neue Fußnote in einer neuen Zeile beginnen. Am Anfang dieser Zeile muss die (mit »%« codierte) Fußnotennummer stehen.

Der Aufruf der gesetzten Fußnoten geschieht beim Satz des Textes über die im Text enthaltenen Fußnotenverweise.

### Parameter ABSCHNITTE

Das dem Makro \*FUNO zugrunde liegende Programm geht davon aus, dass die Fußnoten mehrere Zeilen umfassen können. Außerdem wird versucht, das Fehlen von Kennzeichnungen für den Anfang bzw. das Ende von Fußnoten festzustellen und den Umfang der durch solche Fehler verursachten falschen Aufteilungen möglichst zu begrenzen. Zu diesem Zweck erwartet das Programm, dass ihm die wichtigsten Steueranweisungen, mit deren Hilfe der Text in Abschnitte eingeteilt wurde, mitgeteilt werden, damit es abschnittsweise nach Anfangs- und Ende-Codierungen für die Fußnoten suchen kann. Es dürfen nur Steueranweisungen angegeben werden, die nicht auch in den Fußnoten zur Kennzeichnung von Abschnitten vorkommen. Die Angabe geschieht über folgende Parameter:

- AA** Zeichenfolgen, die am Satzanfang den Anfang neuer Abschnitte (einschl. Zwischenüberschriften) eindeutig kennzeichnen [ IXa ]
- AE** Zeichenfolgen, die am Ende eines Eingabesatzes das Ende von Abschnitten (einschl. Zwischenüberschriften) eindeutig kennzeichnen [ IXb ]

Die Angaben in [ ] geben die Parameterarten an; diese sind in der Beschreibung »TUSTEP-Grundlagen« im Kapitel »Parameter« definiert.

## EPS-Grafiken für \*PSAUS aufbereiten

#\*GRAFIK

### Makro:

#\*GRAFIK

QUELLE	= datei	Name der Datei, die eine EPS-Grafik bzw. pdfmark-Anweisungen enthält. Angabe mehrerer Dateinamen, durch Apostroph getrennt, möglich.
	= -	Soll nur ein Inhaltsverzeichnis der zu ZIEL angegebenen Datei erstellt werden, so braucht keine Datei zu QUELLE angegeben zu werden.
ZIEL	= datei	Name der Datei, in der die aufbereiteten Grafiken bzw. pdfmarks zum Einbinden in die Ausgabe mit dem Makro *PSAUS gesammelt werden sollen. Es können, durch Apostroph getrennt, zwei Dateinamen angegeben werden. In diesem Fall werden die von TUSTEP hinzugefügten Zusatzinformationen für die einzelnen Grafiken in die erste, die eigentlichen Grafikdaten in die zweite ZIEL-Datei ausgegeben.
LOESCHEN	= -STD-	In der ZIEL-Datei Grafiken, die die gleiche Nummer haben wie eine der neuen Grafiken, durch diese ersetzen. Daten in der zu INHALT angegebenen Datei zuerst löschen.
	= -	* Daten in der ZIEL-Datei und in der zu INHALT angegebenen Datei nicht löschen. Enthält die ZIEL-Datei schon eine Grafik mit der gleichen Nummer wie eine der neuen Grafiken, so wird im Dialog nachgefragt, ob die Verarbeitung abgebrochen oder die Grafik ersetzt werden soll; wurde das Makro im Batch aufgerufen, so wird die Verarbeitung abgebrochen, wenn schon eine solche Grafik vorhanden ist.
NUMMER	= n	Nummer, die die Grafik in der ZIEL-Datei erhalten soll. Sind zu QUELLE mehrere Dateien angegeben, so werden die in den einzelnen QUELL-Dateien jeweils enthaltenen Grafiken in der Reihenfolge der Angabe der QUELL-Dateien durchnummeriert, beginnend mit der angegebenen Nummer. Die größte Nummer, die vergeben werden kann, ist 999999.
ABSTAND	= x*y	Zahlenpaar; die beiden Werte x und y geben an, um

		wieviel Punkt der Nullpunkt der Grafik horizontal (x) bzw. vertikal (y) verschoben werden soll gegenüber der Stelle, an der die Grafik aufgerufen wird. Es müssen so viele Zahlenpaare angegeben werden wie Dateien zu QUELLE angegeben sind.
	= -STD-	Die Angaben zum Verschieben der Grafik werden der BoundingBox der Grafik entnommen: der Nullpunkt der Grafik wird um den für den linken Rand angegebenen X-Wert nach links und um den für den oberen Rand angegebenen Y-Wert nach unten verschoben.
	= 0*0	* Der Nullpunkt der Grafik soll nicht verschoben werden gegenüber der Stelle des Aufrufs der Grafik.
DREHEN	= n	Grafik um n Grad gegen den Uhrzeigersinn drehen. Ist mehr als eine Datei zu QUELLE angegeben, so werden alle Grafiken um den angegebenen Winkel gedreht.
	= -n	Grafik spiegeln und um n Grad gegen den Uhrzeigersinn drehen. Ist mehr als eine Datei zu QUELLE angegeben, so werden alle Grafiken gespiegelt und um den angegebenen Winkel gedreht.
	= -	* Grafik nicht drehen oder spiegeln.
FAKTOR	= n	Vergrößerungs- oder Verkleinerungsfaktor (in Prozent; eine Dezimalstelle möglich), mit dem die Größe der Grafik verändert werden soll. Es müssen so viele Faktoren angegeben werden wie Dateien zu QUELLE angegeben sind.
	= -STD-	* Keine Größenveränderung der Grafik.
	= -	Die zu QUELLE angegebene Datei enthält nicht Grafikdaten, sondern pdfmarks, die zum Einbinden in die Satzausgabe mit dem Makro *PSAUS aufbereitet werden sollen.
SCHNEIDEN	= x*y+x*y	2 Koordinatenpaare, die die linke obere und die rechte untere Ecke eines Rechtecks angeben, innerhalb dessen die eingebundene Grafik stehen muss; was über die Ränder dieses Rechtecks hinaussteht, wird abgeschnitten. Der Nullpunkt für diese Angaben ist die linke obere Ecke der Aussparung. Es müssen so viele Koordinatenpaare angegeben werden wie Dateien zu QUELLE angegeben sind.
	= -STD-	Die Angaben zum Schneiden werden der BoundingBox der Grafik entnommen: was über die an die linke obere Ecke der Aussparung verschobene BoundingBox hinaussteht, wird abgeschnitten.

	= -	* Die eingebundene Grafik soll nicht beschnitten werden.
NAME	= name	Name der Grafik in der ZIEL-Datei. Es müssen so viele Namen angegeben werden wie Dateien zu QUELLE angegeben sind.
	= -STD-	* Die Grafik erhält in der ZIEL-Datei den Namen »Abbildung n«, wobei n die zu NUMMER angegebene Nummer ist.
INHALT	= +	Neues Inhaltsverzeichnis der ZIEL-Datei ins Ablaufprotokoll ausgeben.
	= datei	Inhaltsverzeichnis in die angegebene Datei ausgeben.
	= -	* Kein Inhaltsverzeichnis ausgeben.
BINAER	= -	* Keine Binärinformation in der QUELL-Datei.
	= +	QUELL-Datei enthält binäre Bild-Information.
UMFANG	= -STD-	* Normalfall, keine Angabe zum Umfang der Grafik-Datei(en)
	= +	Hat die zu QUELLE angegebene EPS-Datei eine Größe von mehr als ca. 500.000 Zeilen, sollen dennoch kurze Zeilen nicht zusammengefasst werden (näheres unten unter »Hinweise«).

## Leistung

Dieses Makro dient zur Vorbereitung des Einbindens von Grafiken und pdfmark-Anweisungen in die PostScript-Daten beim Aufbereiten der Satzausgabe-Daten mit dem Makro \*PSAUS. Die Grafiken, die in den Text eingebunden werden sollen, müssen als Encapsulated PostScript-Dateien (EPS) vorliegen. Für den Rest der Beschreibung sind Grafiken und pdfmark-Anweisungen als »Abbildung« bezeichnet.

Die einzubindenden Abbildungen müssen zu diesem Zweck mit Hilfe des Makros \*GRAFIK mit einer eindeutigen Nummer versehen und nach diesen Nummern geordnet auf einer eigenen Datei gesammelt werden, die bei einem anschließenden Aufruf des Makros \*PSAUS zur Spezifikation GRAFIK angegeben werden muss.

## Hinweise

Die EPS-Dateien sollten ohne Vorschau-Option erstellt sein und müssen den Anforderungen genügen, die in Kapitel 2.8 »Ensuring Portability« der »Encapsulated PostScript File Format Specification, Version 3.0« von Adobe genannt sind. Dazu gehört z. B. in Adobe Photoshop die Wahl von ASCII (nicht: binär) als Codierungs-Option.

Wenn viele große EPS-Dateien mit dem Makro \*GRAFIK bearbeitet werden sollen, kann es sinnvoll sein, nur die vom Makro \*GRAFIK erzeugten Zusatzinformationen für alle zu einem Satzauftrag gehörenden Grafiken in einer gemeinsame Datei (die erste zur Spezifikation ZIEL angegeben wird) abzulegen und die eigentlichen Grafikdaten auf mehrere Dateien zu verteilen. Der Name der Datei, in der die eigentlichen Grafikdaten abgelegt werden sollen, ist beim Aufruf des Makros \*GRAFIK zur Spezifikation ZIEL jeweils an zweiter Stelle anzugeben. Dies geschieht durch Angabe von zwei ZIEL-Dateien. Bei EPS-Dateien mit mehr als 500.000 Zeilen Umfang müssen in jedem Fall zwei ZIEL-Dateien angegeben werden, wovon die zweite vom Typ SEQ-P (nicht: RAN) sein muss und jeweils nur die Daten für eine einzige Grafik enthalten darf.

Falls eine zweite ZIEL-Datei angegeben ist, wird deshalb deren Name und der Träger, auf dem sie angemeldet ist, in die vom Makro \*GRAFIK erzeugte Zusatzinformation eingesetzt; diese wird in der ersten ZIEL-Datei abgelegt. Diese Zusatzinformation wird später vom Makro \*PSAUS benutzt, um jeweils die Datei anzumelden, die die Grafikdaten enthält. Die zweite ZIEL-Datei muss deshalb beim Aufruf des Makros \*GRAFIK auf dem selben Träger angemeldet sein, auf dem sie später durch das Makro \*PSAUS angemeldet werden soll.

Zu UMFANG sind normalerweise keine Angaben nötig; das Makro versucht bei großen EPS-Dateien mit mehr als 500.000 Zeilen durch Eliminieren nicht notwendiger Zeilenwechsel die Satzzahl so zu reduzieren, dass die Zieldatei auch im Editor noch bearbeitet werden kann. Bei manchen intern in der EPS-Datei verwendeten Prozeduren ist dies jedoch nicht statthaft; die eingebundene Grafik kann dann nicht mehr angezeigt werden. Nur in solchen Fällen muss UMFANG=+ angegeben werden.

## Erläuterungen

### Allgemeines

Bei der Aufbereitung mit dem Makro \*GRAFIK müssen die Abbildungen so verschoben und ggf. gedreht werden, dass sie in die beim Setzen dafür vorgesehenen Aussparungen passen. Wenn in einer zu QUELLE angegebenen Datei mehr als eine Abbildung steht oder wenn Teile einer Abbildung (z. B. eine Bildunterschrift) nicht mit übernommen werden soll, muss diese Abbildung zusätzlich ausgeschnitten werden. Dies geschieht mit Hilfe von entsprechenden Angaben zu den Spezifikationen ABSTAND, DREHEN und SCHNEIDEN.

### Angaben zur Spezifikation ABSTAND

Die x- und y-Abstände, die beim Makro \*GRAFIK zur Spezifikation ABSTAND anzugeben sind, können wie folgt bestimmt werden:

Man markiert in einem Ausdruck der Abbildung den Nullpunkt der Abbildung (das ist in der Regel die ursprüngliche linke untere Ecke des Blattes). Soll die Abbildung gedreht werden, so ist das Blatt anschließend so zu drehen, dass die Abbildung in der

endgültigen Ausrichtung steht. Dann markiert man in der (evtl. schon gedrehten) Abbildung die Stelle, die dem Aufruf der Abbildung in der Satzausgabe entsprechen soll (im Fall von Aussparungen in der Regel die linke obere Ecke der – evtl. schon gedrehten – Abbildung) und misst anschließend den Abstand des Nullpunkts der Abbildung in x- und y-Richtung von der zuletzt markierten Stelle, die dem Aufruf der Abbildung entspricht; auf diese Weise erhält man die x- und y-Werte, um die die Abbildung verschoben werden muss.

Die Angaben sind in (Didot-)Punkt zu machen.

Für x-Werte gilt: Liegt der Nullpunkt des in die endgültige Richtung gedrehten Ausdrucks links vom Ankerpunkt, so ist der gemessene Abstand als negativer x-Wert anzugeben; liegt er rechts, so ist der Abstand als positiver x-Wert anzugeben.

Für y-Werte gilt: Liegt der Nullpunkt des in die endgültige Richtung gedrehten Ausdrucks unterhalb vom Ankerpunkt, so ist der gemessene Abstand als negativer y-Wert anzugeben; liegt er oberhalb, so ist der Abstand als positiver y-Wert anzugeben.

### **Angaben zur Spezifikation** DREHEN

Muss die Abbildung gedreht werden, so ist zur Spezifikation DREHEN der Winkel, um den die Abbildung gedreht werden soll, in Grad anzugeben. Die Drehung erfolgt gegen den Uhrzeigersinn. Eine durch negative Angabe verlangte Spiegelung wird vor dem Drehen ausgeführt.

### **Angaben zur Spezifikation** SCHNEIDEN

Nachdem die Abbildung so verschoben und gedreht ist, dass sie an der richtigen Stelle steht, wird sie mit den Angaben zu SCHNEIDEN beschnitten. Dazu wird über den (bereits verschobenen und ggf. gedrehten) Ausdruck ein Rechteck gelegt, innerhalb dessen der gewünschte Ausschnitt steht. Der Nullpunkt für die Koordinatenangaben der linken oberen und der rechten unteren Ecke dieses Rechtecks ist die linke obere Ecke der Aussparung. Die Koordinatenangaben können positiv (x-Richtung nach rechts, y-Richtung nach oben) oder negativ (x-Richtung nach links, y-Richtung nach unten) gemacht werden. Was außerhalb dieses Rechtecks steht, wird abgeschnitten, d. h. nicht mit ausgegeben.



# Hieroglyphen in Grafikdatei einfügen

#\*HIEROGR

## Makro:

#\*HIEROGR

QUELLE	= datei	Name der Datei, die eine Hieroglyphe im PS-Export-Format enthält.
ZIEL	= datei	Name der Datei, in der die aufbereiteten Hieroglyphen als Grafiken zum Einbinden in die Ausgabe mit dem Makro *PSAUS gesammelt werden sollen. Es können, durch Apostroph getrennt, zwei Dateinamen angegeben werden. In diesem Fall werden die von TUSTEP hinzugefügten Zusatzinformationen für die einzelnen Hieroglyphen in die erste, die eigentlichen Grafikdaten in die zweite ZIEL-Datei ausgegeben.
LOESCHEN	= -STD-	In der ZIEL-Datei Grafiken, die die gleiche Nummer haben wie eine der neuen Grafiken, durch diese ersetzen.
	= -	* Enthält die ZIEL-Datei schon eine Grafik mit der gleichen Nummer wie eine der neuen Grafiken, so wird im Dialog nachgefragt, ob die Verarbeitung abgebrochen oder die Grafik ersetzt werden soll; wurde das Makro im Batch aufgerufen, so wird die Verarbeitung abgebrochen, wenn schon eine solche Grafik vorhanden ist.
NUMMER	= n	Nummer, die die Grafik in der ZIEL-Datei erhalten soll. Die größte Nummer, die vergeben werden kann, ist 999999.
ABSTAND	= x*y	Zahlenpaar; die erste Zahl x gibt den horizontalen Abstand des linken Randes der Grafik vom linken Rand der im Text dafür vorgesehenen Aussparung in Punkt an; die zweite Zahl y gibt den vertikalen Abstand des unteren Randes der Grafik vom oberen Rand der Aussparung in Punkt an. Nur ganzzahlige Angaben sind möglich.
	= -STD-	* (Gleichbedeutend mit 0*-3): Die Hieroglyphe sitzt bei FAKTOR=-STD- (gleichbedeutend mit FAKTOR=12) etwa auf der Geviertunterkante (d. h. sie beginnt mit den Unterlängen).

---

SPIEGELN	= 0	* Hieroglyphe nicht spiegeln.
	= 1	Hieroglyphe horizontal spiegeln.
FAKTOR	= n	Faktor (in Prozent, ganzzahlig), um den die Größe der Hieroglyphe verändert werden soll.
	= -STD-	* Faktor 12 (passend zu 12-Punkt-Schriften).
DICKTEN	= datei	Dicke (Breite in 1/1000 Geviert der zu FAKTOR angegebenen Schriftgröße) der Hieroglyphe in die angegebene Datei ausgeben. Der Dicktenwert wird immer hinten an die Datei angehängt.
	= -	* Dicktenwerte ins Ablaufprotokoll ausgeben.
MODUS	= -	* Normalfall: Hieroglyphe ohne Kartusche.
	= +	Um die Hieroglyphe wird eine Kartusche gezeichnet.
	= ++	Um die Hieroglyphe wird eine Kartusche mit senkrechtem Strich gezeichnet.

## Leistung

Dieses Makro dient zum Umwandeln von Hieroglyphen in Grafiken, die in die Satzausgabe mit dem Makro \*PSAUS eingebunden werden können. Die Hieroglyphen müssen mit dem Programm Glyph for Windows des CCER der Universität Utrecht erstellt und im Menü ANSICHT mit der Option PS-Export abgespeichert worden sein (Dateiname: xxxx.eps).

Innerhalb des Makros \*HIEROGR wird das Makro \*GRAFIK benutzt, um die Hieroglyphen als Grafiken aufzubereiten und in der Form auf die ZIEL-Datei(en) zu schreiben, die für die Ausgabe mit dem Makro \*PSAUS benötigt wird. Deshalb können Hieroglyphen zusammen mit anderen Grafiken in einer gemeinsamen Datei stehen.

## Hinweis

Wenn viele Hieroglyphen mit dem Makro \*HIEROGR bearbeitet werden sollen, kann es sinnvoll sein, nur die vom Makro \*HIEROGR erzeugten Zusatzinformationen für alle zu einem Satzauftrag gehörenden Hieroglyphen in einer gemeinsamen Datei (die als erste zur Spezifikation ZIEL angegeben wird) abzulegen und die zugehörigen Grafikdaten auf mehrere Dateien zu verteilen. Der Name der Datei, in der die eigentlichen Grafikdaten abgelegt werden sollen, ist beim Aufruf des Makros \*HIEROGR zur Spezifikation ZIEL jeweils an zweiter Stelle anzugeben.

Falls eine zweite ZIEL-Datei angegeben ist, wird deshalb deren Name und der Träger, auf dem sie angemeldet ist, in die vom Makro \*HIEROGR erzeugte Zusatzinformation eingesetzt; diese wird in der ersten ZIEL-Datei abgelegt. Diese Zusatzinformation wird später vom Makro \*PSAUS benutzt, um jeweils die Datei anzumelden, die die Grafikdaten enthält. Die zweite ZIEL-Datei muss deshalb beim Aufruf des Makros \*HIEROGR auf dem selben Träger angemeldet sein, auf dem sie später durch das Makro \*PSAUS angemeldet werden soll.

# Kerningtabellen in KOPIERE-Parameter umwandeln

#\*KERNPAR

## Makro:

#\*KERNPAR

QUELLE	=	datei	Name der Datei, die den AFM-File (Adobe Font Metrics) einer PostScript-Schrift mit Kerning-Information enthält.
ZIEL	=	datei	Name der Datei, in die die KOPIERE-Parameter zum Bearbeiten der QUELL-Datei für das Satzprogramm geschrieben werden sollen.
LOESCHEN	=	-	* Daten in der ZIEL-Datei nicht löschen.
	=	+	Daten in der ZIEL-Datei zuerst löschen.

## Leistung

Mit diesem Makro können Kerningtabellen aus der AFM (Adobe Font Metrics)-Datei einer PostScript-Schrift in Parameter für das Kommando #KOPIERE umgewandelt werden, mit dem anschließend die Kerning-Information in QUELL-Dateien für das Satzprogramm eingebracht werden kann.

Die Steueranweisungen für das Satzprogramm werden dabei berücksichtigt und unverändert belassen, auch wenn (z. B. in Makros) Zeichenkombinationen vorkommen, die auch in der Kerningtabelle enthalten sind. Silbentrennungen der QUELL-Datei werden nicht aufgehoben.

Es wird außerdem vorausgesetzt, dass die Datei, die mit den erzeugten Parametern bearbeitet wird, nur Text enthält, der aus der entsprechenden Schrift gesetzt werden soll. Gegebenenfalls muss die QUELL-Datei für das Satzprogramm zuvor auf entsprechend viele Einzeldateien aufgeteilt werden, die nach der Bearbeitung und vor dem Setzen wieder zusammengemischt werden können.

In vielen Fällen dürfte es einfacher sein, das Kerning erst bei \*PSAUS durchzuführen, indem dort eine mit dem Makro \*PSKERNFILE erzeugte Datei als zweite Datei zur Spezifikation SCHRIFTEN angegeben wird.

## Masken für die Satzausgabe definieren

#\*MASKE

### Makro:

#\*MASKE

QUELLE	= datei	Name der Datei mit den Angaben zu den Masken.
	= *	Angaben zu den Masken folgen, durch *EOF abgeschlossen, auf den Makroaufruf.
ZIEL	= datei	Name der Datei, in die die für die Satzausgabe aufbereiteten Masken geschrieben werden sollen.
MODUS	= P	Masken sollen für PostScript-Ausgabe aufbereitet werden.
LOESCHEN	= -	* Daten in der PROTOKOLL-Datei nicht löschen. (Ist die ZIEL-Datei nicht leer, muss LOESCHEN=+ angegeben werden.)
	= +	Daten in der ZIEL- und in der PROTOKOLL-Datei zuerst löschen.
PROTOKOLL	= +	* Das Protokoll soll ins Ablaufprotokoll ausgegeben werden.
	= -STD-	Das Protokoll soll in die Standard-Protokoll-Datei ausgegeben werden.
	= datei	Name der Datei, in die das Protokoll ausgegeben werden soll.

### Leistung

Mit diesem Makro können Masken definiert werden, die bei der Ausgabe über die einzelnen Seiten belichtet / gedruckt werden sollen. Masken können waagerechte und senkrechte Linien sowie Texte enthalten; für rechte und linke Seiten können verschiedene Masken angegeben werden.

## Angaben zur Maske (Spezifikation QUELLE)

SLmn xxx yy1 yy2 senkrechte Linie

Länge: Vielfaches von 6 Punkt

Lage: xxx Punkt vom linken Rand

Erstreckung: yy1 bis yy2 von oben

WLmn yyy xx1 xx2 waagerechte Linie

Länge: Vielfaches von 3 Punkt

Lage: yyy Punkt von oben

Erstreckung: xx1 bis xx2 vom linken Rand

TXTn xxx yyy nrspb nrkur nrgro

Text Text der Maske

Lage: xxx Punkt vom linken Rand, yyy Punkt vom oberen Anfangspunkt

Schrift: aus Speicherbereich (Umschaltbereich) nrspb, gerade (nrkur=0) / schräg (nrkur=1), Größe nrgro Punkt

Der Text wird in den darauf folgenden Zeilen (ab Spalte 1) angegeben. Diese Zeilen können bis zu 600 Zeichen lang sein. Dabei können nur Buchstaben, Ziffern, Leerzeichen und Satzzeichen unverschlüsselt angegeben werden; alles andere ist mit hardware-nahen Kommandos anzugeben. Diese Kommandos sind als 2- oder 3-stellige Dezimalzahlen zwischen 0 und 255, durch Leerzeichen getrennt, in runde Klammern einzuschließen.

bei SL und WL bedeutet:

**m = 1** magere Linie

**m = 2** halbfette Linie

**m = 3** fette Linie

bei SL, WL, TXT bedeutet:

**n = 0** gilt für alle Seiten

**n = 1** gilt nur für ungerade Seiten

**n = 2** gilt nur für gerade Seiten

Für xxx gilt als 0-Punkt der linke Materialrand (der NFREI0 Punkt vor dem Satzspiegelrand liegt; vgl. Parameter MON im Programm SATZ).

Für yyy liegt der 0-Punkt um 24 Punkt unter der Kopfzeile, die Seitenkennung, Projekt, Datum und Uhrzeit enthält, und 24 Punkt oberhalb der (Geviert)-Oberkante der ersten zu belichtenden Zeile (die gegenüber einer Belichtung ohne Maske um 24 Punkt nach unten verschoben wird). Diese Zeile ist der oben stehende (lebende oder tote) Kolumnentitel oder, falls solche fehlen oder unten auf der Seite stehen, die erste Textzeile.

# SATZ-Ausgabedateien zusammenmontieren

#\*MONT

## Makro:

#\*MONT

QUELLE	= datei	Name der AUSGABE-Dateien von #SATZ, die zusammenmontiert werden sollen. Die Namen der Dateien sind durch Apostroph zu trennen.
ZIEL	= datei	Name der Datei, in die die fertig montierten und zur Belichtung aufbereiteten Seiten geschrieben werden sollen.
LOESCHEN	= +	Daten in der ZIEL-Datei zuerst löschen.

## Leistung

Mit diesem Makro können mehrere SATZ-Ausgabedateien so zusammenmontiert werden, dass bei der Ausgabe Spalten mit gleicher Spaltennummer übereinander belichtet werden. Bei den Satzprogrammläufen, mit denen diese Dateien erstellt wurden, müssen die Angaben zu den Parametern SCH und BIL jeweils identisch sein; die Angaben zu Parameter MON müssen jeweils die gleiche Gesamtbreite (einschließlich der Freiräume vor der ersten und nach der letzten Spalte) ergeben.

Die ZIEL-Datei von \*MONT kann wieder QUELL-Datei eines neuen nachfolgenden Aufrufs von \*MONT werden. Auf diese Weise können nacheinander bis zu 1000 Dateien zusammenmontiert werden.

# PS-Dateien nach OpenType-Konventionen umcodieren

#\*PRECOMPOSED

## Makro:

#\*PRECOMPOSED

QUELLE	=	datei	Name einer von SATZ und *PSAUS erzeugten PostScript-Datei
ZIEL	=	datei	Name einer FDF-Datei (zweckmäßigerweise mit der extension .ps), in die die umgewandelten Postscript-Daten geschrieben werden sollen.
LOESCHEN	=	+	Daten in der ZIEL-Datei zuerst löschen.
SCHRIFTEN	=	-	keine benutzereigenen Schriften.
	=	datei	Name der Datei, die bei *PSAUS zu SCHRIFTEN angegeben wurde.
ZUORDNUNG	=	-	* keine Zuordnung von Type-1- zu OpenType-Schriften. Es wird vorausgesetzt, dass alle verwendeten Fonts die generierten precomposed characters enthalten.
	=	*	Die Angaben zur Zuordnung folgen auf den Makro-Aufruf und sind durch *eof abgeschlossen. Sie haben die Form <code>bnr:type1nr=openotypenr</code> , wobei <code>bnr</code> die Nummer des Umschaltbereichs ist, in dem die Schrift mit der Nummer <code>type1nr</code> beim Parameter <code>SCH</code> im Satzprogramm angegeben ist, und <code>openotypenr</code> die Nummer der OpenType-Schrift ist, aus der die Type-1-Schrift mit der Nummer <code>type1nr</code> erzeugt wurde.  Für nicht aufgeführte Fonts wird keine Umwandlung vorgenommen.  Beide Schriftnummern müssen in der zu SCHRIFTEN genannten Datei aufgeführt sein, ausgenommen die mit 3 beginnenden Nummern der auch ohne eine solche Datei verfügbaren Fonts.
LISTE	=	+	* Ans Ende der PS-Datei wird eine Liste der verwendeten precomposed characters und der übrigen mit <code>glyphshow</code> ausgegebenen Zeichen angehängt.
	=	-	keine Liste.

## Leistung

Das TUSTEP-Satzprogramm arbeitet mit fliegenden Akzenten. Wörter, die Akzentbuchstaben enthalten, werden deshalb bei der Suche in PDF-Dateien nicht gefunden, wenn diese ausschließlich mit SATZ und \*PSAUS vorbereitet wurden. Durch die Nachbereitung der von \*PSAUS erzeugten PS-Dateien durch \*PRECOMPOSED wird dies vermieden.

Derzeit sind in \*PRECOMPOSED außer dem lateinischen nur das griechische und das kyrillische (russische) Alphabet berücksichtigt.

Bei allen Fonts, für die precomposed characters generiert werden, setzt das Makro voraus, dass diese Zeichen in den entsprechenden Fonts vorhanden und mit den vorgesehenen Namen versehen sind.

Bei ZUORDNUNG=– werden für lateinische Fonts nur die im Code CP1252 (Windows Western, cf. Handbuch unter »Code-Tabellen / Tabellen für den ASCII-Code) enthaltenen precomposed characters generiert. Für nicht-lateinische Fonts wird keine Umwandlung vorgenommen, außer für die Fonts mit den Nummern 31751 bis 31763 (Newton Greek und Newton Cyrillic) und 32501 bis 32613 (griechische und kyrillische Libertine- bzw. Biolinum-Fonts).

Bei ZUORDNUNG=\* müssen alle Fonts aufgeführt werden, für die eine Umwandlung in precomposed characters vorgenommen werden soll. Dabei wird bei lateinischen Fonts vorausgesetzt, dass auch die im Text verwendeten Zeichen, die über den Zeichenvorrat von CP1252 hinausgehen und im Makro \*PRECOMPOSED zur Umwandlung in precomposed characters vorgesehen sind, in den zu ZUORDNUNG aufgeführten OpenType-Fonts vorhanden und mit den vorgesehenen Namen versehen sind. Dazu gehören z. Zt. auch die in CP1250 = Windows Central European enthaltenen Zeichen sowie die über CP1252 und CP1250 hinausgehenden Zeichen von Unicode Latin-Extended A = Unicode-Adressen von 0101 bis 017F. (Eine Erweiterung auf die in Unicode Latin Extended Additional enthaltenen Zeichen, für die eine TUSTEP-Codierung vorgesehen ist, ist geplant.)

Wenn dies nicht der Fall ist, oder wenn dort vorhandene Zeichen einen anderen als den im Makro \*PRECOMPOSED verwendeten Namen haben, kann es vorkommen, dass nicht das erwartete Zeichen erzeugt wird. Es wird deshalb empfohlen, die am Ende der ZIEL-Datei angehängte Liste der mit glyphshow ausgegebenen Zeichen und deren Darstellung sorgfältig zu kontrollieren. Für den Produktionslauf kann die Ausgabe dieser Liste mit LISTE=– unterdrückt werden.



# Quelldaten und Parameter aus Satzprotokoll extrahieren

#\*PROT2QU

## Makro:

#\*PROT2QU

QUELLE	= datei	Name der Datei mit dem Protokoll eines Satzprogramm- laufs.
ZIEL	= datei	Name der Datei für den Textteil
MODUS	=	(wird noch nicht ausgewertet)
LOESCHEN	= -	* Daten in der PROTOKOLL-Datei nicht löschen. Die zu ZIEL, FUSSNOTEN und SATZPAR angegebenen Da- teien müssen in diesem Fall leer sein.
	= +	Daten in den zu ZIEL, PROTOKOLL, FUSSNOTEN und SATZPAR angegebenen Dateien zuerst löschen.
PARAMETER	= -	* Keine Parameterangaben.
PROTOKOLL	= +	* Fehlerprotokoll ins Ablaufprotokoll ausgeben
	= -STD-	Fehlerprotokoll in die Standard-Protokoll-Datei aus- geben.
	= datei	Name der Datei für das Fehlerprotokoll.
FUSSNOTEN	= datei	Name der Datei für die Fußnoten
SATZPAR	= datei	Name der Datei für die Satzparameter

## Leistung

Mit diesem Makro können aus der PROTOKOLL-Datei eines Satzprogramm-Laufes die Quelldaten für Text und Fußnoten sowie die Parameter für den betreffenden Satzlauf wieder hergestellt werden. In der PROTOKOLL-Datei des Satzlaufs aufgelöste Makros werden dabei nicht wieder zu Makros.

# PostScript-Datei ausdrucken: 4 Seiten/A4-Blatt

#\*PS4A4

## Makro:

#\*PS4A4

QUELLE	= datei	Name der PostScript-Datei (DATEI von *PSAUS), die ausgegeben werden soll.
GERAET	= -STD-	* Die System-Variable TUSTEP_PRN enthält den Namen des Druckers, auf dem ausgegeben werden soll; falls diese System-Variable nicht definiert ist, wird der Druckernamen vom Betriebssystem erfragt.
	= name	Name des Druckers, auf dem ausgegeben werden soll, oder Name einer System-Variablen, die den Namen dieses Druckers enthält.
	= datei	Name einer Datei, auf der die (verkleinerte) Ausgabe als PostScript-Daten gespeichert werden soll.
FREI	= n	Zahl (1-3) der (Viertel-)Seiten, die bei der Ausgabe auf dem ersten DIN A4-Blatt freigelassen werden sollen.
	= 0	* Das erste DIN A4-Blatt soll ganz gefüllt werden.
FAKTOR	= -STD-	* Anhand der Angaben, die beim Erstellen der QUELL-Datei mit dem Makro *PSAUS zur Spezifikation RAHMEN gemacht wurden, errechnet das Makro den größten Faktor, mit dem die Seite ohne Informationsverlust ausgegeben werden kann.
	= nn	Abbildungsmaßstab (in Prozent, ohne Kommastellen), in dem die Seiten ausgegeben werden sollen.
EINRUECKEN	= 0+0	* Keine zusätzliche Einrückung.
	= x+y	Zusätzliche Einrückung von links und von oben (wird nur ausgewertet, wenn zu FAKTOR ein Abbildungsmaßstab nn angegeben ist).
ABSTAND	= 0+0	* Kein zusätzlicher Abstand zwischen den Viertelseiten.
	= x+y	Zusätzlicher Abstand in x- und y-Richtung zwischen den Viertelseiten (wird nur ausgewertet, wenn zu FAKTOR ein Abbildungsmaßstab nn angegeben ist).
OPTIONEN	= -STD-	* Normalfall: Quellen sind im Hochformat aufbereitet.

	= RL	wie -STD-, aber Seiten von rechts nach links anordnen (z. B. für hebräische oder arabische Texte)
	= quer	Quellen sind im Querformat aufbereitet.
	= D_LANG	Papier doppelseitig bedrucken; Blättern an der langen Papierkante (für Ausgabe im Hochformat).
	= D_KURZ	Papier doppelseitig bedrucken; Blättern an der kurzen Papierkante (für Ausgabe im Querformat).
ZUSATZ	= . . .	Name einer System-Variablen, die zusätzliche Angaben zum Ausgabegerät für das Betriebssystem enthält (vgl. System-Variable TUSTEP_LPR). Welche Angaben sinnvoll oder im Einzelfall notwendig sind, ist beim jeweiligen Rechenzentrum bzw. beim Systemverwalter zu erfragen.

## Leistung

Mit diesem Makro kann eine PostScript-Datei, die mit dem Makro \*PSAUS erstellt wurde (bei dessen Aufruf also die Ausgabe nicht direkt auf den Drucker, sondern auf eine Datei erfolgt ist), Platz sparend auf einem PostScript-Drucker ausgegeben werden. Die Seiten werden dabei so verkleinert, dass 4 Seiten auf einem DIN A4-Blatt untergebracht werden können. War bei \*PSAUS kein Rahmen angegeben, so werden die Seiten auf etwa ein Viertel ihrer ursprünglichen Größe verkleinert.

## Hinweis

Im Unterschied zur direkten Ausgabe mit dem Makro \*PSAUS werden bei der Druckausgabe mit dem Makro \*PS4A4 auch umfangreiche Dateien nicht in Portionen unterteilt. Eine Unterteilung muss ggf. beim Erstellen der zu QUELLE angegebenen Datei mit dem Makro \*PSAUS über die Spezifikation SEITEN erfolgen.

Die Ausgabe mit dem Makro \*PS4A4 ist in der Regel zum eigentlichen Korrekturlesen nicht mehr geeignet, weil die Buchstaben zu klein werden. Sie empfiehlt sich aber, um Papier und Kosten zu sparen, wenn nur noch Dinge wie Anordnung des Textes, Füllungsgrad der Seiten, Vorhandensein von Seitenzahlen, Kolumnentiteln etc. kontrolliert werden sollen.

Für die Ausgabe von 2 bzw. 8 bzw. 16 Seiten pro A4-Blatt stehen die Makros \*PS2A4 bzw. \*PS8A4 bzw. \*PS16A4 zur Verfügung.

# Satzausgabe auf PostScript-Geräten

#\*PSAUS

## Makro:

#\*PSAUS

QUELLE	= datei	Name der Datei mit der Satzausgabe (Spezifikation AUSGABE des Kommandos #SATZ), die in PostScript umgewandelt und ausgegeben werden soll.
SEITEN	= n	Nummer der Seite (Spalte) der QUELL-Datei, die umgewandelt werden soll; mehrere Angaben, durch Apostroph getrennt, möglich. Sind dabei Seitennummern angegeben, die größer sind als die Nummer der letzten in der QUELL-Datei vorhandenen Seite, so wird die Ausführung des Makros abgebrochen. Dies kann durch ein »-« vor der ersten Seitennummer verhindert werden.
	= m-n	Angabe des Bereichs (von Spalte m bis Spalte n) der QUELL-Datei, der umgewandelt werden soll; mehrere Angaben, durch Apostroph getrennt, möglich. Sind dabei Seitennummern angegeben, die größer sind als die Nummer der letzten in der QUELL-Datei vorhandenen Seite, so wird die Ausführung des Makros abgebrochen. Dies kann durch ein »-« vor der ersten Seitennummer verhindert werden.
	= m(n)	Angabe des Bereichs (von Spalte m an n Spalten) der QUELL-Datei, der umgewandelt und ausgegeben werden soll; mehrere Angaben, durch Apostroph getrennt, möglich.
	= 0(999999)	* Die ganze zu QUELLE angegebene Datei soll umgewandelt und ausgegeben werden.
	= 0(999990)	wie 0(999999), jedoch wird Abbruch wegen undefinierter vertikaler Positionen verhindert. (Vorsicht: entsprechende Fehler in der PostScript-Datei bleiben dann möglicherweise unentdeckt.) Wenn nicht alles ausgegeben werden soll, sondern nur eine Auswahl von Seiten, so kann der Abbruch verhindert werden, indem als letzte Seitennummer 999990 angegeben wird.
FAKTOR	= n	Vergößerungs- oder Verkleinerungsfaktor (linear), mit dem ausgegeben werden soll; Angabe in Prozent

		der beim Satz angegebenen Werte, Schrittweite: 0.01%. Um den angegebenen Faktor werden auch die Angaben zu EINRUECKEN, LINIEN und RAHMEN verändert, nicht jedoch die Angaben zu DREHEN.
	= $n_x * n_y$	Wie n, jedoch können, durch »*« getrennt, für die x- und die y-Richtung getrennte Werte für die Skalierung angegeben werden.
	= $n' n_x$	Werden zwei Faktoren, durch Apostroph getrennt, angegeben, so wird nach der Skalierung der Ausgabe mit dem ersten angegebenen Faktor die Ausgabe in X-Richtung noch einmal mit dem zweiten angegebenen Faktor skaliert.
	= 100	* Ausgabe in der beim Satzprogrammmlauf gewählten Größe (100%).
EINRUECKEN	= $m+n$	Angaben, um wieviel Punkt vom linken (m) und vom oberen (n) Materialrand beim Drucken zusätzlich eingerückt werden soll. Negative Angaben möglich. Soll (z. B. beim Drucken auf die Vorder- und Rückseite eines Blattes) jede zweite Seite um einen davon verschiedenen Betrag eingerückt werden, so kann zu diesem Zweck, durch Apostroph vom ersten Wertepaar getrennt, ein zweites Paar von Werten angegeben werden.
	= $!m+n$	Soll die erste angegebene Einrückung jeweils für Seiten mit ungerader Nummer (=rechte Seiten), die zweite angegebene Einrückung für Seiten mit gerader Nummer gelten, so kann dies durch ein ! vor dem ersten Zahlenwert (bei zweispaltigem Satz durch !!) angegeben werden.
	= 40+40	* Beim Drucken vom linken und vom oberen Materialrand um jeweils 40 Punkt einrücken.  Wird zu OPTIONEN weder A4U noch A3U angegeben, so wird die Ausgabe zusätzlich um 10 Punkt vom linken und vom oberen Papierrand eingerückt; die BoundingBox wird entsprechend mit verschoben.
LINIEN	= $x_1 * y_1 + x_2 * y_2$	Angabe von bis zu 6 Paaren von x-y-Koordinaten (durch Apostroph voneinander getrennt), die bei der Ausgabe durch Linien verbunden werden sollen.
	= $x * y +$	Wird statt zweier Koordinatenpaare nur ein Koordinatenpaar, gefolgt von einem »+«, angegeben, so wird an der durch diese Koordinaten bezeichneten Stelle ein Linienkreuz mit einer Balkenlänge von je 12 Punkt ausgegeben.

	=	[x1*y1+x2*y2]	Zwei Koordinatenpaare (in eckigen Klammern), die die linke obere und die rechte untere Ecke eines Rechtecks beschreiben. Sie können statt der ersten vier Paare von x-y-Koordinaten angegeben werden, wenn diese ein Rechteck beschreiben sollen. Danach können, durch Apostroph getrennt, noch zwei Paare von x-y-Koordinaten für zwei weitere Linien angegeben werden.
	=	-	* Keine Linien in die Ausgabe einzeichnen.
LOESCHEN	=	+	Daten in der PROTOKOLL-Datei und ggf. in der zu DATEI angegebenen Datei zuerst löschen.
KOPF	=	+	* Der vom Satzprogramm erzeugte Spaltenkopftext soll mitgedruckt werden.
	=	n	Der vom Satzprogramm erzeugte Spaltenkopftext soll um n Punkt nach oben verschoben werden.
	=	-	Der vom Satzprogramm erzeugte Spaltenkopftext soll nicht gedruckt werden.
PROTOKOLL	=	datei	Name der Datei, in die eine Liste der Nummern der gedruckten Seiten und der eingebundenen Grafiken ausgegeben werden soll.
	=	-STD-	Die Protokoll-Ausgabe erfolgt in die Standard-Protokoll-Datei.
	=	+	* Die Protokoll-Ausgabe erfolgt ins Ablaufprotokoll.
GRAFIK	=	datei	Name der Datei, die die Grafiken (mit dem Makro *GRAFIK aufbereitete EPS-Dateien) oder Bookmarks enthält, die zusammen mit dem Text ausgegeben werden sollen. Bis zu 8 Dateinamen können angegeben werden.
	=	datei!	Wenn die Grafiken auf Kopf- und Grafikdatei aufgeteilt sind oder in der 4. Zeile jeder Grafik die BoundingBox-Information zu dieser Grafik steht, werden statt der Grafik Grauraster in der Größe der Grafik ausgegeben. Grafik-Nummer und -Name werden in das Grauraster eingetragen.
	=	-	* Es sind keine Grafiken oder Bookmarks mit auszugeben.
MASKE	=	datei	Name der Datei, die (mit dem Makro *MASKE vorbereitete) Masken enthält, die über den Text gedruckt werden sollen. Ist zu MASKE eine Datei angegeben, so wird am oberen Seitenrand (nach dem Kopftext, wenn dessen Ausgabe nicht unterdrückt wird) ein zusätzlicher

		Vorschub von 24 Punkt eingefügt, um Platz für eine evtl. in der Maske enthaltene Textzeile am oberen Seitenrand zu schaffen.
	= -	* Es sollen keine Masken mit ausgegeben werden.
OPTIONEN	= A4	* Ausgabe für DIN A4-Format aufbereiten; Ausgabe auf dem A4-Blatt nach oben positionieren.
	= A4M	Ausgabe für DIN A4-Format aufbereiten; angegebenen Rahmen in die Blattmitte rücken.
	= A4U	Wie A4, aber Ausgabe nach unten positionieren.
	= A3	Ausgabe für DIN A3-Format aufbereiten; Ausgabe auf dem A3-Blatt nach oben positionieren.
	= A3U	Wie A3, aber Ausgabe nach unten positionieren.
	= A2	Ausgabe für DIN A2-Format aufbereiten; Ausgabe auf dem A2-Blatt nach oben positionieren.
	= A1	Ausgabe für DIN A1-Format aufbereiten; Ausgabe auf dem A1-Blatt nach oben positionieren.
	= A0	Ausgabe für DIN A0-Format aufbereiten; Ausgabe auf dem A0-Blatt nach oben positionieren.
	= SMPLX	Papier einseitig bedrucken (Normalfall).
	= D_LANG	Papier doppelseitig bedrucken; Blättern an der langen Papierkante (für Ausgabe im Hochformat).
	= D_KURZ	Papier doppelseitig bedrucken; Blättern an der kurzen Papierkante (für Ausgabe im Querformat).
	= ghostview	Die Ausgabe in die Fremd-Datei soll später nicht gedruckt, sondern mit GHOSTVIEW am Bildschirm ausgegeben werden.
	= SR	* Ausgabe seitenrichtig.
	= SV	Ausgabe seitenverkehrt.
	= PLABEL	Beim späteren Umwandeln der erzeugten PS-Datei nach PDF soll für die Seitennavigation nicht nur die laufende Nummer der Seite und die Gesamtzahl der Seiten angezeigt werden, sondern zusätzlich jeweils die aktuelle Seitennummer (derzeit nur bei Spaltennummer = Seitennummer sinnvoll).  (mehrere Angaben durch Apostroph trennen).
UMSCHLAG	= -	* Nur vom System erzeugtes Deckblatt drucken.
	= +	Zusätzlich TUSTEP-Deckblatt drucken.
GERAET	= -STD-	Die System-Variable TUSTEP_PRN enthält den Namen des Druckers, auf dem ausgegeben werden soll;

		falls diese System-Variable nicht definiert ist, wird der Druckername vom Betriebssystem erfragt.
	= name	Name des Druckers, auf dem ausgegeben werden soll, oder Name einer System-Variablen, die den Namen dieses Druckers enthält.
	= -	* Keine Ausgabe auf einen Drucker; die Satzausgabe soll auf die zur Spezifikation DATEI angegebene Datei erfolgen.
DATEI	= datei	Name der Fremd-Datei (zweckmäßigerweise mit der extension .ps), in die die PostScript-Ausgabe geschrieben werden soll. Durch Apostroph vom ersten Dateinamen getrennt kann ein zweiter Dateiname angegeben werden. In diese Datei wird eine Kopie des Inhalts der ersten Datei geschrieben, zu dem am rechten Rand die Meldungen des Satzprotokolls mit ausgegeben werden. Angaben zu GERAET und DATEI schließen sich gegenseitig aus.
	= -	* Die Satzausgabe soll auf den zur Spezifikation GERAET angegebenen Drucker erfolgen.
TYP	= PS	* Aufbereitung für PostScript-Drucker, Hochformat.
	= PS-Q1	Wie PS, Querformat, eine Seite/Blatt.
	= PS-Q2	Wie PS, Querformat, zwei Seiten/Blatt.
	= PS-Q2L	Wie PS-Q2 (erste Seite links auf erstem Blatt).
	= PS-Q2R	Wie PS-Q2, aber erste Seite rechts auf erstem Blatt.
	= HEFT	Wie PS-Q2L. Dabei werden die Seiten auf einem Gerät mit Vorder- und Rückseitendruck in der Reihenfolge ausgegeben, dass der bedruckte Papierstapel, in der Mitte gefaltet, ein Heft ergibt. Dabei müssen immer Vielfache von 4 Seiten ausgegeben werden.
	= HEFTT	Wie HEFT, aber Rückseite bereits im Programm um 180 Grad drehen.
	= HEFT2	Wie HEFT, aber für zweispaltig gesetzte Texte.
	= HEFT2T	Wie HEFT2, aber Rückseite bereits im Programm um 180 Grad drehen.
	= RL	(nur als zweite Angabe, durch Apostroph von der ersten Angabe getrennt): bei der Ausgabe von zwei Seiten/Blatt soll die jeweils erste Seite rechts, die jeweils zweite Seite links ausgegeben werden (z. B. für hebräische oder arabische Texte).



ABSTAND	= -STD-	* (wird nur bei TYP=PS-Q2, PS-Q2L, PS-Q2R ausgewertet:) Abstand vom linken Rand der ersten zum linken Rand der zweiten Seite = halbe Breite des zu OPTIONEN angegebenen Formats (bei A4: 396 Punkt, bei A3: 560 Punkt).
	= nnn	Abstandsangabe in Punkt.
RAHMEN	= x1*y1+x2*y2	Koordinatenangaben (ganzzahlig, in Punkt) für ein Rechteck, innerhalb dessen sich alle zu diesem Auftrag gehörenden Seiten (einschließlich obere und untere Kolumnentitel, Marginalien, Zeilennummern, Oberlänge und Akzente der ersten Zeile, Unterlängen der letzten Zeile) und unter Berücksichtigung der horizontalen und vertikalen Verschiebungen durch die zu EINRUECKEN gemachten Angaben unterbringen lassen.
	= mmx1*y1+x2*y2	Wie x1*y1+x2*y2, Koordinatenangaben jedoch in Millimeter mit bis zu zwei Dezimalstellen.
	= mm!x1*y1+x2*y2	Wie mmx1*y1+x2*y2, jedoch werden diese Werte nicht durch die zu FAKTOR gemachten Angaben modifiziert.
	= :x1*y1+x2*y2	Für Testzwecke kann vor der ersten Zahl bzw. vor dem »mm« ein Doppelpunkt angegeben werden. In der PostScript-Ausgabe wird dann der Rahmen durch waagerechte bzw. senkrechte Linien sichtbar markiert.
	= ::x1*y1+x2*y2	Statt des Rahmens werden Schneidemarken für die zu RAHMEN angegebenen Koordinaten und Passkreuze in der Mitte der Seitenränder ausgegeben. Gegenüber der Ausgabe ohne Schneidemarken wird die Ausgabe zusätzlich zu den bei EINRUECKEN angegebenen Werten um weitere 21 Punkt nach rechts und nach unten (bei OPTION=A4U sowie bei ::mmx1*y1+x2*y2 nach oben) verschoben und die BoundingBox um 42 Punkt vergrößert, um auch die Schneidemarken darin unterbringen zu können. Außerdem wird eine TrimBox angelegt. Format der Passkreuze: Kreisradius innen (ausgefüllter Kreis) 3 pt, außen (Linie) 5 pt, Strichlänge vom Mittelpunkt aus 6 pt, Strichstärke 1/4 pt.
	= ;x1*y1+x2*y2	Wie ::x1*y1+x2*y2, jedoch nur Schneidemarken, keine Passkreuze ausgegeben.
	= -	* Keine Angaben zum Rahmen (Normalfall).
SCHRIFTEN	= datei	Name der Datei, in der die (mit dem Makro *PSFONT vorbereiteten) Dicktentabellen für benutzereigene Schriften stehen.

		Zusätzlich zu der Datei mit den Dickentabellen kann, durch Apostroph getrennt, der Name einer (mit *PSKERNFILE vorbereitete) Datei mit Kerningtabellen angegeben werden. Die Kerning-Information wird beim Erstellen der PostScript-Datei berücksichtigt. Die Schriftnummern müssen mit den Schriftnummern in der Datei mit den Dickentabellen übereinstimmen.
	= -	* Keine benutzereigenen Schriften.
ZUSATZ	= . . .	Name einer System-Variablen, die zusätzliche Angaben zum Ausgabegerät für das Betriebssystem enthält (vgl. System-Variable TUSTEP_LPR). Welche Angaben sinnvoll oder im Einzelfall notwendig sind, ist beim Systemverwalter zu erfragen.
DREHEN	= -	* Normalfall: keine weitere Verschiebung der fertigen Seite.
	= n' nx' ny	Die Seiten sollen (ggf. innerhalb des zu RAHMEN angegebenen Rechtecks) um den Winkel n gegen den Uhrzeigersinn gedreht sowie um nx Punkt in waagerechter Richtung und um ny Punkt in senkrechter Richtung verschoben werden. Soll jede zweite Seite um einen anderen Betrag gedreht und verschoben werden, so können zu diesem Zweck, durch einen Apostroph von den drei ersten Werten getrennt, weitere drei Zahlenwerte angegeben werden. Sollen die Angaben zum Verschieben nicht in (Didot-)Punkt (0,375 mm), sondern in PostScript-Maßeinheiten (DTP-Point, 1/72 Zoll = 0,3528 mm) gemacht werden, so ist hinter dem dritten bzw. sechsten Zahlenwert, durch Apostroph getrennt, ein »!« anzugeben.
HINTERGRUND	= datei	Name der Datei, die eine Grafik enthält, die auf jeder Seite als Hintergrundgrafik mit ausgegeben werden soll. Diese Grafik muss in der angegebenen Datei die Nummer 1 haben; sie wird ggf. bei der Ausgabe so beschnitten, dass sie innerhalb des zu RAHMEN angegebenen Rechtecks steht.
	= -	* Keine Hintergrundgrafik mit ausgeben.
EXTERN	= name	Namen von bis zu 9 PDF-Dateien (durch Apostroph getrennt), die die Dateinummern in externen Verknüpfungen (pdfmarks /ANN, subtype Link) in der Satzausgabe ersetzen sollen.
	= -	* Die QUELL-Datei enthält keine externen Verknüpfungen.

SATZPROT	= -	* Normalfall (wenn nur ein Dateiname zu DATEI angegeben ist)
	= name	<p>Namen der beiden PROTOKOLL-Dateien des Satzlaufes, mit dem die zu QUELLE angegebene Satzausgabe erzeugt wurde. Die Angabe ist erforderlich, wenn zu DATEI zwei PostScript-Dateien angegeben werden.</p> <p>Sollen auch die Fehlermeldungen vom Satz der Fußnoten am rechten Rand mit ausgegeben werden, so ist als dritte Datei zu SATZPROT die Datei mit den Satzprotokoll des Fußnotensatzes anzugeben.</p> <p>Die Zugehörigkeit der Dateien zum selben Satzlauf wird anhand des letzten Schreibzugriffs überprüft: die beiden ersten Dateien und die Quelldatei (=Satz-Ausgabe) müssen gleichzeitig erstellt sein, die dritte Datei darf höchstens eine Woche älter sein. Soll die letztere Prüfung unterbleiben, kann an vierter Stelle ein »!« angegeben werden. »!!« als vierte Angabe schaltet alle Prüfungen aus.</p>
PARAMETER	= -	* Keine Zusatzangaben zur Auswertung des Satzprotokolls
	= datei	Name der Datei mit Parametern für die Auswahl der Meldungen aus dem Satzprotokoll
	= *	Parameter für die Auswahl der Meldungen aus dem Satzprotokoll folgen auf den Makroaufruf und sind durch *EOF abgeschlossen.
NUMMERN	= -	* Keine Auswahl von Grafiken
	= n	Bis zu 8 (durch Apostroph getrennte) Nummern von Grafiken, die trotz »!« hinter dem Namen der sie enthaltenden Grafikdatei nicht als Grauraster, sondern als Grafik ausgegeben werden sollen.
	= -n	Bis zu 8 (durch Apostroph getrennte) Nummern von Grafiken, die bei »!« hinter dem Namen der sie enthaltenden Grafikdatei als einzige als Grauraster ausgegeben werden sollen.
	= ALLE	Trotz »!« hinter dem Namen der Grafikdatei sollen alle Grafiken in Originalform ausgegeben werden.
TITLE	= text	Der angegebene Text (nur lateinische Buchstaben, keine Akzente) wird hinter %%TITLE: im Prolog der PS-Datei eingesetzt
	= -	* im Prolog der PS-Datei wird %%TITLE: PSAUS eingesetzt

## Leistung

Mit diesem Makro kann die Ausgabe des Satzprogramms auf PostScript-Druckern ausgegeben oder zur Ausgabe auf nicht direkt erreichbaren PostScript-Druckern oder -Belichtern entsprechend aufbereitet in eine Datei geschrieben werden. Bei der Ausgabe können Abbildungen und (für die Umwandlung in PDF-Dateien) Bookmarks mit eingebunden werden, die mit einem der Makros \*GRAFIK, \*HIEROGR oder \*TEXGRAF bzw. \*BOOKMARKS vorbereitet wurden. Die Ausgabe kann gegenüber den Angaben beim Satzprogrammablauf vergrößert oder verkleinert erfolgen.

## Hinweis

Bei der Ausgabe von Seiten, die aus mehr als einer Spalte bestehen, sind zur Spezifikation SEITEN die Nummern der Spalten, nicht der Seiten anzugeben. In die PROTOKOLL-Datei werden die Nummern der ausgegebenen Spalten geschrieben.

Die zur Spezifikation GRAFIK angegebenen Dateien müssen mit einem der Makros \*GRAFIK, \*HIEROGR oder \*TEXGRAF vorbereitet worden sein. Wurden beim Aufruf von \*GRAFIK oder \*HIEROGR mehr als eine ZIEL-Datei angegeben, so ist zur Spezifikation GRAFIK die erste dieser Dateien anzugeben. Die jeweils zweite der beim Aufruf des Makros \*GRAFIK zu ZIEL angegebenen Dateien wird anhand der Angaben in der zur Spezifikation GRAFIK angegebenen Datei vor dem Einbinden einer Grafik selbständig angemeldet und wieder abgemeldet, falls sie zum Zeitpunkt des Makros \*PSAUS nicht bereits angemeldet ist.

Zur Spezifikation OPTIONEN den Wert A3 oder A3U anzugeben ist nur sinnvoll, wenn auch auf DIN A3-Papier ausgegeben wird (z. B. durch entsprechende Angabe zur Spezifikation GERAET). Wird auf DIN A3-Papier ausgegeben, ohne mit der Spezifikation OPTIONEN das DIN A3-Format anzugeben, so wird die Ausgabe so positioniert, als würde sie auf DIN A4-Papier erfolgen (im Hochformat also in die linke untere Ecke eines DIN A3-Blattes).

Die Angaben SMPLX, D\_LANG und D\_KURZ sind nur bei Ausgabe auf Druckern mit Duplex-Einrichtung sinnvoll. Außerdem setzen sie PostScript Level 2 voraus. Bei Ausgabe auf Datei statt auf Drucker ist die Angabe dieser Optionen nicht möglich. (Werden mit \*PSAUS erzeugte PostScript-Dateien mit dem Makro \*PSDR auf einen Drucker ausgegeben, so kann Duplex-Druck durch die Angabe der entsprechenden Spezifikationswerte beim Aufruf des Makros \*PSDR verlangt werden.)

Existiert eine (permanente) Datei mit Namen \*SASIM.ACC, so wird dort für jeden Aufruf Datum, Uhrzeit, Zahl der Seiten, Länge der erzeugten Drucker-Datei und Name des verwendeten Druckers abgelegt.

## Testhilfen

Um die Anordnung des Textes im vorgesehenen Satzspiegel besser kontrollieren zu können, kann man zur Spezifikation LINIEN bis zu 6 Paare von x-y-Koordinaten (x für horizontale, y für vertikale Positionen) angeben. Die Punktepaare, die durch je ein Paar von x-y-Koordinaten angegeben sind, werden durch eine Linie miteinander

verbunden. Die Maßangaben werden in typographischen Punkt (1 Punkt = 0,375 mm) erwartet; der Nullpunkt liegt um `NFREI0` Punkt (vgl. Parameter `MON` beim Programm `SATZ`) links vom Satzspiegel auf der Schriftgrundlinie des Spaltenkopftextes, der (außer bei `KOPF=-`) über jeder Spalte ausgegeben wird; er befindet sich 24 Punkt oberhalb der ersten Zeile.

Beispiel für ein Rechteck, das den Satzspiegel einer 288 Punkt breiten und 438 Punkt hohen Seite einschließt (für deren Satz mit dem Parameter `MON` ein Freiraum von 6 Punkt vor dem Satzspiegel angegeben wurde):

```
LINIEN = [6*24+294*462]
```

## Angaben zu RAHMEN

Die Spezifikation `RAHMEN` kann weggelassen werden, wenn auf Einzelblätter ausgegeben wird (z. B. auf Laserdruckern). Es wird dann eine für das gewählte Ausgabeformat ausreichende `BoundingBox` erzeugt.

In allen anderen Fällen sind Angaben zu `RAHMEN` sinnvoll. Sie werden zur Berechnung der PostScript `BoundingBox` herangezogen. Diese muss alle Information einschließen, die bei der Belichtung mit ausgegeben werden soll. Die Voreinstellungen bzw. eigene Angaben zur Spezifikation `EINRUECKEN` müssen dabei berücksichtigt werden.

Werden die Angaben zu `RAHMEN` in mm gemacht, oder wird zu `OPTIONEN` einer der Werte `A4U` oder `A3U` angegeben, so wird die `BoundingBox` als »HiResBoundingBox« ausgegeben; außerdem wird die Ausgabe samt `BoundingBox` nach unten verschoben. Die `BoundingBox` beginnt dann oben an der Schriftgrundlinie des (nicht durch `EINRUECKEN` verschobenen) Spaltenkopftextes, die um `y2` Punkt bzw. mm oberhalb des Nullpunkts des PostScript-Koordinatensystems zu liegen kommt, falls sie nicht durch `EINRUECKEN` verschoben wird; `y1` gibt den vertikalen Abstand der Unterkante der `BoundingBox` vom Nullpunkt des PostScript-Koordinatensystems an.

Wenn weder mm-Angaben zu `RAHMEN` gemacht noch `A4U` oder `A3U` zu `OPTIONEN` angegeben werden, so werden Ausgabe und `BoundingBox` nicht nach unten verschoben; `*PSAUS` nimmt dann an, dass auf Laserdruckern ausgegeben werden soll, und rückt zusätzlich zu den zu `EINRUECKEN` angegebenen Werten jeweils 10 Punkt vom linken und vom oberen Papierrand ein. Auch die `BoundingBox` wird um diese 10 Punkt verschoben. Bei einer nicht nach unten verschobenen `BoundingBox` gibt `y1` den Abstand ihrer Oberkante von der Schriftgrundlinie des Spaltenkopftextes (= 24 Punkt oberhalb der ersten Zeile) an.

Ein Rahmen für das kleinste Rechteck, das alle zur Seite gehörende Information aufnimmt, kann also wie folgt berechnet werden (`m` und `n` sind die zur Spezifikation `EINRUECKEN` gemachten Angaben bzw. Voreinstellungen):

	X-Wert	Y-Wert
<code>x1*y1</code>	<code>m</code>	<code>n</code>
<code>x2*y2</code>	<code>m</code> <code>+ NBREIT</code>	<code>n</code> <code>+ 24 + NHOCH</code> (aus Para-

+ NFREI0		meter HOE)
+ NFREIn	+ 2 * NGRADS	
(aus	+ 1 * NDURCHS	
Parameter MON)	+ 1/4*NGRADT	(für Unterlängen der letzten Zeile, wenn Seitennummer oben)
	+ 2 * NGRADK	(falls
	+ 1 * NDURCHK	Seitennummer
	+ 1 * NDURCHS	unten)

Soll der Rahmen das beschnittene Seitenformat des gedruckten Buches angeben, so sind für  $x_1*y_1$  die Werte  $0*0$  anzugeben; die Seiten müssen durch Angaben zu EINRUECKEN innerhalb dieses Rahmens entsprechend positioniert werden.

Die Korrektheit der Angaben zum Rahmen kann durch eine Probeausgabe nachgeprüft werden, indem vor die erste zu RAHMEN angegebene Zahl ein Doppelpunkt geschrieben wird; es wird dann ein Linien-Rechteck mit den gleichen Werten mit ausgegeben.

Bei Ausgabe im Querformat (TYP=PS-Q1, =PS-Q2 etc.) ist z. Zt. keine Angabe zu RAHMEN möglich.

Beispiel für eine BoundingBox, die außerhalb des Satzspiegels stehende Seitenzahlen, Marginalien, Kolumnentitel etc. ausklammert und lediglich einen 6 Punkt breiten Rahmen um den Satzspiegel einer 288 Punkt breiten und 438 Punkt hohen Seite einschließt (für deren Satz mit dem Parameter MON ein Freiraum von 12 Punkt links und rechts vom Satzspiegel angegeben war):

EINRUECKEN = -6+-18, OPTIONEN = A4U, RAHMEN = 0\*0+300\*450

(Der Satzspiegel endet unten mit der Schriftgrundlinie der letzten Zeile; Unterlängen und Diacritica unter den Buchstaben ragen deshalb bei diesen Angaben in den 6-Punkt-Rahmen hinein).

## Parameter

### Auswählen der Fehlerkommentare

Die Auswahlparameter sind die gleichen wie beim Standard-Makro \*FPROT; eine ausführlichere Beschreibung findet sich dort.

Parameterzeilen, die mit drei Spatien beginnen, gelten als Kommentar.

Sollen im Fehlerprotokoll alle vom Satzprogramm erzeugten Fehlerkommentare berücksichtigt werden, so braucht keiner dieser Parameter angegeben zu werden.

### Fehlerkommentare zur Fußnotennummerierung

**FN-** (keine Angabe im Informationsfeld) Fehlerkommentare, die Sprünge in der Fußnotennummerierung betreffen, sollen übergangen werden.

### Fehlerkommentare zur Spaltenbreite

**SP-** Grenzen für die Spaltenbreiten [ 1 ]

MIN	Untergrenze für Spaltenbreiten, bis zu der ein entsprechender Fehlerkommentar übernommen werden soll.
MAX	Obergrenze für Spaltenbreiten, ab der ein entsprechender Fehlerkommentar übernommen werden soll.

### Übrige Fehlerkommentare

**K-** Zeichenfolgen, die in Fehlerkommentaren enthalten sein müssen, die nicht übernommen werden sollen. [ IX ]

### Auswahl nur über Zeichenfolgen

**K+** Zeichenfolgen, die in Fehlerkommentaren enthalten sein müssen, damit diese übernommen werden. [ IX ]

### Verändern der ausgewählten Fehlerkommentare

**XXP** Zeichenfolgenpaare (und Ausnahmezeichenfolgen), deren jeweils erste Zeichenfolge durch die jeweils zweite Zeichenfolge in den ausgewählten Fehlerkommentaren ersetzt werden soll. [ X ]

## Seiten aus PostScript-Datei auswählen

#\*PSAUSWAHL

### Makro:

#\*PSAUSWAHL

QUELLE	=	datei	Name der PostScript-Datei, aus der Seiten bzw. Bereiche ausgewählt werden sollen.
ZIEL	=	datei	Name einer Fremd-Datei (zweckmäßigerweise mit der extension .ps), in die die PostScript-Daten der ausgewählten Seiten geschrieben werden sollen.
LOESCHEN	=	-	* Enthält die ZIEL-Datei schon Daten, so wird das Makro abgebrochen.
	=	+	Daten in der ZIEL-Datei zuerst löschen.
NUMMERN	=	*	Die Nummern der auszuwählenden Seiten folgen, durch »*EOF« abgeschlossen, auf den Makroaufruf.
	=	datei	Name einer Datei, die die Nummern der auszuwählenden Seiten enthält.

### Leistung

Mit diesem Makro können aus einer von \*PSAUS erzeugten PostScript-Datei einzelne Seiten oder Bereiche ausgewählt und zusammen mit dem notwendigen Vorspann in eine neue PostScript-Datei geschrieben werden.

### Angabe zu NUMMERN

Die Nummern können jeweils einzeln in je einer Zeile oder, durch Komma oder Blank getrennt, zu mehreren pro Zeile angegeben werden. Statt einzelner Seitennummern können auch Bereiche in der Form  $n_1-n_2$  angegeben werden.



# Dicktenliste für PostScript-Font drucken

#\*PSDICKTEN

## Makro:

#\*PSDICKTEN

NUMMER	= n	Nummer, die als Fontnummer in die Überschrift der erzeugten Liste eingesetzt werden soll.
NAME	= name	Name des PostScript-Fonts, dessen Zeichenvorrat und Dickenwerte ausgedruckt werden sollen. Groß- und Kleinschreibung müssen beachtet werden.
GERAET	= -STD-	Die System-Variable TUSTEP_PRN enthält den Namen des Druckers, an den die PostScript-Befehle zum Erzeugen der Liste geschickt werden sollen; falls diese System-Variable nicht definiert ist, wird der Druckername vom Betriebssystem erfragt.
	= name	Name des Druckers, auf dem ausgegeben werden soll, oder Name einer System-Variablen, die den Namen dieses Druckers enthält.
	= -	* Die PostScript-Befehle zum Erzeugen der Liste sollen nicht direkt auf einen Drucker, sondern in die zu DATEI angegebene Datei ausgegeben werden.
DATEI	= -	* Keine Ausgabe auf Datei.
	=	Wird zu GERAET ein Name angegeben, so darf zu DATEI keine Angabe außer »-« gemacht werden.
	= datei	Name der Fremd-Datei, in die die PostScript-Befehle zum Erzeugen der Liste ausgegeben werden sollen.
MODUS	= oct	* Die Adressen der Zeichen im Encoding Vector sollen oktal ausgegeben werden.
	= dec	Die Adressen der Zeichen im Encoding Vector sollen dezimal ausgegeben werden.
	= hex	Die Adressen der Zeichen im Encoding Vector sollen hexadezimal ausgegeben werden.
	= oct2	wie oct, aber Ausgabe auf A3-Format mit größerem Zeilenabstand
OPTIONEN	= -	* Zeichen ohne Zusatz ausgeben
	= RAHMEN	Zeichen mit Rahmen (feine rote Linie am linken Rand, auf der Schriftgrundlinie und am rechten

Rand) ausgeben, mit dessen Hilfe Stand und Dichte auch optisch ablesbar sind.

## ZEICHENSATZ

(Angabe nur für nach Type-1 umgewandelte OpenType-Fonts notwendig)

= -	* Standard-Zeichensatz (lateinisch)
= KAP	Kapitälchen
= GR	Griechischer Zeichensatz
= HB	Hebräischer Zeichensatz
= CY	Kyrillischer Zeichensatz, Font-Encoding
= CY-T	Kyrillischer Zeichensatz, TUSTEP-Encoding

## Leistung

Mit diesem Makro können der Zeichenvorrat eines PostScript-Fonts, die zugehörigen Adressen im zugrunde gelegten Encoding Vector und die Dickenwerte für diese Zeichen auf einem PostScript-Drucker bzw. -Belichter ausgegeben werden. Zeichen, denen im zugrunde gelegten Encoding Vector keine Adresse zugeordnet ist, werden nicht aufgelistet (dies kann mit dem Makro \*PSGLYPHS erfolgen). Der Font muss auf dem Gerät, auf dem die Liste ausgedruckt werden soll, installiert sein oder in die erzeugte PS-Datei eingebunden werden. Die Zeichen und die Dickenwerte werden erst bei der Ausgabe auf diesem Gerät generiert.

Die zu NUMMER angegebene Nummer dient nur der Kennzeichnung des Fonts. Sie wird zusammen mit dem Namen des PostScript-Fonts in die Überschrift der Liste übernommen und sollte deshalb mit der Nummer übereinstimmen, mit der der Font unter TUSTEP angesprochen wird.

# PostScript-Datei ausdrucken

#\*PSDR

## Makro:

#\*PSDR

DATEI	= datei	Name der PostScript-Datei, die auf einem PostScript-Drucker ausgegeben werden soll.
GERAET	= -STD-	* Die System-Variable TUSTEP_PRN enthält den Namen des Druckers, auf dem ausgegeben werden soll; falls diese System-Variable nicht definiert ist, wird der Druckername vom Betriebssystem erfragt.
	= name	Name des Druckers, auf dem ausgegeben werden soll, oder Name einer System-Variablen, die den Namen dieses Druckers enthält.
ANZAHL	= n	Angabe, wie oft die zu DATEI angegebene Datei ausgedruckt werden soll.
	= 1	* Die Datei soll einmal ausgedruckt werden.
OPTIONEN	= -STD-	* Normalfall; keine Änderung der Druckereinstellungen.
	= SMPLX	Papier einseitig bedrucken.
	= D_LANG	Papier doppelseitig bedrucken; Blättern an der langen Papierkante (für Ausgabe im Hochformat).
	= D_KURZ	Papier doppelseitig bedrucken; Blättern an der kurzen Papierkante (für Ausgabe im Querformat).
ZUSATZ	= ...	Name einer System-Variablen, die zusätzliche Angaben zum Ausgabegerät für das Betriebssystem enthält (vgl. System-Variable TUSTEP_LPR). Welche Angaben sinnvoll oder im Einzelfall notwendig sind, ist beim jeweiligen Rechenzentrum bzw. beim Systemverwalter zu erfragen.

## Leistung

Mit diesem Makro kann eine mit dem Makro \*PSAUS erzeugte PostScript-Datei auf einem PostScript-Drucker ausgegeben werden. Dabei kann eine beliebige Anzahl von Exemplaren und, falls der Drucker dies vorsieht, doppelseitiger Druck verlangt werden.

**Hinweis**

Eine von »1« verschiedene Angabe zur Spezifikation ANZAHL bewirkt, dass die auszugebende Datei entsprechend oft hintereinander kopiert und das Ergebnis an den angegebenen Drucker geschickt wird.

Die Angaben SMPLX, D\_LANG und D\_KURZ sind nur bei Ausgabe auf Druckern mit Duplex-Einrichtung sinnvoll. Außerdem setzen sie PostScript Level 2 voraus.

# Dicktenwerte von PostScript-Fonts bereitstellen

#\*PSFONT

## Makro:

#\*PSFONT

QUELLE	=	datei	Name der Datei, die den AFM-File (Adobe Font Metrics) einer PostScript-Schrift enthält, deren Dicktentabelle für das Satzprogramm aufbereitet werden soll.
ZIEL	=	datei	Name der Datei, in die die Dicktentabelle ausgegeben werden soll.
LOESCHEN	=	+	Daten in der ZIEL- und in der PROTOKOLL-Datei zu erst löschen.
	=	-STD-	Enthält die ZIEL-Datei schon eine Dicktentabelle mit der gleichen Nummer wie die neue Schriftnummer, so soll sie durch die neue Dicktentabelle ersetzt werden.
	=	-	* Enthält die ZIEL-Datei schon eine Dicktentabelle mit der gleichen Nummer wie die neue Schriftnummer, so wird nachgefragt, ob die Verarbeitung abgebrochen oder diese Dicktentabelle ersetzt werden soll. Daten in der PROTOKOLL-Datei sollen nicht gelöscht werden.
NUMMER	=	n	Schriftnummer, die die Dicktentabelle in der ZIEL-Datei erhalten soll (unter der die Schrift beim Aufruf des Satzprogramms mit dem Parameter SCH angegeben werden soll).
PROTOKOLL	=	datei	Name der Datei, in die eine Liste ausgegeben werden soll, die die TUSTEP-Codierung, die PostScript-Adresse und die Dicke der Zeichen enthält, die zu der Schrift mit der angegebenen Nummer gehören.
	=	-STD-	Die Liste soll in die Standard-Protokoll-Datei ausgegeben werden.
	=	-	* Keine Liste ausgeben.
MEDLIG	=	datei	Name der Datei, die den AFM-File einer Schrift mit den Mediaevalziffern und den f-Ligaturen zur Schrift mit der angegebenen Schriftnummer enthält.
	=	-	* Zur angegebenen Schrift ist kein AFM-File für Mediaevalziffern und f-Ligaturen vorhanden.

MODUS	= -STD-	* Normalfall: Erzeugen von Dicktentabellen für eine Schrift, die mit dem Parameter SCH im Satzprogramm angegeben werden soll. Dem AFM-File liegt der Adobe Standard Encoding Vector (für Kapitälchenschriften: der Adobe Expert Encoding Vector) zugrunde. Diese Schrift kann gleichzeitig als Sonderzeichen-Font benutzt werden.
	= +	Wie -STD-; zusätzlich werden die Namen der Zeichen ab 128(dez) in die Namen des Standard-Encoding-Vector verwandelt (der vom Makro *PSFONT z. B. für Akzente vorausgesetzt wird).
	= SEV	Dem vorliegenden AFM-File für Kapitälchenschrift liegt nicht der vom Makro *PSAUS für Kapitälchenschriften erwartete Expert Encoding Vector, sondern der Standard Encoding Vector zugrunde.
	= CYR	Dem vorliegenden AFM-File liegt der Encoding Vector für kyrillische Fonts zugrunde.
	= GR	Dem vorliegenden AFM-File liegt der Encoding Vector für griechische Fonts zugrunde.
	= HEB	Dem vorliegenden AFM-File liegt der Encoding Vector für hebräische Fonts zugrunde.
	= -	Erzeugen nur der Tabellen, die für die Benutzung als Sonderzeichen-Font (Parameter BIL bzw. Anweisung &! (#mmmmmm/nnn) im Satzprogramm) notwendig sind.
ZUSATZ	= -	* Normalfall, kein Zusatz.
	= datei	Name einer Datei mit zusätzlichen Zeilen, die in die Daten aus dem AFM-File eingefügt werden sollen.

## Leistung

Mit diesem Makro können AFM (Adobe Font Metrics)-Files von PostScript-Schriften in Dicktentabellen für das TUSTEP-Satzprogramm umgewandelt und, mit einer Schriftnummer versehen, auf einer Datei abgelegt werden. Wenn diese Schriften verwendet werden sollen, muss diese Datei beim Aufruf des Kommandos #SATZ und des Makros \*PSAUS jeweils zur Spezifikation SCHRIFTEN angegeben werden. Beim Satz wird die Schriftnummer zum Parameter SCH angegeben.

## Regeln für die Vergabe der Schriftnummern

Das Kommando #SATZ erwartet für Schriften, deren Dicktentabellen zur Spezifikation SCHRIFTEN angegeben werden, Schriftnummern zwischen 40001 und 49999. Die zu einer Familie gehörenden Schriften sind daran zu erkennen, dass die Schrift-

nummern mit Ausnahme der Einerstelle identisch sind. Es können derzeit also maximal 999 verschiedene Schriftfamilien angegeben werden.

Innerhalb einer Schriftfamilie gibt die Einerstelle der Schriftnummer den Schriftschnitt an:

- 1 = normal
- 2 = kursiv
- 3 = halbfett
- 4 = Kapitälchen
- 5 = halbfett kursiv
- 6 = fett (falls zusätzlich zu halbfett vorhanden)
- 7 = schmal
- 8 = schmal (halb)fett
- 9 = fett kursiv (zusätzlich zu halbfett kursiv)

Die Nummern zwischen 45000 und 45049 sind für hebräische Fonts reserviert.

Die Nummern zwischen 45100 und 45299 sind für griechische Fonts reserviert.

Die Nummern zwischen 46100 und 46299 sind für kyrillische Fonts reserviert.

Die Nummern zwischen 48000 und 48999 sind für Fraktur-Fonts reserviert (derzeit belegt: 48000 bis 48099 für Fonts der Firma Delbanco).

## Vorbereiten einer AFM-Datei für \*PSFONT

#\*PSFONTVOR

### Makro:

#\*PSFONTVOR

QUELLE	= datei	Name der AFM-Datei des aus einem OpenType-Font erzeugten Type-1-Fonts.
ZIEL	= datei	Name der Datei, in der die Zeichennamen und Dickenwerte der zu ZEICHENSATZ gehörenden Zeichen gesammelt und mit dem für TUSTEP gültigen encoding gespeichert werden sollen.
LOESCHEN	= +	Daten in der ZIEL-Datei zuerst löschen.
ZEICHENSATZ	= -	* Zeichen für lateinischen Font auswählen.
	= KAP	Zeichen für lateinischen Kapitalchen-Font auswählen.
	= GR	Zeichensatz für das Griechische (polytonic, Altgriechisch) auswählen.
	= CY	Zeichensatz für das Kyrillische (Russische) auswählen.
	= HEB	Zeichensatz für das Hebräische auswählen.

### Leistung

Das TUSTEP-Satzprogramm arbeitet mit Fonts, die den Konventionen von Type-1-Fonts entsprechen. OpenType-Fonts müssen vor dem Satz entsprechend aufbereitet werden. Dazu werden sie zunächst mit einem geeigneten Tool (Font-Editor) in Type-1-Fonts umgewandelt. Aus der dabei erzeugten AFM-Datei können mit \*PSFONTVOR die Zeichen, die zu dem zu ZEICHENSATZ angegebenen Zeichensatz gehören, ausgewählt und, mit dem von TUSTEP erwarteten encoding versehen, in eine Datei geschrieben werden, die dann mit \*PSFONT in eine Dicktentabelle für das TUSTEP-Satzprogramm umgewandelt werden kann.



# Zeichenliste für PostScript-Font drucken

**#\*PSGLYPHS**

## Makro:

#\*PSGLYPHS

NUMMER	= n	Nummer, die als Fontnummer in die Überschrift der erzeugten Liste eingesetzt werden soll.
QUELLE	= name	Name (einschließlich Pfadangabe) der AFM-Datei des PostScript-Fonts, dessen Zeichenvorrat und Dicktenwerte aufgelistet werden sollen. Groß- und Kleinschreibung müssen beachtet werden.
GERAET	= -STD-	Die System-Variable TUSTEP_PRN enthält den Namen des Druckers, an den die PostScript-Befehle zum Erzeugen der Liste geschickt werden sollen; falls diese System-Variable nicht definiert ist, wird der Druckername vom Betriebssystem erfragt.
	= name	Name des Druckers, auf dem ausgegeben werden soll, oder Name einer System-Variablen, die den Namen dieses Druckers enthält.
	= -	* Die PostScript-Befehle zum Erzeugen der Liste sollen nicht direkt auf einen Drucker, sondern in die zu DATEI angegebene Datei ausgegeben werden.
DATEI	= -	* Keine Ausgabe auf Datei.
	=	Wird zu GERAET ein Name angegeben, so darf zu DATEI keine Angabe außer »-« gemacht werden.
	= datei	Name der Fremd-Datei, in die die PostScript-Befehle zum Erzeugen der Liste ausgegeben werden sollen. Die Zeichen und die Dicktenwerte werden erst bei der Ausgabe auf dem PostScript-Gerät generiert.
ORDNUNG	= -	* Die Zeichen werden in der Reihenfolge aufgelistet, in der sie in der AFM-Datei stehen.
	= ALPHA	Die Zeichen werden alphabetisch nach ihren Namen sortiert ausgegeben.
OPTIONEN	= -	* Zeichen ohne Zusatz ausgeben
	= RAHMEN	Zeichen mit Rahmen (feine rote Linie am linken Rand, auf der Schriftgrundlinie und am rechten Rand) ausgeben, mit dessen Hilfe Stand und Dicke auch optisch ablesbar sind.

## Leistung

Mit diesem Makro können der Zeichenvorrat eines PostScript-Fonts und die Dickenwerte für diese Zeichen auf einem PostScript-Drucker bzw. -Belichter ausgegeben werden, unabhängig davon, ob ihnen im zugrunde gelegten Encoding Vector eine Adresse zugeordnet ist. Der Font muss auf dem Gerät, auf dem die Liste ausgedruckt werden soll, installiert sein. Die Zeichen und die Dickenwerte werden erst bei der Ausgabe auf diesem Gerät generiert.

Die zu `NUMMER` angegebene Nummer dient nur der Kennzeichnung des Fonts. Sie wird zusammen mit dem Namen des PostScript-Fonts in die Überschrift der Liste übernommen und sollte deshalb mit der Nummer übereinstimmen, mit der der Font unter `TUSTEP` angesprochen wird.

# Kerning-Information von PostScript-Fonts bereitstellen

**#\*PSKERNFILE**

## Makro:

#\*PSKERNFILE

QUELLE	= datei	Name der Datei, die den AFM-File (Adobe Font Metrics) einer PostScript-Schrift enthält, aus der die Kerning-Information für das *PSAUS aufbereitet werden soll.
ZIEL	= datei	Name der Datei, in die die Kerning-Information ausgegeben werden soll.
LOESCHEN	= +	Daten in der ZIEL-Datei zuerst löschen.
	= -STD-	Enthält die ZIEL-Datei schon Information zu einem Font mit der gleichen Schriftnummer, so soll sie durch die neue Kerning-Information ersetzt werden.
	= -	* Enthält die ZIEL-Datei schon Information zu einem Font mit der gleichen Schriftnummer, so wird nachgefragt, ob die Verarbeitung abgebrochen oder diese Information ersetzt werden soll.
NUMMER	= n	Schriftnummer, die die Kerning-Information in der ZIEL-Datei erhalten soll. Sie muß identisch sein mit der Nummer, die für diese Schrift beim Aufruf von *PSFONT vergeben wurde.

## Leistung

Mit diesem Makro wird die Kerning-Information aus AFM (Adobe Font Metrics)-Files von PostScript-Schriften so aufbereitet und, mit einer Schriftnummer versehen, in einer Datei so abgelegt, dass sie beim Erstellen der PostScript-Datei mit dem Standard-Makro \*PSAUS berücksichtigt werden kann. Dazu wird diese Datei beim Aufruf des Makros \*PSAUS als zweite Datei zur Spezifikation SCHRIFTEN angegeben.

# PostScript-Dateien zusammenmontieren

#\*PSMONT

## Makro:

#\*PSMONT

QUELLE = datei1' datei2 Name der beiden PostScript-Dateien, die zusammenmontiert werden sollen. Die Namen der Dateien sind durch Apostroph zu trennen.

ZIEL = datei Name einer System-Datei, in die die zusammenmontierten PostScript-Daten ausgegeben werden sollen.

MODUS = -STD- \* Spalten mit gleicher Spaltennummer werden zu einer Spalte zusammenkopiert.

= n Die Dateien werden hintereinander kopiert. Endet die erste Datei mit einer Seite, die die gleiche Nummer hat wie die erste Seite der zweiten Datei, so werden diese Seiten zu einer Seite zusammenkopiert.

Mögliche Angaben für n:

= 11 die erste Datei endet einspaltig, die zweite Datei ist einspaltig

= 12 die erste Datei endet einspaltig, die zweite Datei ist zweisepaltig

= 21 die erste Datei endet zweisepaltig, die zweite Datei ist einspaltig

= 22 die erste Datei endet zweisepaltig, die zweite Datei ist zweisepaltig

LOESCHEN = - \* Enthält die ZIEL-Datei schon Daten, so wird das Makro abgebrochen.

= + Daten in der ZIEL-Datei zuerst löschen.

PRUEFEN = + \* Normalfall: Der Vorspann der zu montierenden PostScript-Dateien muss übereinstimmen. Ist dies nicht der Fall, wird mit FEHLERHALT abgebrochen.

= - Ergibt die Prüfung Abweichungen, so entfällt der Fehlerhalt. Die Abweichungen werden aufgelistet. Vorsicht: Die ZIEL-Datei sollte, wenn Abweichungen festgestellt wurden, nur weiterverwendet werden, wenn sicher ist, dass diese sich nicht auf die Ausgabe auswirken.

= -- Die Prüfung entfällt ganz. Vorsicht: Sollte nur angegeben werden, wenn sicher ist, dass sich eventuelle Unterschiede nicht auf die Ausgabe auswirken.

## Leistung

Mit diesem Makro können zwei mit TUSTEP erstellte PostScript-Dateien zusammenmontiert werden. Der Vorspann der PostScript-Dateien, der die Information über die verwendeten Fonts enthält, muss identisch sein: Bei den Satzprogrammläufen, mit denen diese Dateien erstellt wurden, müssen die Angaben zu den Parametern SCH und BIL sowie zu SGM und SLW jeweils identisch sein; alle in der zweiten Datei über &! (#mmmmmm/nnn) codierten Sonderzeichen müssen in der ersten Datei ebenfalls vorkommen. (Ausnahme: die zweite Datei darf weniger Angaben zu Fonts enthalten, solange die Reihenfolge der Angaben in den Parametern SCH, SGM und SLW identisch ist.)

Ist die erste zu QUELLE angegebene Datei leer, so wird die zweite dort angegebene Datei unverändert kopiert.

Die ZIEL-Datei von \*PSMONT kann wieder QUELL-Datei eines nachfolgenden Aufrufs von \*PSMONT werden. Auf diese Weise können mehr als zwei PostScript-Dateien zusammenkopiert werden.

## Mit Klammern codierte Registerinträge umwandeln

#\*REKLAM

### Makro:

#\*REKLAM

QUELLE	=	datei	Name der Datei mit dem Text, der Registerinträge in Klammern enthält.
ZIEL	=	datei	Name der Datei, in die der Text mit den umcodierten Registerinträgen ausgegeben werden soll.
MODUS	=	-	* Die Registerinträge enthalten kein Typenkennzeichen.
	=	+	Die Registerinträge enthalten als erstes Zeichen in den Klammern ein Typenkennzeichen.
LOESCHEN	=	-	* Daten in der ZIEL-Datei nicht löschen.
	=	+	Daten in der ZIEL-Datei zuerst löschen.
ABSCHNITTE	=	*	Parameter, mit denen die im Text verwendeten Kennzeichnungen der Abschnittsanfänge angegeben sind, folgen auf den Makroaufruf und sind mit *EOF abgeschlossen.
	=	-STD-	* Es wird folgender Parameter benutzt: AA           :\$:&:%{\ \0 }::&.:&!:&X:

### Leistung

Mit diesem Makro können Registerinträge, die zwischen ((...)) bzw. (((...))) im Text stehen, nach den Konventionen des Satzprogramms umcodiert werden. Dabei werden die in doppelten Klammern stehenden Einträge so codiert, dass sie im Text und im Register erscheinen, die in dreifachen Klammern stehenden Einträge so, dass sie nur im Register erscheinen.

## Erläuterungen

Für das Satzprogramm können Registerinträge zwischen den Steueranweisungen »&x« für Kommentaranfang und »&x{« für Kommentarende geschrieben und mit Typenkennzeichen für verschiedene Register versehen werden. Die zwischen diesen Steueranweisungen stehenden Zeichenfolgen werden für die Satzausgabe nicht berücksichtigt, aber in die ZIEL- und in die PROTOKOLL-Datei mit übernommen. Dies ermöglicht, aus der ZIEL-Datei Registerinträge mit der Seiten- und Zeilennummer als Referenz zu extrahieren, die sie beim automatischen Umbruch durch das Satzprogramm erhalten haben, und zwar auch dann, wenn der für einen Registereintrag geltende Wortlaut nicht mit einem zu druckenden Textwortlaut übereinstimmt.

Damit Begriffe, bei denen Text- und Registerwortlaut identisch sind, bei der Erfassung nicht doppelt (nämlich einmal für den Satz, und einmal zwischen den Steueranweisungen »&x« und »&x{« für die Register) geschrieben werden müssen, kann man diese bei der Erfassung in doppelte Klammern einschließen (evtl. zusätzlich mit Typenkennzeichen) und die für den genannten Zweck notwendige Verdoppelung vom Makro \*REKLAM ausführen lassen. Für Registerinträge, deren Wortlaut im Text nicht vorkommt, können drei Klammern benutzt werden; sie werden vom Makro \*REKLAM nicht verdoppelt, sondern lediglich zwischen »&x« und »&x{« geschrieben. Folgen auf drei öffnende Klammern nur zwei schließende Klammern, so werden diese zwei wie drei schließende Klammern behandelt.

### Beispiel

#### Der Eingabetext

```
Goethe ((nGoethe, Johann Wolfgang v.)) wurde am
28. August 1749 zu ((oFrankfurt am Main))
geboren und starb am 22. März 1832 in ((oWeimar));
er hatte u. a. an der ((oLeipzig))er Universität
studiert ...
```

wird mit dem Makro \*REKLAM bei MODUS=+ (mit den Typenkennzeichen »o« für »Orte« und »n« für »Namen«) so umgewandelt:

```
Goethe &xnGoethe, Johann Wolfgang v.&x{ wurde am
28. August 1749 zu Frankfurt am Main&x{oFrankfurt am Main&x{
geboren und starb am 22. März 1832 in Weimar&x{oWeimar&x};
er hatte u. a. an der Leipzig&x{oLeipzig&x{er Universität
studiert ...
```

Der für die Erstellung der AUSGABE-Datei des Kommandos #SATZ berücksichtigte Text ohne Registerinträge lautet dann:

```
Goethe wurde am
28. August 1749 zu Frankfurt am Main
geboren und starb am 22. März 1832 in Weimar;
er hatte u. a. an der Leipziger Universität
studiert ...
```

## Parameter ABSCHNITTE

Das Makro geht davon aus, dass sich ein Registereintrag über mehrere Zeilen erstrecken kann. Außerdem wird versucht, das Fehlen von Kennzeichnungen für den Anfang bzw. das Ende von Registereinträgen festzustellen und den Umfang der durch solche Fehler verursachten falschen Aufteilungen möglichst zu begrenzen. Zu diesem Zweck erwartet das Programm, dass ihm die wichtigsten Steueranweisungen, mit deren Hilfe der Text in Abschnitte eingeteilt wurde, mitgeteilt werden, damit es abschnittsweise nach Anfangs- und Ende-Codierungen für die Registereinträge suchen kann. Dies geschieht über die folgenden Parameter:

- AA** Zeichenfolgen, die am Anfang eines Eingabesatzes den Anfang neuer Abschnitte (einschl. Zwischenüberschriften) eindeutig kennzeichnen.  
[ IXa ]
- AE** Zeichenfolgen, die am Ende eines Eingabesatzes das Ende von Abschnitten (einschl. Zwischenüberschriften) eindeutig kennzeichnen.  
[ IXb ]

Die Angaben in [ ] geben die Parameterarten an; diese sind in der Beschreibung »TUSTEP-Grundlagen« im Kapitel »Parameter« definiert.



# Silbentrennungs-Archiv erstellen

#\*SILARCH

## Makro:

#\*SILARCH

QUELLE	= datei	Name der Datei, die die zu korrigierende und zu archivierende Silbentrennungsliste enthält.
ZIEL	= datei	Name der Datei, in die die korrigierte Silbentrennungsliste ausgegeben werden soll
LOESCHEN	= -	* Daten in der ZIEL- und in der PROTOKOLL-Datei nicht löschen; die neu zu archivierende Silbentrennungen in die bereits in der ZIEL-Datei vorhandenen einmischen.
	= +	Daten in der ZIEL- und in der PROTOKOLL-Datei zuerst löschen.
PROTOKOLL	= datei	Name der Datei, in die das Korrekturprotokoll ausgegeben werden soll.
	= -STD-	* Das Korrekturprotokoll soll in die Standard-Protokoll-Datei ausgegeben werden.
KORREKTUR	= -	* Keine Korrekturanweisungen zu den Silbentrennungen.
	= datei	Name der Datei mit Korrekturanweisungen zu den in QUELLE enthaltenen Silbentrennungen.

## Leistung

Mit diesem Makro kann eine (mit dem Makro \*SILLIST erzeugte) alphabetisch sortierte Silbentrennungsliste korrigiert und archiviert werden. Diese korrigierte Silbentrennungsliste kann beim Erstellen weiterer Silbtrennungslisten mit dem Makro \*SILLIST benutzt werden, um deren Umfang zu reduzieren.

Identische Wörter aus der QUELL-Datei, die an unterschiedlichen Stellen getrennt sind, werden als ein einziger Eintrag mit allen vorkommenden Trennstellen in die ZIEL-Datei geschrieben. Ausgenommen sind in der QUELL-Datei enthaltene getrennte Kuppelwörter. Sie werden mit jeder unterschiedlichen Trennstelle unverändert als eigene Einträge in die ZIEL-Datei geschrieben.

## Die Korrekturanweisungen

Die Korrektur erfolgt über Korrekturanweisungen, die ein Verschieben einer Trennstelle nach links bzw. rechts oder ein Löschen eines Eintrags aus der Silbentrennungsliste angeben.

Die Korrekturanweisungen haben die Form:

- s . z +n**      Verschieben der Trennstelle des Wortes, das in der Silbentrennungsliste unter Seiten-Zeilen-Nummer *s . z* aufgeführt ist, um *n* Stellen nach rechts
- s . z -n**      Verschieben der Trennstelle des Wortes, das in der Silbtrennungsliste unter Seiten-Zeilen-Nummer *s . z* aufgeführt ist, um *n* Stellen nach links
- s . z -**        Das Wort, das in der Silbtrennungsliste unter Seiten-Zeilen-Nummer *s . z* aufgeführt ist, nicht archivieren

Das Makro kopiert die in der QUELL-Datei enthaltene Silbentrennungsliste ohne die zugehörigen Referenzen in die ZIEL-Datei. Ist zu einem Eintrag eine Korrektur vorhanden, so wird die Trennung in der angegebenen Weise verschoben bzw. der Eintrag übergangen.

# Silbentrennungen korrigieren

#\*SILKOR

## Makro:

#*SILKOR		
QUELLE	= datei	Name der Datei mit dem Text, in dem die Silbentrennungen korrigiert werden müssen.
ZIEL	= datei	Name der Datei, in die der Text mit korrigierten Silbentrennungen ausgegeben werden soll.
LOESCHEN	= -	* Daten in der ZIEL- und in der PROTOKOLL-Datei nicht löschen.
	= +	Daten in der ZIEL- und in der PROTOKOLL-Datei zuerst löschen.
TRENNUNGEN	= datei	Name der Datei mit der Silbentrennungsliste.
PROTOKOLL	= datei	Name der Datei, in die das Korrekturprotokoll ausgegeben werden soll.
	= -STD-	* Das Korrekturprotokoll soll in die Standard-Protokoll-Datei ausgegeben werden.
KORREKTUR	= datei	Name der Datei mit den Korrekturanweisungen.

## Leistung

Mit diesem Makro können in einem Text enthaltene Silbentrennungen teilautomatisch korrigiert werden. Die Korrektur erfolgt über Korrekturanweisungen, die sich auf eine mit dem Makro \*SILLIST von diesem Text erstellte Silbentrennungsliste beziehen.

Mit den Korrekturanweisungen kann eine Verschiebung der Trennstelle nach links bzw. rechts oder ein Verbot der Trennung angegeben werden.

Die im Makro \*SILKOR angegebene QUELL-Datei muss identisch sein mit der QUELL-Datei, die beim Erstellen der Silbentrennungsliste mit dem Makro \*SILLIST benutzt wurde.

Die im Makro \*SILKOR zur Spezifikation TRENNUNGEN angegebene Datei muss identisch sein mit der Datei, die beim Erstellen der Silbentrennungsliste als ZIEL-Datei benutzt wurde.

Enthält die erste Zeile in der zu TRENNUNGEN angegebenen Datei den Namen und das Änderungsdatum der QUELL-Datei, so wird geprüft, ob es sich bei der zu QUELLE angegebenen Datei um diese Datei handelt. Wenn nicht, wird das Makro abgebrochen.

## Die Korrekturanweisungen

Zum Zweck der teilautomatischen Korrektur von Silbentrennfehlern kann mit dem Makro \*SILLIST eine korrekturfähige alphabetisch sortierte Liste der in einem Text enthaltenen Silbentrennungen erstellt werden. Zu jedem getrennten Wort sind in dieser Liste die Seiten- und Zeilennummern der Textstellen angegeben, an denen das Wort in der angegebenen Weise getrennt wurde. Jeder Eintrag in dieser Liste hat selbst eine eindeutige Seiten-Zeilen-Nummer, die in der Korrekturanweisung angegeben werden muss. Einträge, die von \*SILLIST eine Unterscheidungsnummer bekommen haben (das sind mehrfach getrennte Wörter), können nicht in Korrekturanweisungen berücksichtigt werden.

Die Korrekturanweisungen haben die Form:

- s . z +n** Verschieben der Trennstelle des Wortes, das in der Silbentrennungsliste unter Seiten-Zeilen-Nummer *s . z* aufgeführt ist, um *n* Stellen nach rechts. Wenn die Trennstelle um mehr als 2 Stellen verschoben wird, wird an der ursprünglichen Trennstelle ein Trennverbot eingetragen.
- s . z -n** Verschieben der Trennstelle des Wortes, das in der Silbentrennungsliste unter Seiten-Zeilen-Nummer *s . z* aufgeführt ist, um *n* Stellen nach links. Wenn die Trennstelle um mehr als 2 Stellen verschoben wird, wird an der ursprünglichen Trennstelle ein Trennverbot eingetragen.
- s . z -** Trennverbot an der Stelle eintragen, an der das in der Silbentrennungsliste unter Seiten-Zeilen-Nummer *s . z* aufgeführte Wort getrennt ist.

# Silbentrennungsliste erstellen

#\*SILLIST

## Makro:

#\*SILLIST

QUELLE	= datei	Name der Datei mit dem Text, für den eine Liste der darin enthaltenen Silbentrennungen erstellt werden soll. Dies ist in der Regel die ZIEL-Datei eines Satzprogrammlaufs.
ZIEL	= datei	Name der Datei, in die die (korrekturfähige) Liste der Silbentrennungen ausgegeben werden soll.
MODUS	= +	* Satzzeichen, Steueranweisungen und Spitzklammermakros, die im Text am Anfang oder am Ende eines getrennten Wortes stehen, werden in der Silbentrennungsliste mit ausgegeben.
	= -	Satzzeichen, Steueranweisungen und Spitzklammermakros, die im Text am Anfang oder am Ende eines getrennten Wortes stehen, werden nicht mit in die Silbentrennungsliste ausgegeben.
	= --	wie MODUS=-, jedoch gelten auch Spitzklammern als Satzzeichen, nicht als Begrenzungszeichen von Spitzklammermakros (Tags).
	= A+	Wie MODUS=+, jedoch werden auch die Trennungen mit ausgegeben, die an (durch »\« markierten) Kann-Trennstellen durchgeführt wurden.
	= A-	Wie MODUS=-, jedoch werden auch die Trennungen mit ausgegeben, die an (durch »\« markierten) Kann-Trennstellen durchgeführt wurden.
	= A--	Wie MODUS=--, jedoch werden auch die Trennungen an durch »\« markierten Stellen mit ausgegeben.
	= A	Gleichbedeutend mit MODUS=A+.
	= P	Die zu QUELLE angegebene Datei ist die PROTOKOLL-Datei eines FORMATIERE-Laufs. Diese Angabe ist nur sinnvoll, wenn der Text mit dem Kommando #FORMATIERE einspaltig aufbereitet wurde.
LOESCHEN	= -	* Daten in der ZIEL- und in der PROTOKOLL-Datei nicht löschen (nur bei leerer ZIEL-Datei möglich).

	= +	Daten in der ZIEL- und in der PROTOKOLL-Datei zuerst löschen.
PROTOKOLL	= datei	Name der Datei, in die die zum Druck aufbereitete Silbentrennungsliste ausgegeben werden soll.
	= -STD-	* Die zum Druck aufbereitete Silbentrennungsliste soll in die Standard-Protokoll-Datei ausgegeben werden.
SPALTEN	= 1	Die Silbentrennungsliste soll einspaltig zum Druck auf schmalem Papier (Format A4) aufbereitet werden.
	= 2	* Die Silbentrennungsliste soll zweispaltig zum Druck auf breitem Papier (Zeilendrucker-Format) aufbereitet werden.
TRENNUNGEN	= datei	Name der Datei mit einer Liste von Silbentrennungen, die nicht in die zu erstellende Silbentrennungsliste aufgenommen werden sollen.
	= -	* Es sollen alle Silbentrennungen aufgenommen werden.
ZUSATZ	= -	Keine weiteren Angaben
	= *	Weitere Zeichenfolgen, die außer den durch MODUS=- bzw. MODUS=-- am Anfang und am Ende des getrennten Wortes unterdrückten Zeichenfolgen entfernt werden sollen, folgen in je einer Zeile, durch *eof abgeschlossen, auf den Makroaufruf.

## Leistung

Mit diesem Makro können die getrennten Wörter aus dem Text extrahiert und in Form einer alphabetisch sortierten Liste ausgegeben werden. Eine solche Liste ermöglicht es, jedes getrennte Wort nur einmal kontrollieren zu müssen, unabhängig davon, wie oft es im Text mit der gleichen Trennung vorkommt.

Zu jedem getrennten Wort werden die zugehörige Seiten-Zeilen-Nummern als Referenzen mit ausgegeben. Diese Silbentrennungsliste in der ZIEL-Datei dient als Grundlage zur halbautomatischen Korrektur der Trennfehler mit Hilfe des Makros \*SILKOR.

Mehrfach getrennte Wörter werden als ganze Wörter in die Liste aufgenommen; sie erhalten eine Seiten-Zeilen-Unterscheidungsnummer als Referenz. Solche Trennungen können nicht mit Hilfe des Makros \*SILKOR korrigiert werden.

Trennungen an Stellen, die durch »\« markiert sind, werden bei der Erstellung der Silbentrennungsliste übergangen, wenn dies nicht mit MODUS=A+, A- oder A ausdrücklich verhindert wird. Bei MODUS=P werden solche Trennungen immer mit ausgegeben, da sie von den anderen nicht unterschieden werden können.

Darüber hinaus kann zur Spezifikation TRENNUNGEN eine Datei angegeben werden, die Wörter mit richtigen Trennungen enthält. Wörter, die so getrennt sind wie in

dieser Datei, werden ebenfalls nicht in die zu erstellende Silbentrennungsliste aufgenommen. Eine Datei mit richtigen Trennungen kann mit Hilfe des Makros \*SILARCH erstellt werden. Für identische Wörter mit unterschiedlichen Trennstellen wird dabei für jede Trennstelle jeweils eigener Eintrag in die Datei geschrieben. Das Makro \*SILLIST berücksichtigt jedoch auch Einträge, bei denen alle möglichen Trennstellen in einem einzigen Eintrag zum betreffenden Wort enthalten sind.

In der zu TRENNUNGEN angegebenen Datei können auch Einträge mit mehr als einer Trennstelle stehen (das Makro \*SILARCH führt identische Wörter mit mehr als einer Trennstelle als separaten Eintrag mit der jeweiligen Trennstelle auf).

In die erste Zeile der ZIEL-Datei wird der volle Name der QUELL-Datei und Datum + Uhrzeit ihrer letzten Änderung geschrieben. Diese Angaben werden bei einem späteren Aufruf von \*SILKOR zur Prüfung auf Identität der QUELL-Dateien benutzt.

## Zur automatischen Silbentrennung

Das Satzprogramm führt, wenn die Silbentrennung nicht über Parameterangaben oder Steueranweisungen unterbunden wird, eine automatische Silbentrennung durch, falls Zeileneinteilung und ggf. Randausgleich dies erfordern. Dabei werden die Regeln der Silbentrennung für die deutsche Sprache zugrunde gelegt.

Keine automatisch durchgeführte Silbentrennung arbeitet fehlerfrei. Selbst umfangreiche Ausnahmelisten würden dem nicht abhelfen, da es viele Wörter gibt, die je nach ihrem Kontext anders zu trennen sind (Beispiel: »Spie-lende Kinder«, aber: »kurz vor Spiel-ende«; »Stau-becken« und »Staub-ecken«). Vor allem zusammengesetzte Wörter bieten eine schier unerschöpfliche Quelle von Fehlermöglichkeiten.

Für deutsche Texte mit einem nicht ungewöhnlich hohen Anteil an zusammengesetzten Wörtern muss der TUSTEP-Benutzer erfahrungsgemäß mit 0,5 Prozent fehlerhaften Trennungen (und einer höheren Rate sogenannter »unschöner« Trennungen wie »Versauf-bau«) rechnen.

Obwohl das verwendete Trennprogramm die für das Deutsche geltenden Trennregeln voraussetzt, ist es auch für romanische Sprachen noch brauchbar; die Fehlerrate liegt dann bei etwa 5 Prozent. Für englisch-sprachige Texte sind diese Trennregeln ungeeignet; hier empfiehlt es sich, die Zahl der Trennungen insgesamt (und damit die Zahl der Falschtrennungen) durch geeignete Parameter zu reduzieren.

## Hinweise zur Spezifikation MODUS

Die Korrektur der als falsch erkannten Trennungen wird bei kurzen Texten zweckmäßigerweise mit Hilfe des TUSTEP-Editors vorgenommen, bei längeren Texten mit Hilfe des Makros \*SILKOR. In beiden Fällen genügt es, die Silbentrennungsliste mit MODUS=- zu erstellen. Sie wird dabei kürzer und leichter lesbar als bei MODUS=+.

Nur wenn die Silbentrennungen mit dem Programm KAUSFUEHRE korrigiert werden sollen, ist MODUS=+ erforderlich, da in den entsprechenden Korrekturanweisungen die am Anfang und am Ende des getrennten Wortes stehenden Interpunktionszeichen und Steueranweisungen mit berücksichtigt werden müssen.

# Silbentrennungsmarkierungen einfügen

#\*SILMARKE

## Makro:

#\*SILMARKE

QUELLE	= datei	Name der Datei mit dem Text, in dem Markierungen für Kann-Trennstellen und/oder Trennverbot ergänzt werden sollen.
ZIEL	= datei	Name der Datei, in die der Text mit den ergänzten Markierungen ausgegeben werden soll.
MODUS	= -STD-	* Normalfall: der verfügbare Arbeitsspeicher reicht für alle markierten Wörter aus, die in jeweils einem (zu BEREICH angegebenen) Bereich der Datei bzw. (bei BEREICH=-STD-) in der ganzen Datei MARKIERUNGEN enthalten sind.
	= FILE	Die markierten Wörter sollen über Zwischendateien verarbeitet werden.
LOESCHEN	= -	* Daten in der ZIEL-Datei nicht löschen.
	= +	Daten in der ZIEL-Datei zuerst löschen.
MARKIERUNGEN	= datei	Name der Datei, die Wörter enthält, in denen Kann-Trennstellen durch \ und Stellen mit Trennverbot durch \\ markiert sind.
BEREICH	= -STD-	* Normalfall: die ganze zur Spezifikation MARKIERUNGEN angegebene Datei soll verarbeitet werden.
	= bereich	Bereichsangabe in der Form s.z-s.z, wenn nicht die ganze zur Spezifikation MARKIERUNGEN angegebene Datei verarbeitet werden soll. Angabe mehrerer Bereiche möglich. Bei MODUS=-STD- werden die angegebenen Bereiche jeweils einzeln nacheinander verarbeitet.
	= name	Wenn die zur Spezifikation MARKIERUNGEN angegebene Datei eine Segment-Datei ist: Name des Segments, das die markierten Wörter enthält. Angabe mehrerer Bereiche möglich. Bei MODUS=-STD- werden die angegebenen Bereiche jeweils einzeln nacheinander verarbeitet.
TAGS	= -	* Der Text in der QUELL-Datei enthält keine XML-Tags.



	= +	Der Text in der QUELL-Datei enthält XML-Tags: Wörter in den Tags nicht verändern.
APOS	= -	* Normalfall: Apostroph zählt als Trennzeichen.
	= +	Apostroph zählt als Wortbestandteil; Wörter mit Apostroph sind in der Datei MARKIERUNGEN eigens aufgeführt.

## Leistung

Mit diesem Makro können in einem Text die Wörter markiert werden, die bei der automatischen Silbentrennung eine Sonderbehandlung erfahren sollen. Beim Kopieren des Textes von der QUELL-Datei in die ZIEL-Datei werden zu diesem Zweck die Wörter, die in der zur Spezifikation MARKIERUNGEN angegebenen Datei enthalten sind und dort die entsprechenden Markierungen aufweisen, im Text ausgetauscht. In der zur Spezifikation MARKIERUNGEN angegebenen Datei müssen die Wörter an den Kann-Trennstellen durch »\« und an den Stellen, an denen sie nicht getrennt werden dürfen, durch »\« markiert sein.

Dieses Verfahren ersetzt die Verwendung von Ausnahmelisten, wie sie bei anderen Silbentrennprogrammen üblich sind.

## Hinweise

Es empfiehlt sich, nur die Wörter in die zur Spezifikation MARKIERUNGEN angegebene Datei aufzunehmen, die bei der automatischen Silbentrennung falsch oder unschön getrennt würden. Mit dem Makro \*SILMARKO kann diese Datei entsprechend optimiert werden.

Enthält die zur Spezifikation MARKIERUNGEN angegebene Datei sehr viele (mehr als ca. 90 000) Einträge, so muss die Verarbeitung intern in mehreren Schritten durchgeführt werden, da das Makro, abhängig von der Länge der einzelnen Einträge, nur etwa 90 000 Einträge auf einmal verarbeiten kann. Sind mehr markierte Wörter zu verarbeiten, so empfiehlt es sich, um Plattenplatz und – vor allem bei langen QUELL-Dateien – Rechenzeit zu sparen, die zu MARKIERUNGEN angegebene Datei mit Hilfe der Spezifikation BEREICH für die Verarbeitung so zu unterteilen, dass die in jeweils einem Bereich stehenden Wörter in den zur Verfügung stehenden Arbeitsspeicher passen. Alternativ kann mit der Angabe MODUS=FILE ein Verfahren zur Markierung gewählt werden, das über Zwischendateien arbeitet, jedoch mehr Plattenplatz und mehr Rechenzeit benötigt.

## Zur automatischen Silbentrennung

Das Satzprogramm führt, wenn die Silbentrennung nicht über Parameterangaben oder Steueranweisungen unterbunden wird, eine automatische Silbentrennung durch, falls Zeileneinteilung und ggf. Randausgleich dies erfordern. Dabei werden die Regeln der Silbentrennung für die deutsche Sprache zugrunde gelegt.

Keine automatisch durchgeführte Silbentrennung arbeitet fehlerfrei. Selbst umfangreiche Ausnahmelisten würden dem nicht abhelfen, da es viele Wörter gibt, die je nach ihrem Kontext anders zu trennen sind (Beispiel: »Spie-lende Kinder«, aber: »kurz vor Spiel-ende«; »Stau-becken« und »Staub-ecken«). Vor allem zusammengesetzte Wörter bieten eine schier unerschöpfliche Quelle von Fehlermöglichkeiten.

Für deutsche Texte mit einem nicht ungewöhnlich hohen Anteil an zusammengesetzten Wörtern muss der TUSTEP-Benutzer erfahrungsgemäß mit ca. 0,5 Prozent fehlerhaften Trennungen (und einer höheren Rate sogenannter »unschöner« Trennungen wie »Versauf-bau«) rechnen.

Obwohl das verwendete Trennprogramm die für das Deutsche geltenden Trennregeln voraussetzt, ist es auch für romanische Sprachen noch brauchbar; die Fehlerrate liegt dann bei etwa 5 Prozent. Für englisch-sprachige Texte sind diese Trennregeln ungeeignet; hier empfiehlt es sich, die Zahl der Trennungen insgesamt (und damit die Zahl der Falschtrennungen) durch geeignete Parameter zu reduzieren.

Um falsche oder unschöne Trennungen zu vermeiden, muss in dem zu trennenden Wort hinter einen Buchstaben, hinter dem nicht getrennt werden soll, die Steueranweisung für Trennverbot »^« oder »\« eingefügt werden. Soll bevorzugt an bestimmten Stellen getrennt werden, so muss hinter den Buchstaben, hinter denen getrennt werden soll, die Steueranweisung für Kann-Trennstelle »\« eingefügt werden. Bei der automatischen Silbentrennung werden so markierte Stellen bevorzugt berücksichtigt. Ein Wort, das durch »\« markierte Kann-Trennstellen enthält, wird an anderen Stellen nur getrennt, wenn mindestens drei Buchstaben oder andere Zeichen zwischen dieser Trennung und der Kann-Trennstelle liegen.

# Silbentrennungsmarkierungen optimieren

#\*SILMARKO

## Makro:

#\*SILMARKO

QUELLE	= datei	Name der Datei, die die zu optimierende Liste von Wörtern mit Silbentrennungsmarkierungen enthält.
ZIEL	= datei	Name der Datei, in die die optimierte Liste von Wörtern mit Silbentrennungsmarkierungen geschrieben werden soll.
MODUS	= -STD-	* Doppelte und überflüssig markierte Wörter aus der Liste entfernen. Überflüssig markiert sind Wörter, die eine Markierung (für Kann-Trennstelle oder für Trennverbot) enthalten, aber bei automatischer Silbentrennung nur und genau an den Stellen getrennt würden, die als Kann-Trennstelle markiert sind.
	= -	Wie MODUS=-STD-, zusätzlich aber auch die Wörter aus der Liste entfernen, die keine Markierung enthalten.
	= +	Wie MODUS=-STD-, aber in den Wörtern an den Stellen eine Markierung für Trennverbot einfügen, an denen die automatische Silbentrennung weitere Trennungen erzeugen würde.
LOESCHEN	= -	* Daten in der ZIEL- und in der KONTROLL-Datei nicht löschen.
	= -STD-	Daten in der KONTROLL-Datei nicht löschen; Einträge, die bereits in der ZIEL-Datei vorhanden sind, in die Prüfung nach den Vorschriften von MODUS mit einbeziehen.
	= +	Daten in der ZIEL- und in der KONTROLL-Datei zuerst löschen.
KONTROLL	= datei	Name der Datei, in die diejenigen Wörter ausgegeben werden sollen, die nicht in die ZIEL-Datei ausgegeben werden. Wörter, die doppelt in der QUELL-Datei vorhanden sind und nur deswegen nicht in die ZIEL-Datei ausgegeben werden, werden jedoch nicht in die KONTROLL-Datei ausgegeben.  Bei MODUS=-STD- und MODUS=+ kann zu dieser Spezifikation, durch Apostroph getrennt, eine zweite

Datei angegeben werden. In diese Datei werden (zusätzlich zur Ausgabe in die ZIEL-Datei) diejenigen Wörter ausgegeben, die bei automatischer Silbentrennung auch an Stellen getrennt würden, die nicht durch »^« oder »\« (für Trennverbot) bzw. »\« (für Kann-Trennstelle) markiert sind. Die zusätzlichen Trennungen werden durch »-« angegeben.

	= -	* Keine Ausgabe zur Kontrolle.
FRUEHEST	= n	Trennen frühestens nach dem n-ten Buchstaben.
	= 1	* Trennen frühestens nach dem ersten Buchstaben.
SPAETEST	= n	Letzte Trennung vor dem n-letzten Buchstaben.
	= 1	* Letzter Trennung vor dem letzten Buchstaben.
REGELN	= NEU	* Bei der Silbentrennung werden die Regeln der Rechtschreibreform von 1996 berücksichtigt.
	= ALT	Der Silbentrennung werden die Regeln vor der Rechtschreibreform von 1996 zugrundegelegt.

## Leistung

Mit diesem Makro können aus Wortlisten mit Silbentrennungsmarkierungen, die im Standard-Makro \*SILMARKE benutzt werden sollen, doppelte und überflüssige Wörter entfernt werden. Überflüssig sind Wörter, die eine Markierung enthalten und bei automatischer Silbentrennung nur und genau an den Stellen getrennt würden, an denen eine Kann-Trennstelle markiert ist. Bei MODUS=- gelten auch die Wörter als überflüssig, die weder eine Markierung für Kann-Trennstelle noch eine Markierung für Trennverbot enthalten. Bei MODUS=+ wird außerdem geprüft, ob ein Wort bei automatischer Silbentrennung an nicht markierten Stellen getrennt würde. An solchen Stellen wird die Markierung »\« für Trennverbot eingefügt.

## Hinweis

Soll die Wortliste zur Markierung eines Textes verwendet werden, bei dessen Formatierung bzw. Satz die Silbentrennung eingeschaltet bleibt, so sollte sie mit MODUS=+ optimiert werden.

Bei fremdsprachigen (insbesondere englischen) Texten genügt es häufig, die Silbentrennung auf lange Wörter zu beschränken. Will man Trennfehler vermeiden, so geschieht dies am einfachsten dadurch, dass beim Formatieren oder Setzen solcher Texte die automatische Silbentrennung ausgeschaltet und Worttrennung somit nur an den durch »\« (für Kann-Trennstelle) markierten Stellen erlaubt wird. Die Wortliste, die zum Einfügen von Silbentrennungsmarkierungen in einen solchen Text benutzt werden soll, wird zweckmäßigerweise mit MODUS=- optimiert.

Weitere Hinweise zur Verwendung solcher Wortlisten als Ausnahmelisten für die automatische Silbentrennung werden in der Beschreibung des Makros \*SILMARKE gegeben.

# Umbrechen von mehrspaltigen Einschüben

**#\*SUMBRUCH**

## Makro:

#\*SUMBRUCH

QUELLE	=	datei	Name der Datei mit dem Protokoll eines Satzprogrammablaufs.
ZIEL	=	datei	Name der Datei, in die der Text mit den für den mehrspaltigen Umbruch notwendigen Steueranweisungen ausgegeben werden soll.
MODUS	=	-	* (wird z. Zt. noch nicht ausgewertet).
LOESCHEN	=	-	* Daten in der PROTOKOLL-Datei nicht löschen. Die ZIEL-Datei muss in diesem Fall leer sein.
	=	+	Daten in der ZIEL- und in der PROTOKOLL-Datei zuerst löschen.
PARAMETER	=	-	* Keine Parameterangaben.
	=	datei	Name der Datei mit den Parametern.
	=	*	Die Parameter folgen auf den Makroaufruf und sind mit *EOF abgeschlossen.
PROTOKOLL	=	+	* Das Fehlerprotokoll soll ins Ablaufprotokoll ausgegeben werden.
	=	-STD-	Das Fehlerprotokoll soll in die Standard-Protokoll-Datei ausgegeben werden.
	=	datei	Name der Datei, in die das Fehlerprotokoll ausgegeben werden soll.

## Leistung

Mit diesem Makro werden für den Umbruch von Texten, in denen ein- und mehrspaltige (fortlaufende) Textblöcke auf ein und derselben Seite zusammen vorkommen, die die endgültige Anordnung fixierenden Steueranweisungen in den Text eingesetzt. Das Makro muss zu diesem Zweck nach einem Lauf des Satzprogramms (mit MODUS=T), bei dem der Umbruch des Textteils festgelegt wird, und vor einem weiteren Lauf des Satzprogramms (mit MODUS=A) aufgerufen werden.

## Erläuterungen

Das Makro \*SUMBRUCH wird nicht für den Satz von mehrspaltigen Tabellen benötigt, ebensowenig für Werke oder Teile von Werken, die insgesamt mehrspaltig gesetzt werden sollen. Es findet vielmehr Anwendung beim Satz von Werken, die zwischen den einspaltigen Textteilen Einschübe von mehrspaltigen Blöcken mit (fortlaufendem) Text enthalten; für das Satzprogramm sind solche mehrspaltigen Blöcke mit der Steueranweisung `&!s (n, mmm)` zu codieren.

Der Umbruch der in solchen Texten enthaltenen mehrspaltigen Einschübe wird zunächst mit dem Satzprogramm (`MODUS=T`) vorbereitet. Dabei wird der endgültige Zeilenumbruch für alle Textzeilen festgelegt und der Platzbedarf für die mehrspaltigen Einschübe berechnet.

Dieser Schritt ist ggf. so lange (nach entsprechender Korrektur) zu wiederholen, bis keine den Umbruch betreffenden Fehlerkommentare mehr auftreten.

Anschließend werden mit dem Makro \*SUMBRUCH die Zeilen der mehrspaltigen Einschübe nach den Bedürfnissen des Seitenumbruchs umgeordnet und mit den für den endgültigen Satz benötigten Steueranweisungen versehen. Eingabe für dieses Makro ist die PROTOKOLL-Datei des vorhergehenden Satzprogrammlaufs; als Ergebnis wird eine Datei erzeugt, die als QUELL-Datei für einen anschließenden Satzprogrammlauf (`MODUS=A`) dient. Erst dieser Lauf stellt den endgültigen seitenumbrochenen Satz einschließlich der richtigen Anordnung der mehrspaltigen Einschübe her.

Aus diesem Ablauf ergibt sich, dass umbruch-verändernde Korrekturen nur möglich sind, solange in der Textfassung korrigiert wird, die als Eingabe für den Satzprogrammlauf vor der Festlegung des Umbruchs und dem Umordnen der Zeilen mehrspaltiger Teile dient.

## Parameter

Das Makro erwartet als einzigen Parameter die Angaben zur Anordnung der mehrspaltigen Blöcke:

**SPA**            `&!s (n, mmm) NPV1 NPN1 NPN2 ... NPNn`

Dabei ist `&!s (n, mmm)` die Angabe, die in der QUELL-Datei den Beginn des  $n$ -spaltigen Teils mit einer Spaltenbreite von je  $mmm$  Punkt kennzeichnet.  $NPV1$  ist ein Zahlenwert und gibt an, wieviel Punkt vor der 1. Spalte freigehalten werden sollen,  $NPNn$  ist ein Zahlenwert und gibt an, wieviel Punkt hinter der  $n$ -ten Spalte freigehalten werden sollen.

Dieser Parameter muss für jede im Text vorkommende Steueranweisung `&!s (n, mmm)` mit verschiedenen Spaltenzahlen und -breiten angegeben werden.

## SGML/XML-Tags in Makros für SATZ umwandeln

#\*TAGS

### Makro:

#\*TAGS

QUELLE	= datei	Name der Datei mit dem nach SGML bzw. XML ausgezeichneten Text.
ZIEL	= datei	Name der Datei, in die die Satzprogramm-Makros ausgegeben werden sollen. Wird, durch Apostroph getrennt, eine zweite ZIEL-Datei angegeben, so wird in diese Datei der Text aus der zu QUELLE angegebenen Datei ausgegeben; dabei werden vor jedem Tag alle an dieser Stelle noch offenen Tags ergänzt.
MODUS	= -STD-	* Außerhalb des root-Elements sowie zwischen dem root-Tag und dem Anfangs-Tag des ersten ihm untergeordneten Elements dürfen keine Zeichen außer Leerzeichen vorkommen. Groß- und Kleinschreibung in den Tags wird unterschieden.
	= -	Die bei MODUS=-STD- vorgesehene Prüfung soll nicht durchgeführt werden. Groß- und Kleinschreibung in den Tags wird nicht unterschieden.
	= K	Die zu QUELLE angegebene Datei war in einem vorangehenden Aufruf von *TAGS als zweite ZIEL-Datei angegeben bzw. enthält entsprechende Ergänzungen. Diese Ergänzungen sollen rückgängig gemacht werden.
LOESCHEN	= +	Daten in den zu ZIEL, PROTOKOLL, REIHENFOLGE oder HIERARCHIE angegebenen Dateien zuerst löschen.
PARAMETER	= -	* Keine Parameterangaben.
	= *	Die nicht aufzunehmenden Tags folgen auf den Makroaufruf und sind mit *EOF abgeschlossen.
	= datei	Name der Datei mit den nicht aufzunehmenden Tags.
PROTOKOLL	= -	* Keine Protokoll-Ausgabe.
	= +	Das Protokoll mit einer alphabetischen Liste der im Text vorkommenden Tags einschließlich ihrer hierarchischen Ordnung und den zugehörigen Stellenangaben soll ins Ablaufprotokoll ausgegeben werden.



	= -STD-	Das Protokoll soll in die Standard-Protokoll-Datei ausgegeben werden.
	= datei	Name der Datei, in die das Protokoll ausgegeben werden soll.
REIHENFOLGE	= -	* Keine Ausgabe der Tags in der Textreihenfolge.
	= datei	Name der Datei, in die die im Text vorkommenden Tags und ihre hierarchische Ordnung in der Textreihenfolge ausgegeben werden sollen.
HIERARCHIE	= -	* Keine Ausgabe der vorkommenden Tags.
	= datei	Name der Datei, in die die im Text vorkommenden Tags in ihrer hierarchischen Ordnung (jedes vorkommende Tag nur einmal, wie in der ZIEL-Datei, jedoch ergänzt um die übergeordneten Tags) ausgegeben werden sollen.
REFERENZEN	= n	In der PROTOKOLL-Datei bei Einträgen mit mehr als n Belegstellen Referenzen nicht mit ausgeben
	= 999999	* alle Referenzen mit ausgeben
META	= -	* Alles auswerten.
	= tag	Namen der Tags (z. B. meta oder teilHeader), die (Meta-)Datenbereiche kennzeichnen, die nicht in die Auswertung einbezogen werden sollen. Mehrere Namen können durch Apostroph getrennt angegeben werden. Groß- und Kleinschreibung beachten!

## Leistung

Mit diesem Makro kann eine Liste der in einer Text-Datei vorkommenden SGML- bzw. XML-Tags unter Berücksichtigung ihrer hierarchischen Ordnung erzeugt und in Parameter für das Satzprogramm verwandelt werden.

Außerdem kann mit diesem Makro eine Kopie der Text-Datei erstellt werden, in der vor jedem Tag alle an der betreffenden Stelle noch offenen Tags ergänzt werden.

Das Makro setzt voraus, dass in der Text-Datei alle Start- und End-Tags zu den einzelnen Elementen vorhanden sind, also keine Minimierung von Start- oder End-Tags vorgenommen wurde.

Tags, zu denen kein End-Tag vorhanden ist (»milestones«), werden in die ZIEL-Datei mit der Parameterkennung MAC für »einfache Makros« ausgegeben; sie tauchen in den Dateien zu PROTOKOLL, REIHENFOLGE und HIERARCHIE nicht auf.

Die paarweise vorkommenden Tags (für Elemente mit Start- und End-Tag) sowie die empty-element-Tags werden mit der Parameterkennung MAH in die ZIEL-Datei ausgegeben, und zwar in der Reihenfolge, die ihrer hierarchischen Ordnung entspricht, die also ggf. ihre Stellung innerhalb von übergeordneten, durch entsprechende Tags

bezeichneten Elementen berücksichtigt. Die Stellung eines Tags innerhalb der Hierarchie wird in der ZIEL-Datei außerdem dadurch gekennzeichnet, dass ab Spalte 11 für jedes übergeordnete Makro, das einem noch offenen (nicht durch ein End-Tag abgeschlossenen) Tag entspricht, ein Punkt vor der öffnenden spitzen Klammer eingesetzt wird. In den zu PROTOKOLL, REIHENFOLGE und HIERARCHIE angegebenen Dateien werden alle noch offenen übergeordneten Tags vor dem jeweiligen Tag ergänzt.

Wenn über Parameter nicht anders verlangt ist, werden Tags mit Attributen nur bis einschließlich des den Namen abschließenden Leerzeichens in die Listen aufgenommen bzw. in Satzmakros verwandelt. In die ZIEL-Datei wird zusätzlich eine Liste dieser Tags mit allen vorkommenden Attributen als Kommentar (mit Leerzeichen als Parameterkennung) ausgegeben.

Bei MODUS=-STD- werden zusätzlich zu den Tags noch die Parameter

```
maz          0 1
```

(Unterscheiden von Groß- und Kleinschreibung in den Tag-Namen) und

```
tbe          sa.za-se.ze
```

(mit den Satznummern der ersten und der letzten Zeile des root-Elements) in die ZIEL-Datei ausgegeben.

## Parameter

Ist zu PARAMETER nichts oder »-« angegeben, so werden alle Tags aus dem Text erhoben. Nur die Tags, zu denen auch End-Tags vorhanden sind, und die empty-element-Tags werden in die hierarchische Ordnung einbezogen. Diese Tags werden in die ZIEL-Datei mit der Parameterkennung MAH ausgegeben; die übrigen Tags werden mit der Parameterkennung MAC ausgegeben.

Das Ende jedes nicht leeren Elements muss im Text mit einem End-Tag markiert sein. Dies gilt auch für leere Elemente, falls diese durch ein Start-Tag mit unmittelbar folgendem End-Tag codiert sind.

Sind zu PARAMETER Tags angegeben, so unterbleibt die Suche nach tags, zu denen kein End-Tag vorhanden ist (»milestones«). Nur die angegebenen Tags und die zugehörigen End-Tags werden mit der Parameterkennung MAC ausgegeben und nicht in die hierarchische Ordnung einbezogen. Zu allen übrigen Tags muss ein End-Tag im Text vorhanden sein.

Als Parameter müssen die Tags, die nicht in die hierarchische Ordnung einbezogen werden sollen, je einzeln linksbündig in einer Zeile aufgeführt werden. Die zugehörigen End-Tags dürfen nicht angegeben werden; Parameter für die zu den angegebenen Anfangs-Tags passenden End-Tags werden automatisch ergänzt.

Tags mit Attributen, die nicht in die Liste der hierarchisch verschachtelten Tags aufgenommen werden sollen, müssen nicht mit allen Attributen aufgeführt werden. Es genügt, diese bis zum ersten Leerzeichen einschließlich aufzuführen.

Sollen hierarchische Tags mit Attributen nicht nur mit ihren Namen, sondern als

ganze Makros (einschließlich der Attribute) in die hierarchische Ordnung einbezogen werden, so ist als Parameter das Tag bis zum ersten Leerzeichen einschließlich anzugeben, gefolgt von einer geschlossenen spitzen Klammer und einem Pluszeichen (die zugehörigen End-Tags dürfen dann nicht angegeben werden). Beispiel:  
<div >+

Leerzeilen oder Zeilen mit Leerzeichen in den Spalten 1–3 der Parameterzeilen gelten als Kommentar.

# Makro-Auflösungen in \*TAGS-Ergebnis übertragen

#\*TAGUEB

## Makro:

#\*TAGUEB

QUELLE	= datei	Name der Datei mit den neuen, noch nicht aufgelösten Tags.
ZIEL	= datei	Name der Datei, in die die Tags aus der zu QUELLE angegebenen Datei ausgegeben werden sollen, nachdem sie um die Auflösungen aus den zu MAKROS angegebenen Satzmakros ergänzt wurden.
MODUS	= -STD-	* Groß- und Kleinschreibung in Tags relevant (wie in XML)
	= -	Groß- und Kleinbuchstaben in Tag-Namen gelten als gleichwertig
LOESCHEN	= -	* Daten in der ZIEL-Datei nicht löschen.
	= +	Daten in der ZIEL-Datei zuerst löschen.
MAKROS	= *	Die Satzmakros, deren Auflösungen in die neuen Makros übernommen werden sollen, folgen auf den Makroaufruf und sind durch *EOF abgeschlossen.
	= datei	Name der Datei mit den Satzmakros, deren Auflösungen übernommen werden sollen.

## Leistung

Mit diesem Makro können Auflösungen von Satzmakros, die mit dem Makro \*TAGS gewonnen und anschließend für das Satzprogramm aufgelöst wurden, auf identische, noch nicht aufgelöste Makros mit gleicher hierarchischer Einordnung übertragen werden.

## TeX-Seiten (PostScript) in Grafikdatei einfügen

#\*TEXGRAF

### Makro:

#\*TEXGRAF

QUELLE	=	datei	Name der Datei, die eine mehrseitige PostScript-Ausgabe von TeX enthält.
ZIEL	=	datei	Name der Datei, in der die einzelnen TeX-Seiten zum Einbinden in die Ausgabe mit dem Makro *PSAUS gesammelt werden sollen.
LOESCHEN	=	-STD-	In der ZIEL-Datei Grafiken, die die gleiche Nummer haben wie eine der neuen Grafiken, durch diese ersetzen.
	=	-	* Enthält die ZIEL-Datei schon eine Grafik mit der gleichen Nummer wie eine der neuen Grafiken, so wird im Dialog nachgefragt, ob die Verarbeitung abgebrochen oder die Grafik ersetzt werden soll; wurde das Makro im Batch aufgerufen, so wird die Verarbeitung abgebrochen, wenn schon eine solche Grafik vorhanden ist.
NUMMER	=	n	Nummer, die die erste TeX-Seite in der ZIEL-Datei erhalten soll. Die größte Nummer, die vergeben werden kann, ist 999999.
ABSTAND	=	x*y	Zahlenpaar; die erste Zahl x gibt den horizontalen Abstand des linken Randes der TeX-Seite vom linken Rand der im Text dafür vorgesehenen Aussparung bzw. von der Position des Aufrufs an; die zweite Zahl y gibt den vertikalen Abstand des unteren Randes der TeX-Seite vom oberen Rand der Aussparung bzw. von der Position des Aufrufs an.
DREHEN	=	n	TeX-Seiten um n Grad gegen den Uhrzeigersinn drehen.
	=	-	* TeX-Seiten nicht drehen.
FAKTOR	=	n	Vergrößerungs- oder Verkleinerungsfaktor (in Prozent), mit dem die Größe der TeX-Seiten verändert werden soll.
	=	-STD-	* Keine Größenveränderung der TeX-Seiten.
SCHNEIDEN	=	x*y+x*y	2 Koordinatenpaare, die die linke obere und die rechte untere Ecke eines Rechtecks angeben, innerhalb

dessen die eingebundenen TeX-Seiten stehen müssen; was über die Ränder dieses Rechtecks hinaussteht, wird abgeschnitten. Der 0-Punkt für diese Angaben ist die linke obere Ecke der Aussparung bzw. die Position des Aufrufs.

	= -	* Die eingebundenen TeX-Seiten werden nicht beschnitten.
NAME	= -STD-	Die TeX-Seiten erhalten in der ZIEL-Datei einen Standardnamen.
INHALT	= +	Neues Inhaltsverzeichnis der ZIEL-Datei ins Ablaufprotokoll ausgeben.
	= -	* Kein Inhaltsverzeichnis ausgeben.

## Leistung

Dieses Makro dient zur Vorbereitung der Integration von längeren TeX-Ausgaben in die TUSTEP-Satzausgabe mit dem Makro \*PSAUS. Die als PostScript-Datei vorliegende TeX-Ausgabe muss zu diesem Zweck in einzelne Seiten aufgeteilt werden, die dann wie Abbildungen in die TUSTEP-Satzausgabe eingebunden werden.

## Erläuterungen

TeX-Ausgabe, die in PostScript-Form vorliegt, kann beim Satz wie eine Abbildung (Steueranweisungen  $\$ \$ m m m / n n n \# i i i \$ \$ \{ \$ \$ \# i i i \$ \$ \{$  etc.) eingebunden werden. Dazu muss jeweils eine TeX-Ausgabeseite in einer eigenen PostScript-Datei stehen. Solche Einzelseiten können mit dem Makro \*GRAFIK zur Integration in die Satzausgabe vorbereitet werden.

Enthält die PostScript-Datei mehr als eine Seite TeX-Ausgabe, so können die einzelnen Seiten dieser Datei mit dem Makro \*TEXGRAF separiert und für \*PSAUS aufbereitet werden.

Bei der Aufbereitung mit dem Makro \*TEXGRAF müssen die Abbildungen so verschoben und ggf. gedreht werden, dass sie in die beim SATZ dafür vorgesehenen Aussparungen passen. Gegebenenfalls müssen in der TeX-Ausgabe enthaltene Kopf- und Fußtexte sowie Seitennummern abgeschnitten werden.

Dies geschieht mit Hilfe von entsprechenden Angaben zu den Spezifikationen ABSTAND, DREHEN und SCHNEIDEN.

## Angaben zur Spezifikation ABSTAND

Die x- und y-Abstände, die beim Makro \*GRAFIK bzw. \*TEXGRAF zur Spezifikation ABSTAND anzugeben sind, können wie folgt bestimmt werden:

Man zeichnet um den Teil der TeX-Ausgabe, der als Abbildung in die TUSTEP-Ausgabe übernommen werden soll, einen Rahmen, der der Aussparung entspricht, der in der Satzausgabe dafür vorgesehen ist. Ist keine Aussparung, sondern nur ein Ankerpunkt für die Abbildung im Satz vorgesehen, so markiert man in der Abbildung die Position dieses Ankerpunktes. Soll die Abbildung gedreht werden, so ist das Blatt zuvor so zu drehen, dass die Abbildung in der endgültigen Ausrichtung steht. Sodann misst man den Abstand des 0-Punkts (= ursprüngliche linke untere Ecke des ungedrehten Blattes) der TeX-Ausgabe in x- und y-Richtung vom Ankerpunkt (= linke obere Ecke) dieses Rahmens.

Die Angaben sind in (Didot-)Punkt zu machen.

Für x-Werte gilt: Liegt der 0-Punkt der in die endgültige Richtung gedrehten TeX-Ausgabe links vom Ankerpunkt, so ist der gemessene Abstand als negativer x-Wert anzugeben; liegt er rechts, so ist der Abstand als positiver x-Wert anzugeben.

Für y-Werte gilt: Liegt der 0-Punkt der in die endgültige Richtung gedrehten TeX-Ausgabe unterhalb vom Ankerpunkt, so ist der gemessene Abstand als negativer y-Wert anzugeben; liegt er oberhalb, so ist der Abstand als positiver y-Wert anzugeben.

### **Angaben zur Spezifikation** DREHEN

Muss die Abbildung gedreht werden, so ist zur Spezifikation DREHEN der Winkel, um den die Abbildung gedreht werden soll, in Grad anzugeben. Drehungen gegen den Uhrzeigersinn sind als positive Werte anzugeben, Drehungen im Uhrzeigersinn als negative Werte.

### **Angaben zur Spezifikation** SCHNEIDEN

Nachdem die Abbildung so verschoben und gedreht ist, dass der gewünschte Ausschnitt an der richtigen Stelle steht, wird mit den Angaben zu SCHNEIDEN die TeX-Seite beschnitten. Dazu wird über die (bereits verschobene und ggf. gedrehte) Seite ein Rechteck gelegt, innerhalb dessen der gewünschte Ausschnitt steht. Der Nullpunkt für die Koordinatenangaben der linken oberen und der rechten unteren Ecke dieses Rechtecks ist die linke obere Ecke der Aussparung. Die Koordinatenangaben können positiv (x-Richtung nach rechts, y-Richtung nach oben) oder negativ (x-Richtung nach links, y-Richtung nach unten) gemacht werden. Was außerhalb dieses Rechtecks steht, wird abgeschnitten, d. h. nicht mit ausgegeben.

## Attribute in XML-Tags sortieren

**#\*XMLATTSOR**

### Makro:

#\*XMLATTSOR

QUELLE	= datei	Name der Datei, in der die Attribute der XML-Tags sortiert werden sollen.
ZIEL	= datei	Name der Datei für die sortierten Daten.
LOESCHEN	= -	* Daten in der ZIEL-Datei nicht löschen
	= +	Daten in der ZIEL-Datei zuerst löschen.
REIHENFOLGE	= ALPHA	* Die Attribute sollen alphabetisch (nach der Standard-Sortierfolge) sortiert werden.
	= *	Die Angaben zur Sortierung folgen auf den Makroaufruf und sind mit *EOF abgeschlossen.

### Leistung

Mit diesem Makro können Attribute in XML-Tags zum Zweck der einfacheren Weiterverarbeitung in eine feste vorgegebene Reihenfolge gebracht werden. Die Sortierung kann alphabetisch nach der Standard-Sortierfolge oder nach den zu REIHENFOLGE angegebenen Vorschriften erfolgen.

### Angaben zu REIHENFOLGE

Zu REIHENFOLGE können Parameterzeilen mit der Kennung XS1 für das intern benutzte Programm #SVORBEREITE angegeben werden.



## XML-Tags in Satz-Zieldatei wiederherstellen

#\*XMLZIEL

### Makro:

#\*XMLZIEL

QUELLE	= datei	Name der ZIEL-Datei eines Satzlaufes mit möglicherweise aufgelösten XML-Tags.
ZIEL	= datei	Name der Datei für die Daten mit den wieder hergestellten XML-Tags
	= -	* (nur bei MODUS=P sinnvoll): Keine XML-Daten ausgeben.
MODUS	= -STD-	* XML-Tags wieder herstellen und anschließend Daten auf Übereinstimmung mit der QUELL-Datei des Satzlaufes überprüfen.
	= S	* wie -STD-, dabei in QUELLE enthaltene Silbentrennungen aufheben.
	= T	nur XML-Tags wieder herstellen.
	= TS	* wie T, dabei in QUELLE enthaltene Silbentrennungen aufheben.
	= P	nur QUELL-Datei und ZIEL-Datei auf Übereinstimmung von Tags und Text überprüfen.
	= PS	* wie P, dabei in QUELLE enthaltene Silbentrennungen aufheben.
LOESCHEN	= -	* Enthält die ZIEL-Datei schon Daten, so wird das Makro abgebrochen. Daten in der zu KORREKTUREN angegebenen Datei nicht löschen.
	= +	Daten in der ZIEL-Datei und in der KORREKTUREN-Datei zuerst löschen.
KORREKTUREN	= -	* Die Liste der Abweichungen zwischen QUELL- und ZIEL-Datei des Satzlaufes soll in eine temporäre Datei geschrieben und im Editor angezeigt werden.
	= datei	Name der Datei für die Liste der Abweichungen.
SATZQUELLE	= -	* (nur bei MODUS=T sinnvoll) Keine Angabe zur QUELL-Datei des Satzlaufes
	= name	Name der QUELL-Datei des Satzlaufes
TAGENTF	= -	* keine Tags entfernen
	= -STD-	Die mit den Tags <att_ausw> und </att_ausw> be-

zeichneten Elemente werden nicht in die ZIEL-Datei übernommen.

= tag      Name der Tags, die statt `<att_ausw>` und `</att_ausw>` die nicht zu übernehmenden Elemente bezeichnen. Mehrere Tag-Namen sind durch Apostroph zu trennen. Bei Tags mit Attributen darf nur der Name angegeben werden. Die zu entfernenden Elemente dürfen nicht ineinander geschachtelt sein.

## Leistung

Das Makro XMLZIEL stellt XML-tags wieder her, die wegen Angabe von »-1« als zweitem Wert im Parameter MAZ in der ZIEL-Datei eines Satzlaufes möglicherweise noch aufgelöst sind, und / oder überprüft (ggf. nach der Wiederherstellung der Tags) die Übereinstimmung von Text und Tags zwischen der QUELL- bzw. (bei MODUS=-STD-) der ZIEL-Datei und der zu SATZQUELLE angegebenen Datei. Unterschiede in der Silbentrennung werden bei der Prüfung übergangen. Die gefundenen Unterschiede werden in der Form von Korrekturanweisungen (mit Originalwortlaut und Abweichung) angezeigt bzw. ausgegeben.

**Register**

Register

Das folgende Register bezieht sich lediglich auf die ersten 843 Seiten. Die Beschreibung der parametergesteuerten Programme (einschließlich Satzprogramm) und der Programme und Makros zur Satzherstellung ist also durch dieses Register nicht erschlossen.

In der Schriftart `Courier` gesetzte Registereinträge verweisen auf Stellen, an denen das betreffende Stichwort in einem Kommando oder einer Anweisung genau in dieser Form verwendet wird.

Die Referenz besteht aus einem Kennzeichen und einer Seitenzahl. Das Kennzeichen ist entweder identisch mit den ersten beiden Buchstaben des Kommandos oder hat folgende Bedeutung:

<code>cdt</code>	=	Code-Tabellen
<code>das</code>	=	Dateistruktur
<code>dat</code>	=	Dateien
<code>dtr</code>	=	Daten-Transfer
<code>edt</code>	=	Editor
<code>kmd</code>	=	Kommandos
<code>mkr</code>	=	Makros
<code>prm</code>	=	Parameter
<code>ruf</code>	=	Aufruf
<code>sys</code>	=	System-Umgebung
<code>zvr</code>	=	Zeichenvorrat

```

!= mkr: 462
#!x zvr: 765 766
#$$?dateiname mkr: 408
#$$makroname mkr: 407
#$$makroname$dateiname mkr: 408
#^_ dat: 36
#'_x zvr: 765 766
#(...) zvr: 767 771
#+nnn dat: 36
#,_x zvr: 765 766
#-nnn dat: 36
#._x zvr: 765 766
#:(NL) edt: 385
#:(PWN) edt: 384 385
#:(xx) edt: 384 385
#;x zvr: 765 766
#=nnn dat: 36
#[xx] UM: 242
#[xxxx] UM: 245
#ABMELDE AB: 119
#AENDERE AE: 122
#ANMELDE AN: 124
#BEENDE BE: 127
#DATEI DA: 128
#DEFINIERE DE: 132
#DRUCKE DR: 142
#DUMPE DU: 147
#DVORBEREITE DV: 148
#EDIERE ED: 149
#EINFUEGE EI: 156
#FAUFBEREITE FA: 157
#FEHLERHALT FE: 158
#FORMATIERE FO: 160
#HALT HA: 161
#HILFE HI: 162
#HOLE HO: 165
#INFORMIERE IN: 167
#KAUSFUEHRE KA: 171
#KOPIERE KO: 172
#LISTE LI: 173
#LOESCHE LO: 182
#MAKRO MA: 186
#MBAUSGABE MBA: 187
#MBEINGABE MBE: 189
#MBINFORMIERE MBI: 191
#MBKOPIERE MBK: 193
#MBLABEL MBL: 197
#MBTEST MBT: 199
#MISCHE MI: 201
#MODI MO: 203
#NORMIERE NO: 205
#NUMMERIERE NU: 206
#PARAMETER PA: 207
#PROTOKOLL PR: 209
#RAUFBEREITE RA: 215
#RETTE RE: 216
#RVORBEREITE RV: 220
#SATZ SA: 221
#SIGNAL SI: 222
#SORTIERE SO: 224
#SPRUEFE SP: 227
#SUCHE SU: 228
#SVORBEREITE SV: 229
#TESTE TE: 230
#TITEL TI: 231
#TUE TU: 233
#UMWANDLE UM: 237
#VAUFBEREITE VA: 247
#VERGLEICHE VE: 248
#WAHLSCHALTER WA: 249
#WISCHEN WI: 250
#x+ zvr: 798
#x- zvr: 798
#ZEIT ZE: 251
$$! mkr: 413
$$* mkr: 432
$$+ mkr: 431
$$- mkr: 431
$$= mkr: 417
% mkr: 477
< mkr: 462
<> mkr: 414 473
<= mkr: 462
<[xy] mkr: 443
<ANZAHL> sys: 76
<datei> edt: 303
<DATEI> sys: 76
<editor> edt: 303
<GERAET> sys: 76
<NAME> sys: 76
<segment> edt: 303
<ZUSATZ> sys: 76
> mkr: 462
>= mkr: 462
>[xy] mkr: 443
*BEISPIEL mkr: 659
*CASH zvr: 785
*CB dtr: 55
*CGI sys: 98
*CMD sys: 90
*DECO DE: 138
*DERI sys: 84 85
*DESI sys: 80 82 85 86
*DOWNLOAD dtr: 58 60 63
*DRUBE HI: 162
*E edt: 257
*EOF sys: 87 kmd: 111 113 DE: 140
*EOF* kmd: 111
*EXPORT dtr: 56
*IMPORT dtr: 56
*KOMA mkr: 659
*M mkr: 408
*MBUPDATE dtr: 60 65
*PB dtr: 55
*PS2PDF dat: 39
*SEND2TUSTEP sys: 103

```

\*SERVER sys: 101  
 \*SESO dat: 35  
 \*SUCHE mkr: 669  
 \*TADE edt: 301  
 \*TUSTEP2STICK inh: 20 21  
 \*UPLOAD dtr: 57 60 63  
 \*VEMO edt: 280  
 \*ZEPDF dat: 40  
 \*ZEPS dat: 39  
 && mkr: 461 602  
 .AB. mkr: 464  
 .AND. mkr: 461 601  
 .CA. mkr: 468  
 .CE. mkr: 468  
 .CO. mkr: 468  
 .CS. mkr: 468  
 .CT. mkr: 467  
 .EI. mkr: 464  
 .EQ. mkr: 462 463 465 469 472  
 .EQV. mkr: 461  
 .EW. mkr: 464  
 .GE. mkr: 462 464  
 .GT. mkr: 462 463  
 .HA. mkr: 465  
 .HN. mkr: 465  
 .ID. mkr: 463  
 .IN. mkr: 471  
 .LE. mkr: 462 464  
 .LT. mkr: 462 463  
 .MA. mkr: 467 468  
 .NA. mkr: 464 465  
 .NC. mkr: 467  
 .NE. mkr: 462 463 465 469 472  
 .NI. mkr: 463  
 .NM. mkr: 468  
 .NOT. mkr: 473  
 .NP. mkr: 464  
 .NS. mkr: 464  
 .ON. mkr: 470 471  
 .OR. mkr: 461 601  
 .PA. mkr: 464  
 .SI. mkr: 464  
 .SS. mkr: 464  
 .SW. mkr: 464  
 .XOR. mkr: 461  
 || mkr: 461 602  
 == mkr: 462  
 @@@@ kmd: 111 mkr: 432  
 @@@@ kmd: 111 mkr: 432  
 {} mkr: 414 473  
  
 Abbruch eines TUSTEP-Programms dat: 40  
 FE: 158  
 Abfragen, ob Austausch-Tabelle definiert ist  
 mkr: 466  
 Abfragen, ob Datei abgeschlossen ist mkr: 471  
 Abfragen, ob Datei ein bestimmtes Segment ent-  
 hält mkr: 471  
 Abfragen, ob Datei eine Scratch-Datei ist LI:  
 178 mkr: 471  
 Abfragen, ob Datei eine Segment-Datei ist  
 mkr: 471  
 Abfragen, ob Datei existiert mkr: 470  
 Abfragen, ob Datei leer ist mkr: 471  
 Abfragen, ob Datei nach Satznummern sortiert  
 ist mkr: 471  
 Abfragen, ob Datei zum Lesen angemeldet ist  
 LI: 178 mkr: 470  
 Abfragen, ob Datei zum Schreiben angemeldet  
 ist LI: 178 mkr: 471  
 Abfragen, ob ein Projekt existiert mkr: 469  
 470  
 Abfragen, ob Recherchier-Tabelle definiert ist  
 mkr: 466  
 Abfragen, ob Sektion definiert ist mkr: 466  
 Abfragen, ob Stapelspeicher definiert ist mkr:  
 466  
 Abfragen, ob Stringgruppe definiert ist mkr:  
 466  
 Abfragen, ob Submakro definiert ist mkr: 466  
 Abfragen, ob Such-Tabelle definiert ist mkr:  
 466  
 Abfragen, ob Variable definiert ist mkr: 466  
 Abfragen, ob Wörterbuch definiert ist mkr:  
 466  
 Abfragen, ob Zeichengruppe definiert ist mkr:  
 466  
 Abkürzen von Kommandonamen kmd: 110  
 Abkürzen von Spezifikationsnamen kmd: 110  
 Abkürzung auflösen edt: 377  
 Abkürzungen auflösen mkr: 512  
 Ablaufprotokoll PR: 209  
 Ablaufprotokoll anhalten PR: 212  
 Abmelden von Dateien dat: 38 AB: 119  
 NO: 205 mkr: 482  
 Abschließen von Dateien dat: 40  
 ACCESS mkr: 597 603 611 619 631  
 ACCESS/ FILE mkr: 598  
 ACCESS/ LINES mkr: 597  
 ACCESS/ PAGES mkr: 597  
 ACCESS/ RECORDS mkr: 597  
 ACCESS/ STREAM mkr: 597  
 ACCESS/ STRUCTURES mkr: 597  
 ACCESS/ VARIABLE mkr: 598  
 ACCUMULATE mkr: 551  
 ADD mkr: 446  
 ADD\_ATTRIBUTES mkr: 562  
 ADD\_END\_TAG edt: 387  
 ADD\_MRK edt: 385  
 ADD\_MRK:text edt: 370  
 ADD\_STRT\_TAG edt: 387  
 ADD\_VALUES mkr: 556  
 Addition mkr: 477  
 ADJUST\_LENGTH mkr: 542  
 ADJUST\_PATH mkr: 499  
 ADJUST\_SIZE mkr: 541

- Adressaufkleber FA: 157  
 Aktualisieren von laufenden Nummern NU: 206  
 Aktualisieren von Verweisen NU: 206  
 aktueller Projektname dat: 26  
 Akzent-Codierungen zvr: 761 773  
 Akzent-Codierungen ignorieren edt: 330  
 Akzentbuchstaben ED: 151 edt: 285 zvr: 761  
 Akzente in der arabischen Schrift zvr: 784  
 Akzente in der kyrillischen Schrift zvr: 788  
 Akzente in der lateinischen Schrift zvr: 773  
 Akzente in der russischen Schrift zvr: 788  
 ALL mkr: 502  
 ALPHA\_INDEX mkr: 570  
 ALPHA\_SORT mkr: 570  
 Alt edt: 260 mkr: 472  
 Alt-kirchenslavische Schrift zvr: 789  
 ALTER mkr: 446  
 Analysieren von Dateien DU: 147  
 AND mkr: 442  
 Ändern von Dateinamen AE: 122  
 Ändern von Projektnamen AE: 122  
 Anfangs-Tag edt: 317 386 mkr: 559 617  
 Anfangskennung prm: 728  
 Anfragen mkr: 426  
 Anführungszeichen ergänzen mkr: 513  
 angemeldete Dateien auflisten LI: 173  
 Anmelden von Dateien dat: 37 AN: 124 mkr: 481  
 ANSI DE: 133 LI: 176 UM: 238 edt: 257 mkr: 576 578  
 ANSI cdt: 822  
 Anweisungen anzeigen/korrigieren/wiederholen edt: 304  
 Anweisungszeile edt: 270  
 Anwendungen beenden mkr: 458  
 Anwendungen starten edt: 394 mkr: 458  
 Anzeige-Modus edt: 395  
 Anzeigen von Dateien dat: 38 edt: 394  
 Apostroph mkr: 417  
 APP\_MRK edt: 385  
 APP\_MRK:text edt: 370  
 APPEND mkr: 446 513 543  
 APPEND\_ATTRIBUTES mkr: 563  
 Arabische Schrift zvr: 782  
 ASCII LI: 176 UM: 238 edt: 257  
 ASCII cdt: 840  
 ASCII, deutsch cdt: 816  
 ASCII, international cdt: 815  
 ASCII-Datei dat: 31 mkr: 430  
 ASCII-Datei edieren edt: 280  
 ASCII-EBCDIC UM: 238  
 ASCII-Zeichen, Zeichengruppen-Kennung für prm: 709  
 ASK mkr: 426 427 583  
 ASK/COLOR:xx mkr: 426  
 ASK/ERROR mkr: 426  
 ASK/SAVE mkr: 426  
 ASK/WARNING mkr: 426  
 ASSIGN mkr: 557  
 atab edt: 332  
 Attribut-Namen edt: 300 317 323  
 Attribut-Namen definieren/voreinstellen ED: 153  
 Attribut-Prüfung definieren/voreinstellen ED: 153  
 Attribut-Werte edt: 300 317  
 Attribute abfragen mkr: 561  
 Attribute einfügen mkr: 562 563  
 Attribute entfernen mkr: 563  
 Attribute ergänzen mkr: 562 563  
 Attribute setzen mkr: 561  
 Attribute sortieren mkr: 564  
 Attribute umbenennen mkr: 563  
 Aufbereiten von Registern RA: 215  
 Aufbereiten zum Drucken FO: 160 SA: 221  
 Auflösen von Abkürzungen mkr: 512  
 Aufruf des Editors sys: 103 104 105 ED: 149  
 Aufruf von Anwendungen edt: 394 mkr: 458  
 Aufruf von Makros ED: 150 mkr: 407  
 Aufruf von Programmen mkr: 457  
 Aufruf von TUSTEP sys: 79 89 98  
 Aufteilen von Zeichenfolgen mkr: 516 517 519  
 AUS mkr: 472 473  
 Ausdruck, Gestaltung des FO: 160 RA: 215 SA: 221  
 Ausdrucken → Drucken  
 Ausgabe auf Drucker dat: 38 DR: 142  
 Ausgabe in eine Band-Datei MBA: 187  
 Ausgabe in eine Datei umleiten mkr: 430  
 Ausgabe ins TUSTEP-Fenster dat: 38 DR: 142  
 Ausnahmezeichenfolgen LI: 179 edt: 331 332 prm: 711 713 715 721 743 745 746 752  
 Austausch-Tabelle edt: 332 prm: 721 752  
 Austausch-Tabellen definieren mkr: 440  
 Austausch-Tabellen freigeben mkr: 442  
 Austausch-Tabellen prüfen mkr: 442  
 Austauschen/Einfügen von Textteilen EI: 156  
 Austauschen/Ersetzen von Zeichenfolgen KO: 172 edt: 311  
 Auswählen der Seiten zum Drucken DR: 144 DV: 148  
 Auswählen von Dateien AB: 120 AN: 125 LI: 175 176 178 LO: 183 MBK: 195  
 Auswählen von Daten KO: 172  
 Auswählen von Textteilen prm: 728  
 Auswählen von Zeichenfolgen mkr: 547 548 549  
 Auszeichnungen zvr: 798  
 Auszeichnungen ignorieren edt: 330  
 Autorenregister RV: 220  
 Backspace kmd: 114 edt: 260 mkr: 669

BACKTAB mkr: 474 646  
 Band-Datei dat: 32 DA: 128  
 Band-Datei einrichten MBL: 197  
 Band-Datei kopieren MBK: 193  
 Band-Datei löschen MBL: 197  
 Band-Datei testen MBT: 199  
 Band-Datei, Ausgabe in eine MBA: 187  
 Band-Datei, Eingabe von einer MBE: 189  
 Band-Datei-Inhalt abfragen MBI: 191  
 BASE64 mkr: 578 580  
 BASENAME mkr: 499  
 BATCH sys: 75 mkr: 473  
 Bausteinbriefe EI: 156  
 Bedingungen mkr: 437 461  
 Beenden der Dateneingabe kmd: 113 edt:  
 273 mkr: 427  
 Beenden der Parametereingabe kmd: 113  
 Beenden des Editors edt: 273  
 Beenden einer TUSTEP-Sitzung sys: 88 BE:  
 127 NO: 205  
 Beenden von Anwendungen mkr: 458  
 BEEP edt: 339 400 mkr: 434  
 BEG\_CMD\_LINE edt: 352  
 BEG\_REC edt: 353  
 Beginnen einer TUSTEP-Sitzung sys: 87  
 begr, Spaltenbegrenzung edt: 329  
 Begrenzungszeichen edt: 331 332 mkr: 444  
 prm: 706 738  
 Begrenzungszeichen für linken Rand prm:  
 720 751  
 Begrenzungszeichen für rechten Rand prm:  
 720 751  
 Belegter Speicherplatz mkr: 502  
 beliebige Zeichen, Zeichengruppen-Kennung für  
 prm: 742  
 Benutzeridentifikation sys: 74  
 Benutzeridentifikation abfragen mkr: 507  
 ber, Dateibereich edt: 328  
 Berechnen des Divisionsrestes mkr: 478  
 Berechnen des Maximums mkr: 478  
 Berechnen des Minimums mkr: 478  
 Berechnen des Zeitraums mkr: 504  
 Bereichsumschaltzeichen prm: 739  
 Beschreibung anzeigen HI: 162  
 Beschreibung ausdrucken HI: 162  
 Beschreibung durchsuchen SU: 228  
 Beschreibung von Makros IN: 167  
 Bibliothekskärtchen FA: 157  
 BINAER UM: 238  
 BINARY mkr: 428 495 496 612 620 639 640  
 BINARY/NLF mkr: 428 495 496  
 BLANK mkr: 441  
 Blättern in Dateien edt: 275 357  
 blockweise Kopieren HO: 165  
 BOM UM: 241  
 Bookmark → Lesezeichen  
 BREAK mkr: 440  
 Breite des Editor-Fensters ED: 150  
 Breite des Editorfensters edt: 337  
 Breite des TUSTEP-Fensters DE: 134  
 Breite eines Feldes abfragen mkr: 592  
 BROWSE edt: 394 mkr: 458 472 640  
 BROWSE/CANCEL mkr: 458  
 BROWSE/WAIT mkr: 458  
 BRS\_MRK edt: 394  
 BSP edt: 361 mkr: 647 672 681 684 687  
 BSP\_BLANK edt: 361  
 Buchstaben, Zeichengruppen-Kennung für  
 mkr: 678 prm: 709 742  
 BUILD mkr: 440 443  
 Bulgarische Schrift zvr: 787  
 BUTTON mkr: 654  
 BUTTON/CANCEL mkr: 655  
 BUTTON/CENTER mkr: 654  
 BUTTON/ENTER mkr: 654  
 BUTTON/FKEY mkr: 655  
 BUTTON/HELP mkr: 655  
 BYTE mkr: 578 580  
 BYTES mkr: 492  
  
 C\_GROUP mkr: 443 444 466  
 CALL mkr: 451  
 CALL\_D DE: 140  
 CANCEL DE: 140 edt: 398 mkr: 460 474  
 646 670 674 675 677  
 CAPS mkr: 510  
 CASE mkr: 439  
 CASE/EXACT mkr: 439  
 cd (change directory) → Projektnamen definie-  
 ren  
 CDATA mkr: 617  
 CDROM mkr: 502  
 CENTER mkr: 511  
 CGI mkr: 474 576 579  
 CGI-Makro sys: 75 98 101 mkr: 407  
 CGI-Script mkr: 429 508  
 CHAR edt: 376  
 CHECK mkr: 442 444 482 608 634  
 CHECK\_BRACKETS mkr: 569  
 CHECK\_FC edt: 291 381  
 CHECK\_FN edt: 291 381  
 CHECK\_FT edt: 291 381  
 CHECK\_FX edt: 381  
 CHECK\_TAGS mkr: 568  
 checkbox mkr: 653  
 CHG\_MACROS:name~ edt: 383  
 CHG\_MACROS:name\* edt: 383  
 CHG\_SETTINGS edt: 395  
 CHG\_TF edt: 399  
 CHOOSE mkr: 588  
 CHOOSE\_ONE mkr: 590  
 CLEAR edt: 365 mkr: 448 449 450 648 651  
 652 653 672 680  
 CLICK edt: 399 mkr: 474 587  
 Clipboard → Zwischenablage



CLOSE mkr: 482  
 CLR\_CMD\_LINE edt: 364  
 CLR\_LINE edt: 364  
 CLR\_MATCH edt: 400  
 CLR\_MRK edt: 370  
 CMD mkr: 474  
 CMD-Makro sys: 75 89 93  
 CMD\_LINE edt: 352  
 Code Page zvr: 761  
 COLLECT mkr: 543  
 COLLECT\_EXACT mkr: 544  
 COLOR mkr: 593 642 644 670 671  
 COLOR:n edt: 399  
 Colorierung definieren/wechseln/löschen/abfragen  
 edt: 298  
 Colorierung einstellen ED: 153  
 Colorierungen löschen ED: 153  
 COLS mkr: 507  
 COMBINE mkr: 552  
 COMMAND sys: 83  
 COMPARE mkr: 489  
 COMPILE mkr: 453 455  
 COMPILED mkr: 506  
 Compilieren von Makroanweisungen mkr:  
 453  
 COMPLETE mkr: 512  
 COMPLETE\_NAME mkr: 497  
 CONCAT mkr: 514  
 CONFIRM edt: 398  
 CONT sys: 88 89  
 CONTINUE mkr: 438  
 COPY mkr: 488  
 COPY edt: 268  
 copy → Kopieren  
 COUNT mkr: 446 537 609 635  
 COUNTER mkr: 448 575  
 COUNTER\_MRK edt: 371  
 cp (copy) → Kopieren  
 CP10000 cdt: 823  
 CP10029 cdt: 824  
 CP1250 cdt: 821  
 CP1252 cdt: 822  
 CP437 cdt: 818  
 CP850 cdt: 819  
 CP852 cdt: 820  
 CPnnn DE: 133 LI: 176 UM: 238 edt: 257  
 CR edt: 394 mkr: 475 648 650 651 652 653  
 654 670 672 673 674 677 680 687  
 CREATE mkr: 445 449 479  
 CREATE/EXACT mkr: 445  
 CREATE/INFIX mkr: 445  
 CREATE/TAGS mkr: 445  
 Ctrl kmd: 114 edt: 259 mkr: 669  
 CUR\_DN DE: 140 edt: 351 mkr: 646 649  
 652 653 654 656 670 671 673 674 677 680  
 683 687  
 CUR\_LE edt: 352 mkr: 646 654 656 671 680  
 683 687  
 CUR\_POS:n;m edt: 357  
 CUR\_RI edt: 352 mkr: 646 654 656 671 680  
 683 687  
 CUR\_UP DE: 140 edt: 351 mkr: 647 650  
 652 653 654 656 670 671 673 674 677 680  
 683 687  
 CURRENT mkr: 500 507  
 CURRENT\_FILE mkr: 508  
 CURRENT\_SEGMENT mkr: 508  
 Cursor edt: 339 351  
 CURSOR\_COLUMN mkr: 592  
 CURSOR\_POSITION mkr: 592  
 CURSOR\_ROW mkr: 592  
 CYCLE mkr: 436  
 Cyrillische Schrift zvr: 786 789  
  
 DATA mkr: 414  
 DATE mkr: 502 503 504  
 DATE\_n edt: 390 mkr: 502  
 DATEI mkr: 470  
 Datei nicht abgeschlossen dat: 41  
 Datei unvollständig RE: 216  
 Datei, permanente dat: 29 DA: 128 130  
 Datei, temporäre dat: 29 DA: 128 130  
 Datei-Transfer mkr: 639  
 Dateianfang mkr: 494  
 Dateianfang auflisten LI: 173  
 Dateianfang zeigen edt: 275  
 Dateiartern dat: 31  
 Dateibereich ber edt: 328  
 Dateien abmelden dat: 38 AB: 119 BE: 127  
 NO: 205 mkr: 482  
 Dateien abschließen dat: 40  
 Dateien analysieren DU: 147  
 Dateien anmelden dat: 37 AN: 124 DA:  
 128 mkr: 481  
 Dateien anzeigen dat: 38 edt: 394  
 Dateien auswählen AB: 120 AN: 125 LI:  
 175 176 178 LO: 183 MBK: 195  
 Dateien dekomprimieren UM: 244  
 Dateien drucken dat: 38 DR: 142  
 Dateien edieren ED: 149  
 Dateien einrichten dat: 37 DA: 128 mkr:  
 479  
 Dateien entschlüsseln UM: 237  
 Dateien entsperren mkr: 491  
 Dateien holen HO: 165 edt: 281  
 Dateien komprimieren UM: 244  
 Dateien löschen dat: 38 AB: 121 BE: 127  
 LO: 182 NO: 205 mkr: 484  
 Dateien mischen EI: 156 MI: 201  
 Dateien mit unzulässigem Namen dat: 29  
 Dateien neu nummerieren HO: 165  
 Dateien prüfen mkr: 482  
 Dateien reorganisieren dat: 40  
 Dateien retten dat: 40 RE: 216 edt: 282  
 283

- Dateien sortieren SO: 224  
 Dateien sperren dat: 41 mkr: 490  
 Dateien testen TE: 230  
 Dateien umbenennen AE: 122 mkr: 486  
 Dateien umwandeln UM: 237  
 Dateien vergleichen TE: 230 VE: 248  
 Dateien verschlüsseln UM: 237  
 Dateiende mkr: 494  
 Dateiende auflisten LI: 173  
 Dateiende zeigen edt: 275  
 Dateigröße das: 46  
 Dateigröße abfragen LI: 178 mkr: 492  
 DATEINAME mkr: 470  
 Dateiname mkr: 428  
 Dateinamen dat: 25 27 29  
 Dateinamen abfragen IN: 168 LI: 173 edt:  
 280 mkr: 494 497 500  
 Dateinamen ändern AE: 122  
 Dateinamen definieren DE: 134 141 mkr:  
 423  
 Dateinamen einstellen edt: 280  
 Dateinamen löschen edt: 278 280 mkr: 424  
 Dateinamen prüfen mkr: 499  
 Dateinamen, Großbuchstaben in dat: 29  
 Dateinamen, Sonderzeichen in dat: 29  
 Dateinamen, Umlaute in dat: 29  
 Dateinamen, unzulässige dat: 29  
 Dateisperre mkr: 490  
 Dateistruktur das: 45  
 Dateititel das: 45 LI: 176 177 edt: 278 286  
 305 mkr: 428 494  
 Dateititel abfragen LI: 173  
 Dateititel definieren TI: 231  
 Dateititel löschen edt: 286  
 Dateititel merken TI: 231  
 Dateititel übertragen TI: 231  
 Dateityp dat: 31 DA: 128 LI: 178  
 Dateityp abfragen LI: 178 mkr: 471  
 Dateizugriffe – Allgemeines mkr: 597  
 Dateizugriffe – Daten mit Anfangs- und Ende-  
 kennungen mkr: 610  
 Dateizugriffe – Daten mit definierten Strukturen  
 mkr: 627  
 Dateizugriffe – Daten mit Tags mkr: 617  
 Dateizugriffe – Sätze, Zeilen, Seiten mkr: 603  
 Daten → Dateien  
 Daten auswählen KO: 172  
 Daten holen HO: 165  
 Daten in Makros mkr: 407  
 Daten löschen LO: 182 edt: 277 309 mkr:  
 483  
 Daten prüfen KO: 172  
 Daten retten RE: 216  
 Daten-Transfer auf andere Rechner dtr: 59  
 Daten-Transfer auf lokalem Rechner dtr: 56  
 Datenausgabe in Dateien mkr: 428  
 Datenaustausch MBA: 188  
 Datenaustauschformat dtr: 59 UM: 243 244  
 Datenbank-Anweisung → Such-Anweisung  
 Dateneingabe kmd: 113 edt: 273 mkr: 427  
 Dateneingabe beenden kmd: 113 edt: 273  
 Dateneingabe von Dateien mkr: 427  
 Datenschutz AB: 121 BE: 127 LO: 184  
 NO: 205 SO: 226 WI: 250  
 Datensicherung dtr: 64  
 Datensicherung auf USB-Speicher-Stick dtr:  
 63  
 Datensicherung mit Band-Dateien dtr: 63  
 Datenstruktur mkr: 627  
 Datenträger → Träger  
 Datum edt: 390 391 mkr: 493 502 503  
 DEC\_MRK\_HEX edt: 371  
 DEC\_NUM edt: 389  
 DECIMAL0 mkr: 577  
 DECIMAL0,DECIMAL1 mkr: 579  
 DECIMAL1 mkr: 577  
 DECIMAL2 mkr: 577 579  
 DECIMAL3 mkr: 577 579  
 DECIMAL4 mkr: 577 579  
 DECIMAL5 mkr: 577 579  
 Deckblatt bei Druckausgaben sys: 74 DE:  
 133 DR: 143 145 PR: 210  
 DECMCS DE: 133 LI: 176 UM: 238 edt:  
 258 cdt: 825  
 DECODE mkr: 576  
 DEFAULT mkr: 439  
 DEFINE mkr: 422 423 424  
 DEFINE/CLIPBOARD mkr: 425  
 DEFINE/Common mkr: 419  
 DEFINE/EDIT\_FILE mkr: 425  
 DEFINE/SYSTEM mkr: 424  
 DEFINE/TUSTEP mkr: 422  
 DEFINE/VOLUME mkr: 420  
 DEFINE:name edt: 386  
 DEFINE\_BM edt: 374  
 DEFINE\_CB edt: 393  
 Definieren einer TUSTEP-Sitzung sys:  
 80  
 Definieren von Trägerangaben mkr: 420  
 Definieren von Variablen mkr: 418  
 Definierte Dateinamen dat: 29 DE: 134 141  
 IN: 168 mkr: 423  
 Definitionen ED: 154  
 Dekomprimieren von Dateien UM: 244  
 DEL edt: 361 mkr: 475 647 651 652 653 672  
 674 680  
 DEL\_BEG edt: 362 mkr: 648 672  
 DEL\_BLANK edt: 361  
 DEL\_END edt: 362 mkr: 648 672  
 DEL\_LE edt: 363  
 DEL\_LINE edt: 364  
 DEL\_REC edt: 276 277 364  
 DEL\_REC\_NEXT edt: 364  
 DEL\_RI edt: 363  
 DEL\_WORD edt: 363 mkr: 647  
 DELAY sys: 83

Delete kmd: 114 HI: 163 edt: 260 mkr: 448 450 472 484 485 669  
 DELETE edt: 268  
 delete → Löschen  
 Deutscher ASCII-Code cdt: 816  
 Deutscher EBCDIC-Code cdt: 839  
 diakritische Zeichen zvr: 773  
 DIALOG sys: 75 mkr: 474  
 DICTIONARY mkr: 445 447 448 466  
 DIGIT\_INDEX mkr: 572  
 DIGIT\_SORT mkr: 571  
 DIGITS mkr: 465  
 directory → Verzeichnis  
 DIRNAME mkr: 499  
 DISCOUNT mkr: 446  
 DISK\_NAME mkr: 502  
 DISPLAY mkr: 582 641  
 DISPLAY:name edt: 384  
 DISPLAY\_MRK edt: 385  
 DISTANCE mkr: 539  
 DISTANCE\_EXACT mkr: 539  
 Division mkr: 477  
 Divisionsrest mkr: 477 478  
 DO mkr: 450  
 DO:name edt: 375  
 DO\_EMPTY:name edt: 378  
 DO\_FIRST:name edt: 379  
 DO\_MATCH:name edt: 380  
 DO\_NOTHING edt: 401  
 DO\_ZERO:name edt: 379  
 Dollar mkr: 417  
 Domain-Name abfragen mkr: 506  
 DONE mkr: 476  
 DOWNLOAD mkr: 639  
 Download dtr: 60  
 DRIVES mkr: 502  
 Druckausgabe auf Drucker dat: 38 DR: 142  
 Druckausgabe in Dateien dat: 38 DR: 142  
 Druckausgabe ins TUSTEP-Fenster dat: 38 DR: 142  
 Druckeffekte zvr: 798  
 Drucken von Dateien dat: 38 DR: 142  
 Drucken vorbereiten DV: 148  
 Drucken, Aufbereiten zum FO: 160 SA: 221  
 Drucken, Auswählen der Seiten DR: 144 DV: 148  
 Drucken, System-Kommando sys: 76  
 DRUCKER mkr: 466  
 Druckernamen abfragen mkr: 506  
 Druckernamen sys: 80 DR: 144 PR: 213  
 Druckertypen abfragen LI: 174  
 DUMMY mkr: 657  
 DUP\_LINE edt: 360  
 Durchsuchen von Texten SU: 228  
 E-mail dtr: 60 mkr: 508  
 EASTER mkr: 504  
 EBCDIC LI: 176 UM: 238  
 EBCDIC cdt: 839 840  
 EBCDIC, deutsch cdt: 839  
 EBCDIC, international cdt: 838  
 EBCDIC-ASCII UM: 238  
 ECHO\_OFF edt: 396  
 ECHO\_ON edt: 396  
 Edieren einer ASCII-Datei edt: 280  
 Edieren im Original edt: 280  
 Edieren in einer Kopie edt: 280  
 Edieren von Daten/Dateien ED: 149  
 EDIT sys: 89 mkr: 458 585 644  
 EDIT\_FILE edt: 390  
 EDIT\_INFO mkr: 494  
 Editierdistanz mkr: 539  
 Editor aufrufen sys: 103 104 105 ED: 149  
 Editor starten sys: 103 104 105 mkr: 458  
 Editor-Datei dat: 36  
 Editor-Datei abfragen/wechseln edt: 284  
 Editor-Fensters, Breite des ED: 150  
 Editor-Fensters, Höhe des ED: 150  
 Editor-Informationen mkr: 425  
 Editor-Zwischenspeicher edt: 258  
 Editoranweisungen, lange edt: 272  
 Editorfenster edt: 270  
 Editorfensters, Breite des edt: 337  
 Editorfensters, Höhe des edt: 337  
 Editormakros, Voreinstellung der eigene Kommandos mkr: 407  
 EIN mkr: 472 473  
 Einfügemodus kmd: 117 edt: 333 339 359 mkr: 672  
 Einfügen im Editorfenster edt: 359 361 362 363 364 365  
 Einfügen von Attributen mkr: 562 563  
 Einfügen von Daten edt: 273 308  
 Einfügen von Sätzen edt: 276  
 Einfügen von Segmenten/Dateien mkr: 452  
 Einfügen von Textteilen EI: 156  
 Einfügen von Zeichenfolgen mkr: 527 545  
 Eingabe von einer Band-Datei MBE: 189  
 Eingabeaufforderung kmd: 113  
 Einrichten von Dateien dat: 37 DA: 128 mkr: 479  
 Einrichten von Projekten DA: 128 mkr: 479  
 Einschließen von Zeichenfolgen mkr: 532 533  
 Einstellungen abfragen mkr: 472  
 Einstellungen für den Editor ED: 154  
 Eintragen von Daten edt: 273  
 Element der Suchzeichenfolge prm: 716 747  
 Elementbereiche prm: 722  
 ELIMINATE mkr: 523 525  
 Eliminieren von Textteilen prm: 728  
 ELSE mkr: 438  
 ELSEIF mkr: 438  
 EMPTY mkr: 471  
 EMPTY\_ELEMENT\_TAG mkr: 560

ENC\_MRK\_HEX edt: 371  
 ENCLOSE mkr: 532 533  
 ENCODE mkr: 578  
 ENCODE\_OFF edt: 395  
 ENCODE\_ON edt: 395  
 End kmd: 114 HI: 164 edt: 260 mkr: 670  
 END\_CMD\_LINE edt: 352  
 END\_REC edt: 353  
 END\_TAG mkr: 560  
 ENDACCESS mkr: 597  
 ENDACCESS/PRINT mkr: 597  
 ENDCOMPILE mkr: 453  
 ENDDATA mkr: 419  
 Ende-Tag edt: 317 386 mkr: 559 560 617  
 Endekennung prm: 728  
 ENDEXCHANGE mkr: 431  
 ENDFILE mkr: 430  
 ENDFILE/PRINT mkr: 430  
 ENDIF mkr: 438  
 ENDLOOP mkr: 435 436 437  
 Endlosschleife mkr: 437  
 ENDSECTION mkr: 450  
 ENDSELECT mkr: 439  
 ENDSTRUCTURE mkr: 627  
 ENDSUBMACRO mkr: 451  
 ENDTRACE mkr: 460  
 ENDWINDOW mkr: 641  
 ENGLISH mkr: 474  
 Enter kmd: 114 edt: 260 398 mkr: 475  
 646 669 670 671 673 675 677 681 684 687  
 ENTER\_MRK:text edt: 370  
 Entfernen von Attributen mkr: 563  
 Entfernen von Tags mkr: 565  
 Entfernen von Zeichenfolgen mkr: 523 525  
 546 548 550  
 ENTITIES mkr: 576 579  
 Entities UM: 244 245  
 Entschlüsseln von Daten/Dateien UM: 237  
 Entsperren von Dateien mkr: 491  
 Environment-Variable → System-Variable  
 EOF mkr: 476 606 607 608 615 623  
 EOR mkr: 475 615 623  
 ERASE mkr: 428 429 483 491 598 603 609  
 612 620 631 635  
 erase → Löschen  
 Ergänzen des Projektnamens dat: 26  
 Ergänzen von Attributen mkr: 562 563  
 Ergänzen von Nullen mkr: 511  
 Ergänzen von Textteilen KO: 172  
 Ergänzen von Wertelisten mkr: 554 556  
 Ergänzen von Zeichen mkr: 511  
 Ergänzen/Einfügen von Textteilen EI: 156  
 ERROR edt: 401 mkr: 433 474  
 ERROR/RESET mkr: 434  
 ERROR/STOP mkr: 433 434  
 ERROR\_STOP mkr: 472  
 Ersatzzeichenfolgen LI: 179 edt: 332 prm:  
 721 752  
 Ersetzen eines Segments edt: 283  
 Ersetzen von Zeichenfolgen mkr: 526 527  
 528 530 531 546  
 Ersetzen/Austauschen von Zeichenfolgen KO:  
 172 edt: 311  
 Ersetzen/Einfügen von Textteilen EI: 156  
 Escape kmd: 114 HI: 163 edt: 260 mkr:  
 669  
 EXACT mkr: 439 441 445  
 EXCH\_CB edt: 393  
 EXCH\_CHAR edt: 397  
 EXCH\_CHAR\_LE edt: 397  
 EXCH\_CHAR\_RI edt: 397  
 EXCH\_CUR edt: 357  
 EXCH\_MRK edt: 371  
 EXCH\_REC\_DN edt: 397  
 EXCH\_REC\_NR edt: 392  
 EXCH\_REC\_UP edt: 397  
 EXCHANGE mkr: 431 447 530 531  
 EXECUTE mkr: 456 457  
 EXECUTE/CONTINUE mkr: 457  
 EXECUTE/QUIET mkr: 456  
 EXIST mkr: 476  
 EXIT sys: 87 89 mkr: 436 607 609 633 634  
 Export einer Datei mkr: 639  
 Export von Daten dtr: 56  
 EXTEND DE: 139 mkr: 591  
 Extension dat: 26 sys: 103 104 105  
 EXTRACT mkr: 519 521  
 Extrahieren von Zeichenfolgen mkr: 519 521  
 522  
 Farbe eines Feldes abfragen mkr: 593  
 Farben abfragen IN: 168  
 Farben definieren/wechseln/löschen/abfragen  
 edt: 298  
 Farben einstellen DE: 134 140 ED: 153  
 edt: 335 mkr: 693  
 Farbgruppen edt: 298  
 FC\_# edt: 290  
 FDF dat: 31 DA: 128 mkr: 471 480  
 FDF-Datei DA: 128  
 Fehlerflag mkr: 433  
 FEHLERHALT mkr: 472  
 Fehlerhalt FE: 158  
 Fehlermeldung kmd: 111 mkr: 426 432 433  
 Fehlersuche edt: 396 mkr: 459  
 FETCH mkr: 422 423 425 427  
 FETCH/CLIPBOARD mkr: 425  
 FETCH/EDIT\_FILE mkr: 425  
 FETCH/SYSTEM mkr: 424 425  
 FETCH/TUSTEP mkr: 423  
 FETCH\_CB edt: 393  
 FETCH\_SEG\_M edt: 386  
 Fettdruck zvr: 798  
 FIELD mkr: 641  
 FIFO mkr: 449

FILE mkr: 428 430 470 494 598  
 FILE/APPEND mkr: 430  
 FILE/BINARY mkr: 428  
 FILE/ERASE mkr: 430  
 FILE/ERASE mkr: 428  
 FILE/ERASE/BINARY mkr: 429 509  
 FILE/FILE mkr: 428  
 FILE/PRINT mkr: 428  
 FILE/PROGRAM mkr: 428 430  
 FILE/SEGMENT mkr: 429  
 FILE/TITLE mkr: 428  
 FILE\_MRK edt: 371  
 FILE\_NAME mkr: 470 497  
 FILE\_NAMES mkr: 500  
 FILES mkr: 500  
 FILTER mkr: 549  
 FILTER\_INDEX mkr: 550  
 FIND mkr: 607 608 609 633 634  
 FIND/NEXT mkr: 608 634  
 FIND/PREVIOUS mkr: 608 634  
 FIND/REVERSE mkr: 608 634  
 FIND\_DIFF mkr: 538  
 FIND\_STRING mkr: 535  
 FINISH mkr: 644  
 FIRST\_LINE edt: 353  
 FIRST\_PAGE mkr: 493  
 first in mkr: 449  
 first out mkr: 449  
 FKEY mkr: 475  
 FLAGS mkr: 652  
 FLAGS/MULTIPLE mkr: 653  
 FLAGS/MULTIPLE/CLICK mkr: 653  
 FLAGS/SINGLE mkr: 652  
 FLAGS/SINGLE/CLICK mkr: 652  
 FLAGS/SINGLE/MULTIPLE mkr: 653  
 FLAGS/SINGLE/MULTIPLE/CLICK mkr: 654  
 Fn mkr: 475  
 FN\_# edt: 290  
 FND\_BEG edt: 356  
 FND\_BLANK edt: 355  
 FND\_END edt: 356  
 FND\_TAG\_POS edt: 388  
 FORMAT mkr: 518  
 Formatieren FO: 160  
 Formatieren von Dateien mit Tags mkr: 566  
 Formatieren von Zeichenfolgen mkr: 519  
 Formulare aufbereiten FA: 157  
 Fortsetzen einer TUSTEP-Sitzung sys: 88  
 Fortsetzungszeilen bei Editor-Definitionen  
 ED: 154  
 Fortsetzungszeilen bei Editoranweisungen  
 edt: 272  
 Fortsetzungszeilen bei Kommandos kmd: 110  
 Fortsetzungszeilen bei langen Sätzen edt: 270  
 338  
 Fortsetzungszeilen bei Parametern prm: 706  
 738  
 FREE\_DISK\_SPACE mkr: 502  
 Freier Speicherplatz mkr: 502  
 Freigabe von Variablen mkr: 419  
 FREILAUFEND PR: 209 212 mkr: 473  
 Fremd-Datei dat: 31 DA: 128 UM: 242  
 mkr: 494  
 FROM mkr: 639  
 FT\_# edt: 290  
 FTP dtr: 60  
 FULL\_NAME mkr: 497 498  
 Funktionen aufrufen/definieren/löschen/abfra-  
 gen edt: 287  
 Funktionstasten inh: 22  
 Funktionstasten kmd: 115 edt: 287  
 Funktionstasten abfragen IN: 168  
 Funktionstasten belegen DE: 133 139  
 Funktionstasten definieren/voreinstellen ED:  
 153  
 Funktionstasten löschen ED: 153  
 Funktionstasten, Voreinstellung der DE: 139  
 Funktionswert mkr: 418  
 Fußnoten EI: 156 FO: 160 SA: 221  
 FIXED mkr: 502  
 FX\_# edt: 290  
 ganze Wörter edt: 330  
 Geräte-Angabe beim Drucken sys: 80  
 GERMAN mkr: 474  
 Gestaltung des Ausdrucks FO: 160 RA: 215  
 SA: 221  
 GET\_ATTRIBUTE mkr: 561  
 GET\_TAG\_NAME mkr: 560  
 GET\_VALUE mkr: 553  
 GET\_WRD\_NUM edt: 392  
 Gib Anweisung ED: 152  
 Gib Kommando kmd: 113  
 Gib Spezifikationen kmd: 113  
 GOTO FIELD mkr: 645  
 GOTO WINDOW mkr: 645  
 GREGORIAN mkr: 504 505  
 Gregorianischer Kalender mkr: 504  
 Griechische Schrift zvr: 775  
 Groß- in Kleinbuchstaben umwandeln prm:  
 723 754  
 Groß- und Kleinbuchstaben kmd: 110 mkr:  
 463 538 539 prm: 707 719 739 750  
 Großbuchstaben mkr: 510  
 Großbuchstaben in Dateinamen dat: 29  
 Großbuchstaben, Zeichengruppen-Kennung für  
 mkr: 678 prm: 709 742  
 Gruppenkennungen edt: 297  
 Halt HA: 161  
 Häufigkeiten RA: 215 edt: 307 315  
 Häufigkeitsbedingungen prm: 717 747  
 Hauptverzeichnis dat: 25  
 Hebräische Schrift zvr: 779

hebräische Zahlen mkr: 578 580  
 HEBREW mkr: 578 580  
 HEBREW\_GADOL mkr: 580  
 Heftchen-Druck DR: 145  
 HEIGHT mkr: 592  
 HELP edt: 384 mkr: 475 646  
 HEX mkr: 577 580  
 HEX2 mkr: 577 579  
 HEX4 mkr: 577 579  
 HEX6 mkr: 577 579  
 HIDE\_MACROS edt: 383  
 HIGHLIGHT mkr: 533  
 Hilfetexte ED: 154  
 Hochformat dat: 38  
 Hochstellung zvr: 798  
 Höhe des Editor-Fensters ED: 150  
 Höhe des Editorfensters edt: 337  
 Höhe des TUSTEP-Fensters DE: 133  
 Höhe eines Feldes abfragen mkr: 592  
 Holen des Inhaltsverzeichnisses einer Segment-  
 Datei edt: 282  
 Holen einer Datei edt: 281  
 Holen eines Segments HO: 165 edt: 281  
 Holen von Daten HO: 165  
 HOME dat: 27 edt: 353 mkr: 498  
 home-directory sys: 102  
 HOST sys: 82 mkr: 506  
 HTML-Seite mkr: 429 508  
  
 IBMPC DE: 133 LI: 176 UM: 238 cdt: 817  
 IDENTIFY mkr: 534  
 Identifizieren von Zeichenfolgen mkr: 534  
 IF mkr: 437 438  
 IF\_EMPTY:name edt: 378  
 IF\_FIRST:name edt: 379  
 IF\_MATCH:name edt: 379  
 IF\_ZERO:name edt: 378  
 IGNORE edt: 398  
 IGNORE edt: 269  
 illegale Daten dtr: 59  
 Import einer Datei mkr: 639  
 Import von Daten dtr: 56  
 INC\_NUM edt: 389  
 INCLUDE mkr: 452  
 INCLUDE/BINARY/FILE mkr: 453  
 INCLUDE/BINARY/VARIABLE mkr: 453  
 INCLUDE/DATA mkr: 453  
 INDENT edt: 354  
 INDENT\_TAGS mkr: 566  
 INDEX mkr: 544  
 INDEX\_EXACT mkr: 544  
 INDEX\_SORT mkr: 572  
 INDEX\_VALUE mkr: 557  
 Informationsfeld bei Parametern prm: 706  
 738  
 Informieren → Abfragen  
 INFIX mkr: 445 447  
  
 Inhalt einer Variablen mkr: 410  
 Inhaltsverzeichnis mkr: 494  
 Inhaltsverzeichnis einer Segment-Datei dat:  
 35  
 Inhaltsverzeichnis einer Segment-Datei holen  
 edt: 282  
 INI-Datei dat: 26 sys: 75 87 88 89 98 DA:  
 131 ED: 149 edt: 334 mkr: 460  
 INIT sys: 87 89  
 INPUT mkr: 646  
 INPUT/EDIT mkr: 649  
 INPUT/SCROLL mkr: 649  
 INPUT/SHIFT mkr: 648  
 INS:text edt: 359  
 INS\_LINE edt: 276 360  
 INS\_LINE\_IND edt: 360  
 Insert kmd: 114 edt: 260 mkr: 545  
 insert mode → Einfügemodus  
 INSERT\_ATTRIBUTE mkr: 562  
 INSERT\_CB edt: 393 mkr: 648  
 INSERT\_INDEX mkr: 545  
 INSERT\_VALUE mkr: 554  
 Installation inh: 19  
 Installationsanleitung inh: 19  
 INT mkr: 478  
 Internationaler ASCII-Code cdt: 815  
 Internationaler EBCDIC-Code cdt: 838  
 Interpretationsmodi einstellen mkr: 413  
 INTERVAL mkr: 504  
 Intervall-Funktion mkr: 478  
 ISO mkr: 495 496 604 612 620  
 ISO-8859-1 sys: 75 99 LI: 176 UM: 238  
 edt: 258 cdt: 828  
 ISO-8859-13 cdt: 834  
 ISO-8859-14 cdt: 835  
 ISO-8859-15 cdt: 836  
 ISO-8859-16 cdt: 837  
 ISO-8859-2 cdt: 829  
 ISO-8859-3 cdt: 830  
 ISO-8859-4 cdt: 831  
 ISO-8859-9 cdt: 832  
 ISO-8859-10 cdt: 833  
 ISO8859 DE: 133 LI: 176 UM: 238 edt:  
 258 mkr: 576 578 579 cdt: 826  
  
 JMP\_DN DE: 140 edt: 356  
 JMP\_DN:xx edt: 356  
 JMP\_REC\_NR edt: 356  
 JMP\_UP DE: 140 edt: 356  
 JMP\_UP:xx edt: 357  
 JOIN edt: 276 398 mkr: 514 515  
 JOIN\_LINE edt: 398  
 JOIN\_TAG mkr: 565  
 JOURNAL mkr: 473  
 JULIAN mkr: 504 505  
 Julianischer Kalender mkr: 504

- Kalender, Gregorianischer mkr: 504  
 Kalender, Julianischer mkr: 504  
 Kalenderdaten, Rechnen mit KO: 172  
 Kalenderwoche mkr: 505  
 Kapitalchen zvr: 798  
 KBYTES mkr: 492  
 Kernzeichenfolge prm: 720 751  
 KEY edt: 376 mkr: 648  
 KEY\_TST edt: 396  
 KEYS mkr: 591  
 KEYWORD mkr: 415  
 Klammern mkr: 417  
 Klammern prüfen edt: 324 mkr: 569  
 Klein- in Großbuchstaben umwandeln prm:  
 723 754  
 Kleinbuchstaben mkr: 510  
 Kleinbuchstaben, Zeichengruppen-Kennung für  
 mkr: 678 prm: 709 742  
 Kolumnentitel SA: 221  
 Kommando-Manager kmd: 117  
 Kommandofolge ausführen kmd: 113 MA:  
 186 TU: 233 edt: 303 mkr: 407  
 Kommandofolge löschen FE: 159 HA: 161  
 Kommandofolge zusammenstellen MA: 186  
 TU: 233 mkr: 407  
 Kommandoname kmd: 110  
 Kommandonamen abkürzen kmd: 110  
 Kommandos ausführen mkr: 456  
 Kommandos definieren mkr: 407  
 Kommandos eingeben kmd: 113  
 Kommandos eingeben/ausgeben/korrigie-  
 ren/wiederholen kmd: 113  
 Kommandos löschen mkr: 460  
 Kommentar kmd: 110 DE: 136 141 ED:  
 154 prm: 706 738  
 Kommentare mkr: 431  
 Komprimieren von Dateien UM: 244  
 Konfiguration inh: 19  
 Kopfzeile edt: 270  
 Kopieren HO: 165 KO: 172 RE: 216 UM:  
 237 edt: 278 309  
 Kopieren im Editorfenster edt: 360 365  
 Koptische Schrift zvr: 777  
 Korrektur formatierter Texte FO: 160  
 Korrekturanweisungen ausführen KA: 171  
 Korrekturanweisungen erzeugen VE: 248  
 Korrekturanweisungen sortieren SV: 229  
 Korrigieren KA: 171 edt: 275  
 Kursiv zvr: 798  
 Kurzformen EI: 156  
 KWIC-Index RA: 215 RV: 220  
 Kyrillische Schrift zvr: 786 789
- Länge einer Zeichenfolge mkr: 510 591  
 LAST\_LINE edt: 353  
 LAST\_PAGE mkr: 493  
 last in mkr: 449  
 Latin1 cdt: 828  
 Latin10 cdt: 837  
 Latin2 cdt: 829  
 Latin3 cdt: 830  
 Latin4 cdt: 831  
 Latin5 cdt: 832  
 Latin6 cdt: 833  
 Latin7 cdt: 834  
 Latin8 cdt: 835  
 Latin9 cdt: 836  
 LATINn LI: 176 UM: 238 edt: 258  
 Laufende Nummer edt: 389 mkr: 530  
 Laufende Nummer aktualisieren NU: 206  
 Laufwerk → Träger  
 Laufwerksbuchstaben mkr: 502  
 LEER mkr: 471  
 Leere Tags ED: 153  
 Leeres Tag edt: 317 386 mkr: 559 560 617  
 Leeres Tags edt: 323  
 Leerzeichen entfernen mkr: 510  
 Leerzeichen in Makros mkr: 407  
 LEFT edt: 345  
 LENGTH mkr: 510  
 LESEN mkr: 470  
 Lesezeichen edt: 373 374 375  
 Letzte Änderung einer Datei mkr: 493  
 Letzte Änderung eines Segments mkr: 493  
 Levenshtein-Distanz mkr: 539  
 LF edt: 353 mkr: 649  
 LIFO mkr: 449  
 LIMIT mkr: 507  
 Line feed edt: 353  
 LINES mkr: 597 604  
 LINUX DE: 133 LI: 176 UM: 238 edt: 258  
 mkr: 414 cdt: 827  
 listbox mkr: 587 588 649  
 Listenfeld mkr: 587 588 649  
 LOAD mkr: 447  
 local sys: 101  
 LOCK mkr: 429 430 469 484 485 487 488 490  
 492  
 LOESCHEN mkr: 472  
 loginid → Benutzeridentifikation  
 LOOK mkr: 449  
 LOOKUP mkr: 446  
 LOOP mkr: 435 436 437  
 LOOP/CLEAR mkr: 435  
 Löschen im Editorfenster edt: 361 365  
 Löschen von Dateien dat: 38 LO: 182 mkr:  
 484  
 Löschen von Daten LO: 182 mkr: 483  
 Löschen von Kommandos mkr: 460  
 Löschen von Projekten LO: 182 mkr: 485  
 Löschen von Segmenten mkr: 484
- Labeln einer Band-Datei MBL: 197  
 landscape → Querformat  
 lange Editoranweisungen edt: 272  
 Länge der Sätze dat: 32 das: 45

- M\_DN edt: 289  
M\_LC edt: 289  
M\_LP edt: 289  
M\_LR edt: 289  
M\_MC edt: 289  
M\_RC edt: 289  
M\_RP edt: 289  
M\_RR edt: 289  
M\_UP edt: 289  
MAC mkr: 415  
MACRO mkr: 500  
MACRO\_FILE mkr: 507  
MACRO\_SEGMENT mkr: 508  
MAIN\_FILE mkr: 507  
MAIN\_SEGMENT mkr: 508  
Makedonische Schrift zvr: 787  
Makro, Beispiel mkr: 659  
Makro-Datei dat: 36 mkr: 408  
Makro-Datei abfragen IN: 167  
Makro-Datei definieren DE: 132  
Makroanweisung mkr: 407  
Makroanweisungen mkr: 413  
Makroaufruf mkr: 407  
Makrodateien abfragen mkr: 500  
Makrofenster mkr: 641  
Makrofunktionen mkr: 418 479  
Makroleisten definieren/löschen/abfragen  
edt: 291  
Makros abfragen IN: 167  
Makros aufrufen ED: 150  
Makros aufrufen/definieren/löschen/abfragen  
edt: 288  
Makros definieren DE: 132  
Makros definieren/voreinstellen ED: 153  
Makros und Makroleisten löschen ED: 153  
Makros, Beschreibung von IN: 167  
Makrovariable definieren mkr: 411  
Makrovariablen mkr: 410 420  
Makrovariablenname mkr: 410  
Marginalien SA: 221  
MARK mkr: 529  
Markieren im Editorfenster edt: 365  
Markieren von Zeichenfolgen mkr: 529 532  
533  
MATCH mkr: 447  
Mausaktionen edt: 268  
Mausrad-Verzögerung abfragen IN: 169  
Mausrad-Verzögerung einstellen DE: 134 141  
MAX mkr: 478  
MAX\_LENGTH mkr: 542  
Maximalhäufigkeit prm: 748  
Maximum berechnen mkr: 478  
MAXLEN mkr: 493  
MBYTES mkr: 492  
md (make directory) → Einrichten von Projek-  
ten  
Mega-Segment-Datei dat: 35 RE: 218  
Mehrfach-Druck DR: 145  
Meldung mkr: 431 432 433  
Meldungszeile edt: 270  
MEM\_OFF edt: 400  
MEM\_ON edt: 400  
Menü-Zeile edt: 270  
Menüs ED: 154  
MESSAGE edt: 382 mkr: 581  
MIDDLE edt: 345  
MIN mkr: 478  
MIN\_LENGTH mkr: 542  
Mindesthäufigkeit prm: 747  
Minimum berechnen mkr: 478  
MINLEN mkr: 492  
Mischen von Dateien EI: 156 MI: 201  
Mittellinie zvr: 798  
MIXED\_INDEX mkr: 571  
MIXED\_SORT mkr: 571  
mkdir (make directory) → Einrichten von Pro-  
jekten  
MOD mkr: 478  
MODE mkr: 414 415  
MODIFIED mkr: 474 493  
MODIFY ACCESS mkr: 605 614 615 623 624  
625  
MODIFY FIELD mkr: 644  
MODIFY WINDOW mkr: 642  
Modulo-Funktion → Divisionsrest  
Modus abfragen/einstellen edt: 285  
Modus einstellen ED: 151  
MOUSE edt: 345  
MOVE edt: 268  
move → Umbenennen  
MRK\_ASK edt: 367  
MRK\_BEG edt: 365  
MRK\_CB edt: 392  
MRK\_CHG:xx edt: 368  
MRK\_CON edt: 365  
MRK\_DEL\_BEG edt: 366  
MRK\_DEL\_CB edt: 392  
MRK\_DEL\_DEL edt: 361 366  
MRK\_DEL\_END edt: 366  
MRK\_DEL\_FND edt: 367  
MRK\_DEL\_INS edt: 366  
MRK\_DEL\_REP edt: 366  
MRK\_END edt: 366  
MRK\_FND edt: 367  
MRK\_IGN edt: 365  
MRK\_INI edt: 365  
MRK\_INS edt: 366  
MRK\_INS\_CB edt: 393  
MRK\_MRK edt: 369  
MRK\_REP edt: 365  
MRK\_STR:stab edt: 369  
MRK\_TAG\_DEL edt: 367  
MRK\_TST:stab edt: 369  
Multiplikation mkr: 477  
Muster zur Auswahl der Namen/Dateien LI:  
178



mv (move) → Umbenennen  
 NAME mkr: 466 506 507  
 Name abfragen IN: 168 mkr: 507  
 Name einer Makrovariablen mkr: 410  
 Name eines Makros mkr: 408  
 Name eines Segments dat: 35  
 Namen abfragen/einstellen/löschen edt: 280  
 Namen definieren sys: 74 DE: 133  
 Navigation HI: 162  
 NEU mkr: 472  
 neu nummerieren mkr: 528  
 NEUE\_SEITE PR: 209  
 NEW mkr: 472  
 NEXT\_BM edt: 374  
 NEXT\_BTAB:\* edt: 378  
 NEXT\_BTAB:name edt: 378  
 NEXT\_CR:\* edt: 377  
 NEXT\_CR:name edt: 377 395  
 NEXT\_LINE edt: 353  
 NEXT\_M\_AR:\* edt: 378  
 NEXT\_M\_AR:name edt: 378  
 NEXT\_REC edt: 354  
 NEXT\_TAB:\* edt: 377  
 NEXT\_TAB:name edt: 377  
 NEXT\_VBM edt: 374  
 nicht abgeschlossene Datei dat: 41  
 NLF mkr: 428  
 NO\_MATCH:name edt: 379  
 NOT\_EMPTY:name edt: 378  
 NOT\_FIRST:name edt: 379  
 NOT\_ZERO:name edt: 379  
 NSP mkr: 612 620  
 Nullen ergänzen mkr: 511  
 NUMBER mkr: 465 504  
 Nummerieren NU: 206 mkr: 528  
 Nummerieren im Programmmodus HO: 165  
 Nummerierung im Programmmodus dat: 33  
 ED: 151 edt: 285  
 Nummerierung im Textmodus dat: 33 ED:  
 151 edt: 285  
 OFF mkr: 472 473  
 Öffnen mit dem TUSTEP-Editor sys: 103  
 OK mkr: 471  
 OLD mkr: 472  
 ON mkr: 472 473  
 Online-Hilfe HI: 162 edt: 271  
 OPEN mkr: 481  
 Optionen edt: 333  
 Optionen definieren/voreinstellen ED: 154  
 Optionen, Voreinstellung der edt: 333  
 Optionsfeld mkr: 652  
 OR mkr: 442  
 Ordnen → Sortieren  
 Ortsregister RV: 220  
 Osterdatum mkr: 504  
 OUTPUT mkr: 655  
 OUTPUT/CENTER mkr: 656  
 OUTPUT/CLICK mkr: 657  
 OUTPUT/SCROLL mkr: 656  
 OUTPUT/SHIFT mkr: 656  
 OUTPUT/WRAP mkr: 657  
 P mkr: 480  
 PAGE\_NR edt: 389  
 PAGE\_NR\_DEC edt: 390  
 PAGE\_NR\_INC edt: 390  
 PAGES mkr: 493 597 604  
 PALETTE mkr: 590  
 PARAMETER mkr: 472 473  
 Parameter abfragen/löschen/kopieren edt:  
 316  
 Parameter definieren/löschen edt: 312  
 Parameter definieren/voreinstellen ED: 153  
 Parameter löschen ED: 153  
 Parameter, bedingte Auswertung prm: 706  
 738  
 Parameter, Fortsetzungszeilen prm: 706 738  
 Parameter-Modi einstellen PA: 207  
 Parameter-Protokollierung PA: 207  
 Parameterart I prm: 707 740  
 Parameterart II prm: 707 740  
 Parameterart III prm: 708 740  
 Parameterart IV prm: 708 741  
 Parameterart V prm: 709 741  
 Parameterart VI prm: 712 744  
 Parameterart VII prm: 712 744  
 Parameterart VIII prm: 713 744  
 Parameterart IX prm: 714 746  
 Parameterart X prm: 721 752  
 Parameterart XI prm: 724 755  
 Parameterart XII prm: 724 756  
 Parametereingabe beenden kmd: 113  
 Parametergruppe edt: 312  
 Parameterkennung edt: 312 prm: 706 738  
 PARTITIONED mkr: 473  
 PASSWD sys: 83  
 Pasteboard kmd: 116  
 Pasteboard → Zwischenablage  
 path → Pfad  
 PEEK mkr: 449  
 permanente Datei dat: 29 DA: 128 130  
 Persische Schrift zvr: 784  
 Personenregister RV: 220  
 Pfad dat: 25 27  
 Pfad abfragen mkr: 497 498  
 Pfad anpassen mkr: 499  
 Pfade suchen edt: 319  
 Pfl kmd: 114 HI: 164 edt: 260  
 Pfo kmd: 114 HI: 164 edt: 260  
 Pfr kmd: 114 HI: 164 edt: 260  
 Pfu kmd: 114 HI: 164 edt: 260

PgDn kmd: 115 HI: 164 edt: 260 mkr: 670  
PgUp kmd: 115 HI: 164 edt: 260 mkr: 670  
Phonetische Zeichen zvr: 791  
Plus edt: 260  
pointer → Zeiger  
POP mkr: 449  
PopUp-Fenster mkr: 581  
PORT sys: 82  
Port sys: 101  
PORTIONIERT PR: 209 212 mkr: 473  
portrait → Hochformat  
pos, Satzposition edt: 328  
Pos1 kmd: 114 HI: 164 edt: 260 mkr: 670  
PREV\_BM edt: 374  
PREV\_REC edt: 353  
PREV\_VBM edt: 374  
PREVIEW mkr: 640  
preview → Druckausgabe ins TUSTEP-Fenster  
PRINT edt: 400 mkr: 428 429 432 670 671  
PRINT/COLOR:xx mkr: 432  
PRINT/DATA mkr: 433  
PRINT/ERROR mkr: 432 434  
PRINT/FILE mkr: 433  
PRINT/WARNING mkr: 432  
PRINTER mkr: 466 506  
PROGRAM mkr: 428 604  
Programm abfragen mkr: 498  
Programm-Datei dat: 31 35 TU: 233  
Programmabbruch dat: 40 FE: 158  
Programme ausführen mkr: 457  
Programmmodus, Nummerierung im dat: 33  
ED: 151 edt: 285  
PROJECT mkr: 469 470 490 506  
PROJECT\_NAME mkr: 469 497  
PROJECT\_NAMES mkr: 501  
PROJEKT DA: 128 mkr: 469 470  
Projekt dat: 25 27  
Projekt, Voreinstellung für sys: 74  
Projekte einrichten DA: 128 mkr: 479  
Projekte löschen LO: 182 mkr: 485  
Projekte umbenennen AE: 122 mkr: 486  
PROJEKTNAME mkr: 469  
Projektname, aktueller dat: 26  
Projektname, ursprünglicher dat: 26  
Projektnamen dat: 25  
Projektnamen abfragen IN: 168 LI: 174  
mkr: 497 501 506  
Projektnamen ändern AE: 122  
Projektnamen definieren DE: 132  
Projektnamen ergänzen dat: 26  
PROTOKOLL mkr: 473  
Protokoll PR: 209  
Protokoll-Ausgabe unterdrücken PR: 209  
Protokoll-Datei dat: 36  
Protokoll-Datei drucken DR: 142  
Protokollieren der Makroanweisungen mkr:  
459  
Protokollierung der Parameter PA: 207  
Prüfen von Austausch-Tabellen mkr: 442  
Prüfen von Dateien mkr: 482  
Prüfen von Daten KO: 172  
Prüfen von Klammern edt: 324 mkr: 569  
594  
Prüfen von Projektnamen, Dateinamen und Da-  
teien mkr: 469  
Prüfen von Recherchier-Tabellen mkr: 442  
Prüfen von Stringgruppen mkr: 444  
Prüfen von Such-Tabellen mkr: 442  
Prüfen von Tags edt: 317 mkr: 568  
Prüfen von Zeichenfolgen mkr: 539  
Prüfen von Zeichenfolgen mit Hilfe von Schlüs-  
selwörtern mkr: 465  
Prüfen von Zeichenfolgen mit Hilfe von Tabel-  
len mkr: 467  
Prüfen von Zeichengruppen mkr: 444  
PUSH mkr: 449  
PUT\_WRD\_NUM edt: 392  
PIXEL mkr: 507  
  
Quellen-Information mkr: 507  
Querformat dat: 38  
Querverweise → Verweise  
QUESTION mkr: 581  
QUIET mkr: 446 448 639 640  
Quoted-Printable-Codierung UM: 244  
QUOTES mkr: 513  
  
R\_TABLE mkr: 440 442 466  
radio button mkr: 652  
RAMDISK mkr: 502  
RAN dat: 31 33 34 das: 51 DA: 128 mkr:  
471 480  
RAN-Datei DA: 128  
Randausgleich FO: 160 SA: 221  
Randbedingung → Umgebungsbedingung  
random Datei DA: 128  
RANDOM\_NUMBERS mkr: 575  
Randzeichenfolge prm: 720 751  
RAW mkr: 603 606 608  
RAW mkr: 603  
rd (remove directory) → Löschen von Projekten  
RD\_NUM edt: 389  
RD\_REC\_NR edt: 389  
RD\_TAG\_NAME edt: 387  
READ mkr: 470 481 482 491 496 597 606 607  
608 615 623 632 633 634  
READ/CHECK mkr: 607 633  
READ/EXIT mkr: 615 624  
READ/IGNORE mkr: 607  
READ/NEXT mkr: 606 633  
READ/PREVIOUS mkr: 606 633  
READ/QUIET mkr: 624  
READ/RAW mkr: 603  
READ/RAW/REVERSE mkr: 603

READ/RECORDS mkr: 603  
 READ/STREAM mkr: 611 620  
 READ/STRUCTURES mkr: 631  
 REARRANGE mkr: 487  
 REC\_NR edt: 389  
 REC\_NR\_OFF edt: 395  
 REC\_NR\_ON edt: 395  
 RECALL DE: 140 mkr: 447  
 RECALL\_CB edt: 393  
 RECALL\_CHAR edt: 362  
 RECALL\_DEL edt: 372  
 RECALL\_MOD edt: 373  
 RECALL\_MRK edt: 372  
 Rechenanweisungen mkr: 418 477  
 Rechenoperationen KO: 172  
 Rechenzeit ZE: 251  
 Recherchier-Tabelle edt: 332 prm: 724 756  
 Recherchier-Tabellen definieren mkr: 440  
 Recherchier-Tabellen freigeben mkr: 442  
 Recherchier-Tabellen prüfen mkr: 442  
 Rechnen mit Kalenderdaten KO: 172  
 Rechnername abfragen mkr: 506  
 RECLEN mkr: 492  
 RECO sys: 88 89  
 record → Satz  
 RECORDS mkr: 492 597 604 612 620  
 RECORDS mkr: 603  
 REDUCE mkr: 550  
 REDUCE\_INDEX mkr: 550  
 REFRESH edt: 399 mkr: 670 671  
 Register aufbereiten RA: 215  
 Register sortieren → Sortieren  
 Register vorbereiten RV: 220  
 Registerinträge erzeugen RV: 220  
 Reihenfolge der Spezifikationen kmd: 110  
 mkr: 451  
 RELEASE mkr: 442 443  
 REMOTE sys: 75 101 mkr: 474 502  
 REMOVE mkr: 423 424 546  
 remove → Löschen  
 REMOVE/SYSTEM mkr: 425  
 REMOVE/TUSTEP mkr: 423  
 REMOVE/VOLUME mkr: 420  
 REMOVE\_ATTRIBUTE mkr: 563  
 REMOVE\_ATTRIBUTES mkr: 564  
 REMOVE\_TAGS mkr: 565  
 REMOVE\_VALUE mkr: 555  
 REMOVEABLE mkr: 502  
 RENAME mkr: 486 487  
 rename → Umbenennen  
 RENAME\_ATTRIBUTE mkr: 563  
 RENUMBER mkr: 528  
 Reorganisieren von Dateien dat: 40  
 REP:text edt: 359  
 REPAIR mkr: 489  
 REPEAT mkr: 513  
 REPL\_ABBR edt: 377  
 REPLACE mkr: 546 640  
 replace mode → Überschreibmodus  
 REQUEST mkr: 429 508  
 RESET mkr: 448  
 RESHOW edt: 399  
 REST\_CMD edt: 398  
 REST\_CUR edt: 357  
 REST\_MRK edt: 371  
 REST\_REC\_NR edt: 391  
 RETRY edt: 376  
 Retten einer Datei edt: 282 283  
 Retten eines Segments RE: 216 edt: 283  
 Retten von Dateien dat: 40  
 Retten von Daten RE: 216  
 Return kmd: 114 HI: 163 edt: 260 mkr:  
 450 452 669  
 REVERSE edt: 355 367 mkr: 553 603  
 REVERSE mkr: 603  
 RIGHT edt: 345  
 RLM UM: 241  
 rm (remove) → Löschen  
 rmdir (remove directory) → Löschen von Pro-  
 jekten  
 ROMAN mkr: 465 578 580  
 römische Zahlen mkr: 578 580  
 root directory → Hauptverzeichnis  
 ROWS mkr: 507  
 RST\_CR edt: 395  
 RST\_NUM edt: 389  
 RST\_REC\_NR edt: 392  
 rtab edt: 332  
 Russische Schrift zvr: 786  
  
 S\_DN edt: 289  
 S\_GROUP mkr: 443 444 466  
 S\_LP edt: 289  
 S\_LR edt: 289  
 S\_RC edt: 289  
 S\_RP edt: 289  
 S\_RR edt: 289  
 S\_TABLE mkr: 440 442 466  
 S\_UP edt: 289  
 Sachregister RV: 220  
 Sammeln von Zeichenfolgen mkr: 543  
 Satz dat: 32 das: 45  
 Sätze löschen edt: 276 277 309  
 Satzherstellung SA: 221  
 Satzlänge dat: 32 das: 45 46  
 Satzlänge abfragen mkr: 492 493  
 Satznummer dat: 32 33 das: 52 ED: 152  
 edt: 271 389 mkr: 598 603  
 Satznummernkonflikt dat: 33 das: 52  
 Satzposition edt: 391  
 Satzposition pos edt: 328  
 Satzzahl MBI: 191  
 Satzzahl abfragen LI: 178 mkr: 492  
 SAVE\_CMD edt: 397  
 SAVE\_CUR edt: 357

SAVE\_MRK edt: 371  
 SAVE\_REC\_NR edt: 391  
 Schaltfläche mkr: 654  
 Schleifen TU: 234 mkr: 435  
 SCHREIBEN mkr: 471  
 Schreiberlaubnis edt: 272  
 Schrift, Alt-kirchenslavische zvr: 789  
 Schrift, Arabische zvr: 782  
 Schrift, Bulgarische zvr: 787  
 Schrift, Cyrillische zvr: 786 789  
 Schrift, Griechische zvr: 775  
 Schrift, Hebräische zvr: 779  
 Schrift, Koptische zvr: 777  
 Schrift, Kyrillische zvr: 786 789  
 Schrift, Makedonische zvr: 787  
 Schrift, Persische zvr: 784  
 Schrift, Russische zvr: 786  
 Schrift, Serbische zvr: 787  
 Schrift, Syrische zvr: 781  
 Schrift, Ukrainische zvr: 788  
 Schrift, Weißrussische zvr: 788  
 Schriftumschaltungen zvr: 798  
 Schriftumschaltungen ignorieren edt: 330  
 Schwester-Dateien dat: 37  
 SCP dtr: 60  
 SCR\_DN edt: 358  
 SCR\_DN\_BOTH edt: 358  
 SCR\_UP edt: 358  
 SCR\_UP\_BOTH edt: 358  
 SCRATCH mkr: 471 500  
 Scratch-Datei dat: 29 AB: 121 DA: 130  
 Scratch-Dateien abfragen LI: 173  
 Scrollen in Dateien edt: 357  
 SCROLLING mkr: 473  
 SEARCH mkr: 536  
 SEARCH\_ALL mkr: 537  
 SECTION mkr: 450 466  
 SEGM\_FILE edt: 390  
 SEGM\_NAME edt: 390  
 SEGMENT mkr: 429 471 495  
 Segment ausführen TU: 233  
 Segment einer Datei mkr: 494 495  
 Segment ersetzen edt: 283  
 Segment holen HO: 165 edt: 281  
 Segment kopieren edt: 278  
 Segment löschen edt: 283 284  
 Segment retten RE: 216 edt: 283  
 Segment-Datei dat: 31 35 kmd: 113 RE:  
     218 TU: 233 edt: 281 282 283 mkr: 407  
     493 495  
 Segment-Datei, Inhaltsverzeichnis einer dat:  
     35  
 Segment-Dateien neu nummerieren dat: 35  
 Segment-Dateien zusammenfassen dat: 35  
     RE: 219  
 Segment-Titel edt: 286 mkr: 429  
 SEGMENT/ERASE mkr: 429  
 SEGMENT/PRINT mkr: 429  
 SEGMENT/TITLE mkr: 429  
 SEGMENT\_MRK edt: 371  
 Segmente abfragen LI: 173  
 Segmente löschen mkr: 484  
 Segmente sortieren dat: 35  
 Segmente umbenennen mkr: 487  
 Segmentname dat: 35 mkr: 429  
 Segmentnamen abfragen edt: 280 mkr: 494  
 Segmentnamen einstellen edt: 280  
 Segmentnamen löschen edt: 278 280  
 Seiten-Auswahl DR: 144  
 Seitennummer dat: 32 33 das: 46 edt: 328  
     389  
 Seitennummer abfragen mkr: 493  
 Seitenumbruch FO: 160 SA: 221  
 Seitenzahl abfragen mkr: 493  
 Sektionen mkr: 450  
 SELECT mkr: 439 547 649  
 SELECT/EXACT mkr: 439  
 SELECT/MULTIPLE mkr: 651  
 SELECT/MULTIPLE/CLICK mkr: 651  
 SELECT/SINGLE mkr: 649  
 SELECT/SINGLE/CLICK mkr: 651  
 SELECT/SINGLE/MULTIPLE mkr: 651  
 SELECT/SINGLE/MULTIPLE/CLICK mkr: 652  
 SELECT:name edt: 384  
 SELECT\_ABBR edt: 377  
 SELECT\_BM edt: 374  
 SELECT\_CHAR edt: 399  
 SELECT\_TAG edt: 385  
 SELECT\_TEXT edt: 359  
 SELECT\_VBM edt: 375  
 SEND\_MAIL mkr: 508  
 Senden an sys: 103  
 SEPARATE mkr: 548  
 SEPARATE\_EXACT mkr: 549  
 SEQ dat: 31 33 34 das: 51 DA: 128 mkr:  
     471 480  
 SEQ-Datei DA: 128  
 sequentielle Datei DA: 128  
 Serbische Schrift zvr: 787  
 Serienbriefe EI: 156  
 SESSION mkr: 458 507  
 SET mkr: 418 419  
 SET/DATA mkr: 419  
 SET\_ATTRIBUTE mkr: 561  
 SET\_FND:text edt: 367  
 SET\_INS edt: 359  
 SET\_MATCH edt: 400  
 SET\_MRK:text edt: 370  
 SET\_REP edt: 359  
 SET\_TAG\_NAME mkr: 560  
 SET\_TAG\_POS edt: 387  
 SET\_VALUE mkr: 554  
 Setzen von Attributen mkr: 561  
 SHELL sys: 83  
 Shift kmd: 114 edt: 259 mkr: 669  
 SHORTEN mkr: 510 511

SHOW mkr: 432  
 SHOW\_MACROS edt: 383  
 SHOW\_TAGS edt: 388  
 SHW\_BEG edt: 382  
 SHW\_CUR edt: 382  
 SHW\_DN edt: 357 mkr: 647 650 671 680 683  
     687  
 SHW\_END edt: 382  
 SHW\_END\_TAG edt: 388  
 SHW\_STRT\_TAG edt: 388  
 SHW\_UP edt: 357 mkr: 647 650 671 680 683  
     687  
 SIGNAL mkr: 472  
 Signalton SI: 222 mkr: 434  
 Silbentrennung FO: 160 SA: 221  
 Sitzungsnummer abfragen mkr: 507  
 SIZE mkr: 448 450 541  
 SKIP mkr: 608 634  
 SKP\_BEG edt: 354 mkr: 647 650 671 672  
     680 683 686  
 SKP\_END edt: 355 mkr: 647 650 671 672  
     680 683 686  
 SKP\_LE edt: 355 mkr: 647  
 SKP\_RI edt: 355 mkr: 647  
 SKP\_WORD edt: 355 mkr: 687  
 SLEEP → WAIT  
 Sonder-Sortierfolge mkr: 463 545 570 571  
     572 cdt: 841  
 Sonderbuchstaben zvr: 766  
 Sonderzeichen zvr: 765 767 799  
 Sonderzeichen in Dateinamen dat: 29  
 Sonderzeichen, Zeichengruppen-Kennung für  
     mkr: 678 prm: 709 742  
 SOR mkr: 475 615 623  
 SORT\_ATTRIBUTES mkr: 564  
 SORT\_VALUES mkr: 556  
 SORTED mkr: 471  
 Sortieralphabet cdt: 841  
 Sortieralphabet-Tabelle prm: 712 744  
 Sortieren mkr: 570  
 Sortieren von Attributen mkr: 564  
 Sortieren von Daten/Dateien SO: 224  
 Sortieren von Segmenten dat: 35  
 Sortieren von Wertelisten mkr: 556  
 Sortieren von Zahlen mkr: 571  
 Sortieren von Zeichenfolgen mkr: 570 571  
 Sortieren vorbereiten SV: 229  
 Sortierfolge cdt: 841  
 Sortierschlüssel prüfen SP: 227  
 SOURCE mkr: 507  
 Spaltenbegrenzung begr: edt: 329  
 Spaltenzahl abfragen IN: 168 mkr: 507  
 Spaltenzahl einstellen DE: 134 ED: 150  
 Speicherplatz, Belegter mkr: 502  
 Speicherplatz, Freier mkr: 502  
 Sperren der Daten dat: 41  
 Sperren von Dateien dat: 41 mkr: 490  
 Sperrung zvr: 798  
 Spezifiaktionsnamen abkürzen kmd: 110  
 Spezifikationen kmd: 110 mkr: 413  
 Spezifikationen, Reihenfolge der kmd: 110  
     mkr: 451  
 Spezifikationen, Voreinstellung für kmd: 110  
     mkr: 413  
 Spezifikationsnamen kmd: 110  
 Spezifikationswerte kmd: 110  
 SPLIT edt: 276 360 mkr: 449 516 517  
 SPLIT\_TAG mkr: 564  
 SQUEEZE mkr: 510  
 stab edt: 331  
 STACK mkr: 449 450 466  
 Stammverzeichnis → Hauptverzeichnis  
 Standard-Dateien dat: 30  
 Standard-Daten-Datei dat: 30  
 Standard-Editor-Datei dat: 30 36 ED: 149  
 Standard-Protokoll-Datei dat: 30  
 Standard-Sortierfolge mkr: 463 570 571 572  
     cdt: 841  
 Standard-Text-Datei dat: 30  
 Stapelspeicher mkr: 448  
 START mkr: 458  
 START\_TAG mkr: 559  
 Starten des TUSTEP-Editors sys: 103 104 105  
     mkr: 458  
 Starten einer TUSTEP-Sitzung sys: 80  
 Starten einer TUSTEP-Sitzung mkr: 458  
 Starten von Anwendungen edt: 394 mkr:  
     458  
 Starten von TUSTEP sys: 79 89 98  
 STATEMENT mkr: 414  
 STATUS edt: 382  
 Statuszeile edt: 270  
 Sternvariablen mkr: 411  
 Steuerbefehle edt: 346 350 351  
 Steuerzeichen ändern mkr: 417  
 STOP edt: 401 mkr: 460  
 STREAM mkr: 597  
 STREAM mkr: 611 620  
 Stringgruppen prm: 709 710 741 743  
 Stringgruppen definieren mkr: 443  
 Stringgruppen definieren/löschen/abfragen  
     edt: 296  
 Stringgruppen definieren/voreinstellen ED:  
     153  
 Stringgruppen freigeben mkr: 443  
 Stringgruppen löschen ED: 153  
 Stringgruppen prüfen mkr: 444  
 STRINGS mkr: 522  
 STRUCTURE mkr: 627  
 STRUCTURES mkr: 597  
 STRUCTURES mkr: 631  
 strukturierte Daten edt: 312  
 Strukturname mkr: 627  
 STZ zvr: 761  
 SUBMACRO mkr: 451 466  
 Submakros mkr: 451

SUBSTITUTE mkr: 526 527 528  
 Subtraktion mkr: 477  
 Such-Anweisungen edt: 312  
 Such-Tabelle edt: 331 prm: 714 746  
 Such-Tabellen definieren mkr: 440  
 Such-Tabellen freigeben mkr: 442  
 Such-Tabellen prüfen mkr: 442  
 Suchbedingung edt: 312 313 mkr: 599  
 Suchen edt: 307 315  
 Suchen im Editorfenster edt: 365  
 Suchen von Tags edt: 317  
 Suchen von Zeichenfolgen mkr: 535 536 537  
 Suchzeichenfolge, Element der prm: 716 747  
 Suchzeichenfolgen LI: 179 edt: 331 332  
 prm: 711 713 715 721 743 745 746 752  
 SUM mkr: 551  
 SWITCH:erg? edt: 376  
 SWITCH:erg?name edt: 375  
 SWITCH:name edt: 375  
 SYM UM: 241  
 Syrische Schrift zvr: 781  
 SYSTEM PR: 212 mkr: 473 506  
 System-Kommando sys: 89  
 System-Kommandos ausführen mkr: 457  
 System-Variablen sys: 73 mkr: 424  
 System-Variablen abfragen LI: 174 mkr: 425  
 506  
 System-Variablen definieren mkr: 424  
 System-Variablen löschen mkr: 425  
  
 TAB edt: 354 mkr: 475 646  
 Tabellen mkr: 421  
 Tabellenname mkr: 440  
 Tabulatoren definieren/voreinstellen ED: 152  
 Tabulatoren, Voreinstellung der edt: 286  
 Tag-Attribut-Prüfungen löschen ED: 153  
 Tag-Definitionen edt: 300  
 Tag-Gruppen edt: 301  
 Tag-Namen edt: 300 317 323 387  
 Tag-Namen definieren/voreinstellen ED: 153  
 Tag-Position edt: 387 388  
 Tag-Prüfung definieren/voreinstellen ED: 153  
 Tag-Prüfung definieren/wechseln/löschen/abfragen  
 edt: 300  
 Tagesdatum edt: 390 391 mkr: 504  
 Tagesnummer mkr: 504  
 TAGS mkr: 445  
 Tags edt: 386 mkr: 465 559  
 Tags anzeigen edt: 318 320  
 Tags entfernen mkr: 565  
 Tags prüfen edt: 317 mkr: 568  
 Tags schützen edt: 299  
 Tags suchen edt: 321  
 TAGS:n edt: 399  
 TAPE DA: 128 mkr: 471 480  
 Tastenbezeichnungen kmd: 114 edt: 259  
 mkr: 669  
  
 Tastenkombinationen kmd: 115 edt: 261  
 Tastenkombinationen für Funktionsaufrufe  
 edt: 341  
 Tastenkombinationen für Makroaufrufe edt:  
 342  
 Tastenkombinationen für Steuerbefehle edt:  
 346 351  
 Teilzeichenfolgen mkr: 411  
 temporäre Datei dat: 29 DA: 128 130  
 TERM sys: 88 89 mkr: 460  
 TERMINATE mkr: 460  
 TEST\_MRK:stab edt: 369  
 Testhilfen mkr: 658  
 text edt: 382 mkr: 442  
 Text-Datei dat: 31 34  
 textbox mkr: 585 646  
 Textteil einfügen edt: 359 377  
 Textfeld edt: 270 285  
 Textfeld teilen edt: 285  
 Textmodus, Nummerierung im dat: 33 ED:  
 151 edt: 285  
 Textteile prm: 707 740  
 Textteile auswählen prm: 728  
 Textteile eliminieren prm: 728  
 Textteile-Austausch-Tabelle prm: 708 741  
 Textteile-Vergleichs-Tabelle prm: 708 740  
 Textversionen VA: 247 VE: 248  
 Textversionen/Varianten vergleichen VE: 248  
 t f sys: 103 104 105  
 TGL\_INS edt: 359 mkr: 648 672  
 THEN mkr: 438  
 Tiefstellung zvr: 798  
 Tilde dat: 27 mkr: 498  
 TIME mkr: 503  
 TIME\_INTERVAL mkr: 503  
 TIME\_n edt: 391 mkr: 503  
 TIMEOUT sys: 83  
 Titel mkr: 429  
 Titel einer Datei mkr: 494  
 Titel einer Datei definieren TI: 231  
 Titel einer Datei merken TI: 231  
 Titel einer Datei übertragen TI: 231  
 TITLE mkr: 467 494  
 TO mkr: 639  
 TODAY mkr: 504  
 Tonfolge FE: 158 SI: 222 mkr: 434  
 TRACE mkr: 459 460  
 TRACE\_OFF edt: 396  
 TRACE\_ON edt: 396  
 Träger dat: 25 27  
 Träger abfragen LI: 178 mkr: 499 501  
 Träger, Voreinstellung für sys: 73  
 TRY\_SPLIT:name edt: 361  
 tstp sys: 103 104 105  
 TURN mkr: 540 541  
 TUSCRIPT mkr: 415  
 TUSTEP mkr: 506  
 TUSTEP aufrufen sys: 79 89 98

TUSTEP-Code, ASCII cdt: 815  
 TUSTEP-Code, EBCDIC cdt: 838  
 TUSTEP-Datei dat: 31 DA: 128  
 TUSTEP-Fensters, Breite des DE: 134  
 TUSTEP-Fensters, Höhe des DE: 133  
 TUSTEP-Sitzung sys: 74  
 TUSTEP-Sitzung beenden sys: 88 BE: 127  
 NO: 205  
 TUSTEP-Sitzung beginnen sys: 87  
 TUSTEP-Sitzung definieren sys: 80  
 TUSTEP-Sitzung fortsetzen sys: 88  
 TUSTEP-Sitzung normieren NO: 205  
 TUSTEP-Sitzung starten sys: 80  
 TUSTEP-Sitzung starten mkr: 458  
 TUSTEP-Sitzung unterbrechen sys: 87 kmd:  
 117 BE: 127  
 TUSTEP-Variablen sys: 73 mkr: 411 422  
 TUSTEP-Variablen abfragen IN: 167 mkr:  
 422 506  
 TUSTEP-Variablen definieren DE: 132 mkr:  
 422  
 TUSTEP-Variablen löschen mkr: 423  
 TUSTEP.INI → INI-Datei  
 TUSTEP.xxx DA: 131  
 TUSTEP\_CGI sys: 75  
 TUSTEP\_CMD sys: 75  
 TUSTEP\_COLS sys: 74  
 TUSTEP\_DSK sys: 73  
 TUSTEP\_HST sys: 73  
 TUSTEP\_INI sys: 75  
 TUSTEP\_LIB sys: 75  
 TUSTEP\_LPR sys: 76  
 TUSTEP\_MDS sys: 75  
 TUSTEP\_MEM sys: 74  
 TUSTEP\_NAM sys: 74  
 TUSTEP\_PKZ sys: 76  
 TUSTEP\_PRJ dat: 26 sys: 74  
 TUSTEP\_PRN sys: 75 DR: 142 PR: 210  
 TUSTEP\_RCP sys: 76  
 TUSTEP\_ROWS sys: 74  
 TUSTEP\_SCR sys: 73 74  
 TUSTEP\_TTL sys: 73  
 TUSTEP\_UCB sys: 76  
 TUSTEP\_USR sys: 74  
 TXT\_CHG:xx edt: 400  
 TXT\_DEL:xx edt: 363  
 TYPE mkr: 492 605 614  
  
 Überschreiben im Editorfenster edt: 359  
 Überschreibmodus kmd: 117 edt: 333 339  
 359 mkr: 672  
 Überstreichung zvr: 798  
 Übertragen von TUSTEP-Dateien dtr: 59  
 Uhrzeit edt: 391 mkr: 493 502 503  
 Ukrainische Schrift zvr: 788  
 Umbenennen von Attributen mkr: 563  
 Umbenennen von Dateien AE: 122 mkr: 486  
 Umbenennen von Projekten AE: 122 mkr:  
 486  
 Umbenennen von Segmentenen mkr: 487  
 Umbruch von Texten FO: 160 SA: 221  
 Umcodiertabelle cdt: 840  
 Umdrehen von Zeichenfolgen mkr: 540 541  
 553  
 Umgebungsbedingungen prm: 720 751  
 Umgebungsvariable → System-Variable  
 Umlaute in Dateinamen dat: 29  
 Umnummerieren HO: 165 NU: 206 edt:  
 279  
 Umschlag bei Druckausgaben sys: 74 DE:  
 133 DR: 143 145 PR: 210  
 Umstellen edt: 279 310  
 Umstellen von Textteilen KO: 172 prm: 728  
 Umwandeln von Daten/Dateien UM: 237  
 Umwandeln von Groß- in Kleinbuchstaben  
 prm: 723 754  
 Umwandeln von Klein- in Großbuchstaben  
 prm: 723 754  
 UND\_BEG edt: 362 mkr: 648  
 UND\_CHAR edt: 361 mkr: 647  
 UND\_END edt: 362 mkr: 648  
 UND\_LE edt: 363  
 UND\_LINE edt: 364  
 UND\_REC edt: 364  
 UND\_RI edt: 363  
 UND\_WORD edt: 363 mkr: 648  
 Undo edt: 362 363 364  
 UNICODE LI: 176 UM: 238 edt: 258  
 UNIQUE mkr: 499  
 UNLOAD mkr: 447  
 UNLOCK mkr: 429 431 469 484 485 487 488  
 490 491 492  
 UNSET/STRUCTURE mkr: 419  
 UNSET/VARIABLE mkr: 419  
 Unterbrechen einer TUSTEP-Sitzung sys: 87  
 kmd: 117 BE: 127  
 Unterbrechung FE: 158 HA: 161 edt: 400  
 Unterlegung zvr: 798  
 Unterpunktierung zvr: 798  
 Unterscheidungsnummer dat: 32 33 edt:  
 328  
 Unterstreichung zvr: 798  
 unvollständige Datei RE: 216  
 unzulässige Dateinamen dat: 29  
 UPDATE mkr: 446 482 597 605  
 UPDATE/RECORDS mkr: 603  
 UPDATE/STRUCTURES mkr: 631  
 UPLOAD mkr: 639  
 Upload dtr: 60  
 UPTO mkr: 607 609 633 635  
 ursprünglicher Projektname dat: 26  
 US-EBCDIC-Code cdt: 839  
 USB-Speicher-Stick inh: 20 21 dtr: 56 60  
 USED mkr: 492  
 USED\_DISK\_SPACE mkr: 502

USER mkr: 442 507  
 USERID sys: 82  
 userid → Benutzeridentifikation  
 UTF8 sys: 75 99 LI: 176 UM: 238 244 245  
 edt: 258 mkr: 495 496 576 578 579 604  
 612 620  
 UTF8/UTF16 mkr: 579  
 UTF16 LI: 176 UM: 238 244 245 edt: 258  
 mkr: 495 496 576 578 604 612 620  
 UTF16/UTF8 mkr: 576  
  
 VALUE mkr: 509  
 VARIABLE mkr: 415 466 598  
 Variablen abfragen IN: 167 mkr: 422 425  
 Variablen definieren DE: 132 mkr: 418 422  
 424  
 Variablen freigeben mkr: 419  
 Variablen löschen mkr: 419 423 425  
 Variablenname mkr: 410  
 VARIABLES mkr: 506  
 Varianten VA: 247 VE: 248  
 Vergleichen fortlaufender Texte VE: 248  
 Vergleichen von Zahlen mkr: 462  
 Vergleichen von Zeichenfolgen mkr: 463 538  
 539  
 Vergleichs-Tabelle edt: 331 prm: 713 744  
 Vergleichsergebnisse aufbereiten VA: 247  
 VERIFY mkr: 539  
 Vermutlich Endlosschleife mkr: 437  
 Verschlüsseln von Daten/Dateien UM: 237  
 VERSION mkr: 506  
 Verweise aktualisieren NU: 206  
 Verweise auf Parameter prm: 706 738  
 Verweise auf Zeichen(folgen) prm: 719 722  
 750 753  
 Verzeichnis dat: 25 27 mkr: 500 501  
 Volltext EI: 156  
 VOLUME mkr: 469 499  
 VOLUMES mkr: 501  
 Vorbereiten eines Registers RV: 220  
 Vorbereiten zum Drucken DV: 148  
 Vorbereiten zum Sortieren SV: 229  
 Vordefinierte Zeichengruppen prm: 709 742  
 Voreinstellung der Editormakros edt: 289  
 Voreinstellung der Funktionstasten DE: 139  
 Voreinstellung der Optionen edt: 333  
 Voreinstellung der Tabulatoren edt: 286  
 Voreinstellung des Namens sys: 74  
 Voreinstellung für Projekt sys: 74  
 Voreinstellung für Spezifikationen kmd: 110  
 mkr: 413  
 Voreinstellung für Träger sys: 73  
 Vorrangkommandos FE: 159 HA: 161  
 Vorschubzeichen dat: 36  
 vtab edt: 331  
  
 Wahlschalter WA: 249 mkr: 473 prm: 706  
 738  
 WAIT edt: 400 mkr: 434 475  
 Warnung kmd: 111 mkr: 426 432  
 Warten mkr: 434  
 Weißrussische Schrift zvr: 788  
 Wert einer Variablen mkr: 410  
 Werteliste mkr: 553  
 Wertelisten ergänzen mkr: 554 556  
 Wertelisten sortieren mkr: 556  
 Wertzuweisungen mkr: 418  
 WHICH\_MACRO edt: 383  
 WHICH\_SEGM edt: 390  
 WHILE mkr: 607 609 633 635  
 WIDTH mkr: 592  
 Wiederholen von Zeichenfolgen mkr: 513  
 wild cards → Muster zur Auswahl von Namen  
 WINDOW mkr: 641 671  
 WINDOWS mkr: 414  
 WINRMT mkr: 474  
 Wischen/Überschreiben von Daten WI: 250  
 Wochentag edt: 390 mkr: 502 503  
 WORD mkr: 441  
 Wörter entfernen mkr: 511  
 Wörterbuch mkr: 445  
 Wortformenregister RA: 215 RV: 220  
 Wortposition edt: 392  
 WR\_NUM edt: 389  
 WR\_REC\_NR edt: 389  
 WR\_TAG\_NAME edt: 387  
 WRITE mkr: 471 481 482 491 495 597 605  
 613 622 632  
 WRITE/ADJUST mkr: 605 614  
 WRITE/BREAK mkr: 614  
 WRITE/CLEAR mkr: 605 614 623  
 WRITE/LAST mkr: 614 622  
 WRITE/NEXT mkr: 605 614 622 632  
 WRITE/RAW mkr: 603  
 WRITE/RECORDS mkr: 603  
 WRITE/STREAM mkr: 612 620  
 WRITE/STRUCTURES mkr: 631  
 WRITE/UPDATE mkr: 605 632  
 WSn mkr: 473  
 WWW-Server sys: 98 101 mkr: 429 508  
  
 X\_MRK:atab edt: 371  
 X\_TABLE mkr: 440 442 466  
 XFER\_REC\_NR edt: 392  
 XPATH\_MRK edt: 388  
  
 ZAHL mkr: 465  
 Zahlen sortieren mkr: 571  
 Zählen von Zeichenfolgen edt: 307 315  
 mkr: 537  
 Zahlenwertbedingungen prm: 716 747  
 Zahlenwerte prm: 707 740



- Zähler mkr: 575  
 Zeichen ergänzen mkr: 511  
 Zeichen-Tabelle prm: 712 744  
 Zeichenfolgen aufteilen mkr: 516 517 519  
 Zeichenfolgen auswählen mkr: 547 548 549  
 Zeichenfolgen einfügen mkr: 527 545  
 Zeichenfolgen einschließen mkr: 532 533  
 Zeichenfolgen entfernen mkr: 523 525 546  
 548 550  
 Zeichenfolgen ersetzen mkr: 526 527 528 530  
 531 546  
 Zeichenfolgen extrahieren mkr: 519 521 522  
 Zeichenfolgen formatieren mkr: 519  
 Zeichenfolgen identifizieren mkr: 534  
 Zeichenfolgen markieren mkr: 529 532 533  
 Zeichenfolgen prüfen mkr: 539  
 Zeichenfolgen sammeln mkr: 543  
 Zeichenfolgen schützen edt: 299  
 Zeichenfolgen sortieren mkr: 570 571  
 Zeichenfolgen suchen mkr: 535 536 537  
 Zeichenfolgen umdrehen mkr: 540 541 553  
 Zeichenfolgen vergleichen mkr: 538 539  
 Zeichenfolgen verkürzen mkr: 510 511  
 Zeichenfolgen vervollständigen mkr: 512  
 Zeichenfolgen wiederholen mkr: 513  
 Zeichenfolgen zählen edt: 307 315 mkr: 537  
 Zeichenfolgen zentrieren mkr: 511  
 Zeichenfolgen zusammenfassen mkr: 551  
 Zeichenfolgen zusammenhängen mkr: 513  
 514 515 543  
 Zeichenfolgruppen → Stringgruppen  
 Zeichengruppen prm: 709 710 741 742  
 Zeichengruppen definieren mkr: 443  
 Zeichengruppen definieren/löschen/abfragen  
 edt: 296  
 Zeichengruppen definieren/voreinstellen ED:  
 152  
 Zeichengruppen freigeben mkr: 443  
 Zeichengruppen löschen ED: 152  
 Zeichengruppen prüfen mkr: 444  
 Zeichengruppen und Stringgruppen prm: 716  
 Zeichengruppen, Vordefinierte prm: 709 742  
 Zeichengruppen-Kennung für ASCII-Zeichen  
 prm: 709  
 Zeichengruppen-Kennung für beliebige Zeichen  
 prm: 742  
 Zeichengruppen-Kennung für Buchstaben  
 mkr: 678 prm: 709 742  
 Zeichengruppen-Kennung für Großbuchstaben  
 mkr: 678 prm: 709 742  
 Zeichengruppen-Kennung für Kleinbuchstaben  
 mkr: 678 prm: 709 742  
 Zeichengruppen-Kennung für Sonderzeichen  
 mkr: 678 prm: 709 742  
 Zeichengruppen-Kennung für Ziffern mkr:  
 678 prm: 709 742  
 Zeichenvorrat edt: 259  
 Zeigeformat edt: 312 314  
 Zeigen edt: 275 307  
 Zeiger das: 46  
 Zeigertabelle das: 51  
 Zeilennummer dat: 32 33 das: 46 edt: 328  
 zeilensynoptisch Drucken VA: 247  
 Zeilenumbruch FO: 160 SA: 221  
 Zeilenzahl abfragen IN: 168 mkr: 507  
 Zeilenzahl einstellen DE: 133 ED: 150  
 Zeitabfrage ZE: 251  
 Zeitlimit mkr: 482 483 496 598 604 612 620  
 631  
 Zeitraum mkr: 503  
 Zeitraum berechnen mkr: 504  
 Zentrieren von Zeichenfolgen mkr: 511  
 zerstörte Daten dtr: 59  
 ZIFFERN mkr: 465  
 Ziffern, Zeichengruppen-Kennung für mkr:  
 678 prm: 709 742  
 ZIRKUMFLEX UM: 241  
 Zirkumflex prm: 739  
 Zufallszahlen mkr: 575  
 Zusammenfassen von Sätzen edt: 276 313  
 Zusammenfassen von Zeichenfolgen mkr: 551  
 Zusammenhängen von Zeichenfolgen mkr:  
 513 514 515 543  
 Zweitprotokoll PR: 209  
 Zwischenablage dtr: 55 kmd: 116 edt: 258  
 262 274 392 393 mkr: 425 648

## Neuerungen in TUSTEP Version 2020

gegenüber Version 2018 (Übersicht)

Stand: 12.6.2020

### Tastenkombinationen auf Kommando-Ebene

Ctrl+G	Zeigt in einem Fenster die letzten eingegebenen Zeilen an	{115}
--------	---	-------

### Kommando-Manager

{117}

Erlaubt u.a., aus den mit Ctrl+G oder einer anderen entsprechend definierten Funktionstaste angezeigten Eingabezeilen eine auszuwählen und unverändert oder abgeändert noch einmal einzugeben.

### #DEFINIERE

Fn=RECALL	Damit werden die gemerkten Eingaben nicht mehr im Ablaufprotokoll aufgelistet, sondern in einem Fenster angezeigt (siehe oben "Kommando-Manager").	{140}
-----------	--	-------

### Editor

#### Suchen und Prüfen von Tags

g,tpn	zeigt an, welche Tag-Namen als zulässig definiert sind.	{323}
-------	---	-------

g,tpl und g,tsl	zeigt an, ob und ggf. welche leeren Tags definiert sind.	
-----------------	--	--

#### Steuerbefehle / Aktuelles Datum

DATE_4	Datum in der Form xxxx-xx-xx	{391}
--------	------------------------------	-------

### MAKROS

#### Freigabe von Variablen

\$\$ UNSET/VARIABLE	Freigabe der angegebenen Variablen	{419}
\$\$ UNSET/STRUCTURE	Freigabe aller Variablen, die in der angegebenen Struktur aufgeführt sind.	

#### Anfragen

\$\$ ASK/SAVE	Die Option SAVE nimmt die Antwort auf die Anfrage in die Liste der auf Kommandoebene eingegebenen Zeilen auf.	{426}
---------------	---	-------

#### Datenausgabe in Dateien

\$\$ FILE "dateiname" = variablenname	UTF16 als weitere Option für Code. BOM als weitere Option unterdrückt die Byte-Order-Mark am Dateianfang.	{428}
---------------------------------------	--	-------

\$\$ FILE/ERASE/BINARY "dateiname" = REQUEST (url, query)	ruft eine HTML-Seite von einem WWW-Server	{429}
---	---	-------

ab oder ruft ein CGI-Script auf einem WWW-Server auf. Die Antwort des WWW-Servers wird in die angegebene Datei geschrieben.

### Schleifen

\$\$ LOOP variable = Anfangswert TO Endwert STEP Schrittweite {436}  
 ist eine Alternative zu der Anweisungsform, bei der statt TO und STEP jeweils Kommata stehen.

### Stapelspeicher

\$\$ STACK Beim Lesen von einem leeren Stapelspeicher {450}  
 verhindert die Option QUIET (wie bisher schon die Option EXIT beim Lesen innerhalb von Schleifen), dass eine Fehlermeldung ausgegeben und das Fehlerflag gesetzt wird.

### Makrofunktionen zur Dateiverwaltung

COMPARE (name1, name2) {489}  
 drei statt nur zwei Funktionswerte, wenn die Daten verglichen werden konnten:  
 - "YES", wenn die Daten und die Satznummern identisch sind,  
 - "DATA", wenn nur die Daten identisch sind,  
 - "NO", wenn die Daten nicht identisch sind.

### Makrofunktionen zur Dateisperre

LOCK (name, modus, limit) {490}  
 limit ist eigenes, durch Komma statt durch Schrägstrich von modus abgetrenntes Argument.

### Makrofunktionen für Dateiinhalte

FILE (name, auswahl, anzahl, code) {494}  
 Das neue Argument CODE erspart bei Fremddateien die Umwandlung der Daten in die TUSTEP-Codierung mit DECODE. Vorgesehen sind: ISO, UTF8, UTF16, BINARY oder "-" (ohne Anführungszeichen) oder BINARY/NLF.

WRITE (pfad, daten, code, limit) {495}  
 Neue Makrofunktion:  
 Schreibt den Inhalt der Variablen daten in die mit pfad angegebene Datei.

READ (pfad, daten, code, limit) {496}  
 Neue Makrofunktion:  
 Liest den Inhalt der mit pfad angegebenen Datei und speichert ihn in die Variable daten.

## Makrofunktionen für beliebige Variablen-Inhalte

FIND\_DIFF (var1, apos1, epos1, var2, apos2, epos2, char1,  
char2) {538}

Falls var1 und var2 Sternvariablen sind,  
kann nach char2 nach einem Komma noch  
eine Variable nummer angegeben werden,  
in die die laufende Nummer der Zeile  
gespeichert wird, in der ein Unterschied  
gefunden wurde.

COMBINE (werte, erg2, erg3, erg4) {552}

Beim Zusammenfassen von aufeinanderfolgenden  
Zahlenwerten können jetzt bis zu drei Variablen  
angegeben werden, die Zeichenfolgen enthalten,  
die an die jeweils erste Zahl angehängt werden.  
Der Inhalt der ersten Variable wird dort  
eingefügt, wenn nur zwei aufeinanderfolgende  
Zahlenwerte vorhanden sind; der Inhalt der  
zweiten Variablen gilt für 3, der Inhalt der  
dritten Variablen für 4 und mehr aufeinander-  
folgende Zahlenwerte.

## Makrofunktionen für Tags

RENAME\_ATTRIBUTE (tag, altername, neuename) {563}

ersetzt die bisherige Funktion  
RENAME\_ATTRIBUTE (tag, namen)

## Makrofunktion zum Decodieren

DECODE (variable, modus) {577}

Als modus ist jetzt auch HEX6 (für Zahlen,  
die mit 6 Hexadezimalzeichen dargestellt sind)  
und HEX möglich. Mit dem Modus HEX wird jeder  
Hexadezimal-Code (jeweils 2 Bytes) in das  
entsprechende Zeichen (ein Byte) umgewandelt.

## Makrofunktion zum Codieren

ENCODE (variable, modus) {578}

Als modus ist jetzt auch HEX6 (für Zahlen,  
für die 6 Hexadezimalzeichen erforderlich sind)  
und HEX möglich. Mit dem Modus HEX wird jedes  
Zeichen (jedes Byte) in den entsprechenden  
Hexadezimal-Code umgewandelt.

## Dateizugriffe, Allgemeines

ACCESS daten: modus/option "dateiname" ... {597}

Wenn die Daten durch frei wählbare Zeichenfolgen  
oder XML-Tags in Portionen eingeteilt werden  
(Option STREAM), können mit der READ-Anweisung  
auch Daten statt aus einer Datei aus einer  
Variablen gelesen werden und mit der WRITE-

Anweisung in eine Variable statt in eine Datei geschrieben werden. In diesem Fall muss nach den Optionen statt eines Dateinamens der Name der entsprechenden Variablen angegeben werden.

Ist nach `modus/option` eine Variable statt {598}  
eines (in Anführungszeichen stehenden)  
Dateinamens angegeben, so muss zu Optionen  
an erster Stelle `FILE` oder `VARIABLE` angegeben  
werden.

Der "Dateizugriff" auf eine Variable erfolgt nach den gleichen Regeln wie auf eine Fremd-Datei, jedoch kann die Angabe eines Codes entfallen, wenn die Daten nicht umcodiert werden sollen.

## #SATZ

### Parameter

SCH	neu: 32001 TusLibertineO-Heb (so genannt in <code>tustep*szschrift</code> )	{1158}
AUS	Für den Mindesteinzug in Ausgangszeilen kann zu <code>IEINZR</code> (zweiter Wert) außer 1 (= mindestens 10 Punkt Einzug) ein beliebiger Wert (z.B. 40 für mindestens 40 Punkt Einzug, falls die Satzausgabe um den Faktor 4 größer aufbereitet wird) angegeben werden.	{1170}
MAH	Sollen bei Tags mit Attributen Attributwerte unverändert in die Makro-Auflösung übernommen werden, so kann statt des Attributs ein "*" angegeben werden; in der Makro-Auflösung wird an der Stelle, an der der Wert des Attributs eingesetzt werden soll, "{=an}" angegeben, wobei für n die laufende Nummer des im Tag durch "*" ersetzten Attributs ist.	{1191}
	Beispiel: Mit dem Parameter MAH <code>&lt;page n="*" /&gt;</code> (Seite {=a1}) wird erreicht, dass aus dem Quelltext "weiter unten <code>&lt;page n="13" /&gt;</code> beschrieben." beim Satz folgender Text erzeugt wird: "weiter unten (Seite 13) beschrieben."	{1192}
	MAH <code>&lt;see kap="*" pg="*"&gt;</code> (Seite {=a2}, "{=a1}") ergibt beim Quelltext "haben wir bereits <code>&lt;see kap="Einleitung" pg="3"&gt;</code> beschrieben" in der Satzausgabe: "haben wir bereits (Seite 3, "Einleitung") beschrieben"	

Steueranweisungen

#9+ ... #9-	tiefergestellte Unterstreichung	{1258}
#49+ ... #49-	Unterpunktierung + Unterstreichung	
^&^#x00A0;	wird, wenn als 3. Wert von Parameter LAU eine 1 angegeben ist, wie #._ behandelt (non-breaking space, das aber zum Zeilenlängen-Ausgleich genutzt wird)	{1278}

Makros für die Satzumgebung#\*FUNO

KENNUNGEN	Zu KENNUNGEN können jetzt auch Zeichenfolgen für den Anfang und das Ende der Fußnoten sowie für die Rückschaltung auf Fußnoten-Nummer 1 angegeben werden, die nicht aus dem Text entfernt werden sollen (z.B. XML-Tags <fussnote> </fussnote> <ul>). Dazu muss ein "+" oder " +" hinter der betr. Zeichenfolge angegeben werden	{1345}
-----------	---	--------

#\*PS2PDF zusätzliche Spezifikationen:

BREITE=nnn	Breite der auszugebenden PDF-Seiten in Punkt
HOEHE=nnn	Höhe der auszugebenden PDF-Seiten in Punkt

#\*PSAUS {1364}

PARAMETER=	in den Parametern sind jetzt auch Kommentare möglich (sie beginnen mit drei Spatien)	{1371}
------------	--	--------

#\*PSDICKTEN

ZEICHENSATZ=HB	hebräischer Zeichensatz	{1378}
----------------	-------------------------	--------

#\*PSFONT {1381}

MODUS=HEB	Dem vorliegenden AFM-File liegt der Encoding Vector für hebräische Fonts zugrunde.  (Schriftnummern 45001 bis 45049 für hebräische Fonts reserviert)
-----------	--

#\*PSFONTVOR

ZEICHENSATZ=HEB	Zeichensatz für das Hebräische auswählen	{1383}
-----------------	--	--------

#\*TAGS {1408}

META=tag	Zur Spezifikation META angegebene Namen werden jetzt auch innerhalb anderer Elemente von der Auswertung ausgenommen
----------	---

(kommt z.B. in Sammelwerken vor, die Meta-Daten für die einzelnen Beiträge haben)

#\*TAGUEB

{1412}

MODUS=-STD- (Voreinstellung): Groß- und Kleinschreibung in den Tags wird unterschieden (wie in XML)

MODUS=- Groß- und Kleinbuchstaben in den Tags gelten als gleichwertig

#\*XMLZIEL

{1417}

MODUS=S | TS | PS wie MODUS =-STD- | T | P  
aber Silbentrennung aufheben

TAGENTF=tag Zur Spezifikation TAGENTF angegebene Tag-Namen schließen auch Elemente ein, die mit Tags mit Attributen gekennzeichnet sind.

