

### Steuerbefehle (Kommando-Ebene; Editor)

Die für das VT100-Terminal geltenden Tastenkombinationen sind nicht mehr beschrieben.

### INI-Datei

Auch unter Linux wird zu Beginn einer Sitzung, für die es noch keine INI-Datei gibt, nachgefragt, ob eine solche eingerichtet werden soll. {67}

### #DEFINIERE

Unter Linux ist die Angabe CODE=-STD- gleichbedeutend mit der Angabe CODE=LINUX (statt wie biser CODE=-). {109}

Funktionstasten auf Kommando-Ebene {114}

F6: Voreinstellung = Aufruf des Standard-Makros #\*D statt wie bisher #\*E

### #LISTE

Zur Spezifikation CODE kann jetzt auch LATIN7 bis LATIN10 (= ISO 8859-13 bis -16: Baltisch, Keltisch, Westeuropäisch, Südosteuropäisch) angegeben werden. {146}

### #UMWANDLE

Zur Spezifikation CODE kann jetzt auch LATIN7 bis LATIN10 (= ISO 8859-13 bis -16) angegeben werden. {207}

Bei MODUS = <> werden werden die Namen von Character-Entities (z.B. &auml;) auch nach Umschalten auf eine nicht-lateinische Schrift in lateinischer Schrift beibehalten. {212}

### Editor

#### Organisatorische Anweisungen:

n!,,segment benennt das Segment (nicht nur in der Editor-Datei) um {235}

h,,n           Holen des Segments mit Seitennummer n {236}

Auswählen aus Popup-Listen: {242}

(c, d?, f, i, y, gg, gg...)

Klick mit linker Maustaste oder RETURN bestätigt Auswahl.

Klick mit rechter Maustaste oder ENTER kopiert die ausgewählte Zeile in die Anweisungszeile / ins Editorfenster, wo sie ggf. modifiziert werden kann.

DEL streicht einen ausgewählten Eintrag aus der Liste.

Leertaste oder CANCEL schließt die Liste

Wird die Liste mit CANCEL geschlossen,  
so bleiben die mit DEL gestrichenen  
Einträge erhalten.

Funktionen, Funktionstasten:

f=-STD- stellt vordefinierte Funktionen wieder ein {242}  
f! löscht die Definition aller Funktionen {243}

Makros:

Neue vordefinierte Makros: {243}

Y,A=ADD\_STRT\_TAG  
Y,B=DEFINE\_BM  
Y,C=RECALL\_CB  
Y,D=SELECT\_VBM  
Y,E=ADD\_END\_TAG  
Y,F=PREV\_VBM  
Y,H=FETCH\_SEGM  
Y,I=SELECT\_BM  
Y,J=JOIN  
Y,K=RECALL\_MOD  
Y,L=RECALL\_DEL  
Y,M=RECALL\_MRK  
Y,N=INS\_LINE\_IND  
Y,O=PREV\_BM  
Y,P=REPL\_ABBR  
Y,Q=SELECT\_CHAR  
Y,R=SHW\_STRT\_TAG  
Y,S=NEXT\_VBM  
Y,T=SHOW\_TAGS  
Y,U=NEXT\_BM  
Y,V=SHW\_END\_TAG  
Y,W=WHICH\_SEGM  
Y,X=SELECT\_TEXT  
Y,Y=SELECT\_ABBR  
Y,Z=RECALL\_CHAR  
  
Y,S\_LC=SHW\_CUR

y=-std- stellt vordefinierte Makros wieder ein {244}

y! löscht die Definition aller Makros {244}

y,name.= Makros, deren Name mit einem Abkürzungspunkt {244}  
endet, sind als Abkürzungen gedacht, die mit  
REPL\_ABBR aufgelöst oder deren Auflösung  
mit SELECT\_ABBR in den Text eingefügt werden  
kann.

ggx Zeigt die letzten (bis zu 40) gemerkten {255}  
Anweisungen, die mit dem Buchstaben x beginnen

Begrenzung auf Satznummer:

z,,#,.../ sucht nur in den Satznummern {275}  
cn,m=XX,#:/.../ coloriert nur in den Satznummern

Nur ganze Wörter suchen/austauschen:

z,,\_,/.../        suchen/colorieren/austauschen        {276}  
a,,\_,/.../...    nur, wenn die gesuchte Zeichenfolge  
c,m=XX,\_\_:/.../ vorn und hinten an Worttrenner stößt.  
                  "\_" statt "\_\_": "\_" gilt nicht als  
                  Worttrenner.

Tastenkombinationen für Steuerbefehle

ALT+links        EXCH\_CHAR\_LE                                {294}  
ALT+rechts       EXCH\_CHAR\_RI  
Ctrl+A            END\_CMD\_LINE                                {295}  
Ctrl+B            kann statt "+" des Ziffernblocks betätigt werden  
Ctrl+N            JMP\_REC\_NR  
Ctrl+Y            CHG\_SETTINGS (CR, INS/REP, Scroll)  
Ctrl+Z            UND\_CHAR

Steuerbefehle

Cursor:

CUR\_UP            wenn SCROLL-Modus aktiv ist, wie SCR\_UP        {298}  
CUR\_DN            wenn SCROLL-Modus aktiv ist, wie SCR\_DN  
CUR\_RI            am Zeilenende: Sprung an Textanfang der        {299}  
                  nächsten Zeile  
CUR\_LE            am Textanfang: Sprung ans Textende der  
                  vorhergehenden Zeile  
END\_CMD\_LINE    Cursor springt ans Textende in                {299}  
                  Anweisungszeile  
INDENT            Cursor springt an Satzanfang und rückt        {300}  
                  Text so weit ein wie sichtbare Zeile  
                  davor  
JMP\_REC\_NR        Cursor springt aus dem Textfeld ins            {302}  
                  Nummernfeld der Zeile, in der er steht;  
                  aus der Anweisungszeile springt er an den  
                  Textanfang der \*-Zeile, falls diese  
                  mit Satznummer angezeigt ist

Einfügen, Überschreiben:

SELECT\_TEXT     Markierten Text merken bzw., wenn kein        {305}  
                  Text markiert ist, Anfang der gemerkten  
                  Texte anzeigen.  
                  Angezeigte gemerkte Texte können (mit  
                  Cursor+CR oder linkem Mausclick) an der  
                  aktuellen Cursor-Position ins  
                  Editor-Fenster eingefügt werden.  
INS\_LINE\_IND     wie INS\_LINE, aber einrücken (ALT+N)        {305}

## Löschen und Einfügen:

DEL_BLANK	wie DEL, falls Cursor auf Leerzeichen steht	{306}
BSP	Cursor springt vom Anfang einer Forsetzungszeile ans Textende der vorhergehenden Zeile. Steht er am Anfang eines Satzes, hängt er diesen mit dem im Textfenster vorangehenden Satz zusammen.	{306}
BSP_BLANK	wie BSP, falls Cursor rechts von Leerzeichen steht	{306}
UND_CHAR	Zuletzt mit DEL oder BSP gelöschte Zeichenfolgen vor aktueller Cursor-Position einfügen	{306}
RECALL_CHAR	zeigt mit DEL oder BSP gelöschte Zeichenfolgen zur Auswahl an	{307}

## Markieren, Kopieren, Löschen, Einfügen, Suchen

RECALL_MRK	zeigt frühere Zwischenspeicher-Inhalte zur Auswahl an	{315}
------------	--	-------

## Wiederherstellen gelöschter/geänderter Sätze:

RECALL_DEL	zeigt die Sätze, die durch Löschen im Textfenster (z.B. mit DEL_REC) in der Editor-Datei gelöscht wurden, mit dem Inhalt, den sie zuvor hatten, zur Auswahl an	{315}
RECALL_MOD	Zeigt die Sätze, die durch Ändern im Textfenster der Editor-Datei geändert wurden, mit dem Inhalt, den sie vor der Änderung hatten, zur Auswahl an.	{316}

## Lesezeichen:

DEFINE_BM	Lesezeichen auf aktuelle Satzpositon setzen bzw. auf Satz, in dem der Cursor steht	{316}
*=name	Soll ein Lesezeichen mit einem Namen definiert werden, so kann in der Anweisungszeile die Anweisung "*=name" eingegeben und mit <<ENTER>> abgeschickt werden; damit wird auf die aktuelle Satzposition ein Lesezeichen mit dem angegebenen Namen gesetzt.	
PREV_BM	zeigt Umgebung von Satz, der als nächster vor aktuellem Satz ein Lesezeichen hat	{317}
NEXT_BM	zeigt Umgebung von Satz, der als nächster nach aktuellem Satz ein Lesezeichen hat	{317}
SELECT_BM	zeigt den Anfang der Sätze mit Lesezeichen in der Textreihenfolge zur Auswahl an.	{317}
PREV_VBM	zeigt Umgebung des Satzes, dessen Lesezeichen zuletzt verwendet wurde	{317}

NEXT_VBM	(nur unmittelbar nach PREV_VBM): zeigt Umgebung des zuletzt mit PREV_VBM besuchten Satzes an	{317}
SELECT_VBM	zeigt den Anfang der Sätze mit Lesezeichen zur Auswahl an; Reihenfolge = Verwendung der Lesezeichen	{317}

## Verzweigen in andere Makros:

REPL_ABBR	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.	{319}
SELECT_ABBR	zeigt den Anfang der Makros, deren Namen mit Abkürzungspunkt endet, zur Auswahl an.	{319}
NEXT_CR:*	Abarbeiten des Makros unterbrechen, Fortführen bei nächstem CR	{320}
NEXT_TAB:name	bei nächstem TAB das Makro name ausführen	{320}
NEXT_TAB:*	Abarbeiten des Makros unterbrechen, Fortführen bei nächstem TAB	{320}
CHECK_FN	Wenn mit dem Makro FN_# eine Zeichenfolgen-Such-Tabelle definiert ist, wird eine entsprechende Zeichenfolge im Dateinamen gesucht und dann das Makro FN_n aufgerufen, wobei n die Nummer der als ersten gefundenen Zeichenfolge in der Suchtabelle ist bzw. 0 ist, falls keine Zeichenfolge gefunden wurde.	{323}
CHECK_FT	wie CHECK_FN, jedoch Suche im Dateititel	{323}
CHECH_FC	wie CHECK_FN, jedoch Suche im ersten Satz	{323}

## Anzeigen von Daten:

SHW_CUR	zeigt Sätze um die aktuelle Satzposition bzw. um die Cursor-Position	{323}
---------	--	-------

## Anzeigen von Makroleisten:

WHICH_MACRO	zeigt ggf. Name der permanenten Makroleiste	{324}
-------------	---	-------

## Anzeigen des Inhalts einer Segment-Datei:

FETCH_SEGM	zeigt Inhaltsverzeichnis der eingestellten Segment-Datei zur Auswahl an	{325}
------------	---	-------

## Tags:

ADD_STRT_TAG	Sucht im Textfenster 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.	{327}
--------------	--	-------

ADD\_END\_TAG Sucht im Textfenster 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. {327}

SHOW\_TAGS zeigt an Cursor-Position bzw. aktueller  
Satzposition noch offene Tags an {328}

SHW\_STRT\_TAG sucht rückwärts nach offenem Start-Tag {328}

SHW\_END\_TAG sucht vorwärts nach einem Ende-Tag, zu  
dem im durchsuchten Text kein  
Anfangs-Tag vorhanden ist. {328}

Dateiname und Segmentname:

SEGM\_FILE Schreibt Namen der eingestellten  
Segment-Datei {330}

SEGM\_NAME Schreibt Namen des eingestellten Segments {330}

WHICH\_SEGM zeigt ggf. Name des Segments, in dem der  
Cursor steht {330}

Aktuelle Satzposition:

RST\_REC\_NR setzt aktuelle Satznummer auf die beim  
Öffnen der Datei gültige aktuelle  
Satznummer {331}

Zwischenablage von Windows:

RECALL\_CB zeigt frühere Inhalte der Windows-  
Zwischenablage, die mit TUSTEP dorthin  
geschrieben wurden, zur Auswahl an. {333}

Carriage Return:

CHG\_SETTINGS zeigt ein Menu, mit dem Einstellungen  
geändert werden können {334}

NEXT\_CR:name beim nächsten CR soll das Makro name  
ausgeführt werden {335}

Weitere Steuerbefehle:

EXCH\_CHAR\_LE Zeichen austauschen mit links davon  
stehendem {336}

EXCH\_CHAR\_RI Zeichen austauschen mit rechts davon  
stehendem {336}

SELECT\_CHAR zeigt Sonderzeichen und Umlaute, die auf  
manchen Tastaturen fehlen oder schwer  
erreichbar sind, zur Auswahl an {338}

Makros

## Makro-Aufrufe

`#$?.$datei,...` {346}

Enthält eine Datei, die keine Segment-Datei ist, ein Makro, so kann beim Aufruf an Stelle des Segmentnamens ein Fragezeichen angegeben werden.

## Makrofenster (Eingabefelder)

`UND_CHAR` Fügt die zuletzt mit DEL und/oder BSP gelöschten Zeichen vor der aktuellen Cursor-Position ein {553}

Standard-Makros`##*CLIPBOARD`

unter Linux: Clipboard in die TUSTEP-Zwischenablage kopieren und umgekehrt

`##*D` Datei-Manager zum Einrichten, Edieren, Kopieren, Löschen, ... von Dateien

`##*DECO` Definieren von Codes für den Editor

Mit dem Makro `##*DECO` können unter Windows eigene Codes zur Anzeige der Daten im Editor definiert werden. Der Zeichenvorrat dafür kann aus dem für das TUSTEP-Fenster eingestellten Font ausgewählt werden.

Die Codes bekommen einen frei wählbaren Namen und werden in der Datei `*TUSTEP.CDS` gespeichert. Mit dem Kommando `##DEFINIERE, CODE=name` kann der jeweils gewünschte Code eingestellt werden.

`##*DESI` {63}

Bei Remote-Sitzungen kann, falls nicht der Standard-Port für SSL-Verbindungen des UNIX-Rechners verwendet werden soll, die zu verwendende Port-Nummer angegeben werden.

`##*M` Aufruf von Makros mit Hilfe einer Eingabemaske

##SATZ

## Allgemeines

## PROTOKOLL

Bei zweispaltig gesetzten Fußnoten wird in die Protokolldatei ausgegeben, wo der Wechsel auf die zweite Spalte geschieht.

Zeile "ZUSTAND" in der Protokoll-Datei vervollständigt  
- es fehlten einige am Seitenende noch eingestellte Eigenschaften wie `&!c(...)`

## Fußnoten:

Es sind jetzt bis zu 5-stellige (statt bisher: max. 4-stellige) Fußnotennummern möglich.

## Parameter

LAU: nach I1LAUF und ILIG kann jetzt ein dritter Wert `{1020}` IFENT angegeben werden. Ist dieser = 1, so werden die Character-Entities `&amp;`, `&lt;`, `&gt;`, `&apos;`, `&quot;`, die in XML für diese Zeichen in bestimmter Umgebung benutzt werden müssen, als zu setzende Zeichen `&`, `<`, `>`, `'`, `"` interpretiert.

SEI: hinter 9. Angabe im Parameter kann, durch `:"` `{1034}` getrennt, ein Zahlenwert angegeben werden, der bei Mischung römischer und arabischer Seitennummern angibt, in welcher Größe römische Seitennummern zu setzen sind; die mit Parameter GRO angegebene Größe gilt dann nur für arabischen Seitennummern.

## Zwischenüberschriften

`&l&...&9& &a&... &w&`: negativer Durchhuss möglich `{1046}`

## Makros

MAC, MAA, MAH: Bei Tags mit mehr als einem Attribut `{1054}` kann jetzt an beliebigen Stellen durch `" >` angegeben werden, dass das Tag nur bis einschl. dem davorstehen Attribut ausgewertet werden soll, wenn es nicht mit allen Attributen angegeben ist; Beispiel:

```
mah <tag >
mah <tag attr1="x">
mah <tag attr1="x" >
mah <tag attr1="x" attr2="y">
mah <tag attr1="x" attr2="y" >
```

Eine auf Stufe 1 beginnende und wieder endende Folge von `{1056}` MAH-Makros, die im Text innerhalb von anderen MAH-Folgen vorkommt, wird dort eingesetzt, wenn sie dort nicht explizit definiert ist. Eine weitergehende Schachtelung ist (noch) nicht vorgesehen.

MAZ: Voreinstellung, wenn MAH verwendet wird: `"1 -1"` `{1056}`  
Wenn MAH verwendet wird, muss 2. Wert 1, -1 oder -2 sein.

## Steueranweisungen

## Seitenumbruch

`&!S(2)` ab hier: Spaltennummer bei jedem Spaltenwechsel `{1065}`  
um 2 erhöhen

`&!S(1)` ab hier: Spaltennummer bei jedem Spaltenwechsel  
um 1 erhöhen (Normalfall)

&!Q.(n) &!Q-(n) wie &!Q. bzw. &!Q-	{1068}
&!L.(n) &!L-(n) wie &!L. bzw. &!L-	
&!N.(n) &!N-(n) wie &!N. bzw. &!N-	
aber für insgesamt n Seiten	
Mischen von ein- und mehrspaltigem Blocksatz	
&!S(n,mmm) Für nnn kann jetzt ein Bruch (z.B. 1/3)	{1073}
angegeben werden, um Bruchteile der (Grundtext)-	
Satzspiegelbreite zu bezeichnen.	
Verändern der Satzbreite	
&!S(1,mmm) Für nnn kann jetzt ein Bruch (z.B. 1/3)	{1074}
angegeben werden, um Bruchteile der (Grundtext)-	
Satzspiegelbreite zu bezeichnen.	
&!S(1,mmm) und &!S{ beenden nicht mehr wie bisher	
ggf. noch aktive Auszeichnungen	
Fußnoten	
&!FS ab hier Fußnoten seitenweise nummerieren	{1077}
&!FN ab hier Fußnoten mit Original-Nummern ausgeben	
Die Anweisungen müssen im Text und in den Fußnoten	
an den logisch entsprechenden Stellen stehn.	
Absätze	
&!A- vor \$ sorgt dafür, dass dort auch dann	{1079}
eingerückt wird, wenn Einrückung nach Überschriften	
oder Abschnitten wegfallen würde.	
Blindzeilen	
\$\$\$=n\$\$\$ Zeilenwechsel und n Leerzeilen	{1080}
\$\$\$=n.\$\$\$ Zeilenwechsel und n Punkt Freiraum	
Zeilenumbruch	
\$\$\$\\\$\$\$ unterdrückt eine unmittelbar darauf	{1092}
folgende Zeilenwechselanweisung wie \$, \$\$\$,	
\$\$\$= \$\$\$, \$\$\$-9\$\$\$= \$\$\$	
@+ auch nachfolgendes Spatium wird unterdrückt	{1094}
Einrücken	
&=\ bedeutet: nächste &=nnn-Anweisung ignorieren	{1100}
Zentrieren	
&=- (R) Folgezeilen rechts zentrieren	{1101}
@.o @..o @...o wie @.0 @..0 @...0; wenn dabei	{1101}
jedoch nur 1 Punkt erzeugt würde, wird dieser	
nicht ausgegeben	

## Merkstellen; Positionieren; Textfelder

&!P(+-) macht vorhergehende Verschiebung des Satzspiegels rückgängig {1106}

## Sperrren

#Z(nn)+...#Z- Sperrren mit nn Bildlinien {1110}

## Wahl der Druckfarbe

&!C(Hnm,prozent) Wie &C(Hnm), jedoch wird die HKS-Farbe nicht nur durch CMYK-Farben simuliert, sondern für Farbseparation vorbereitet. {1118}

## Satzzeichen, Sonderzeichen

&!B! macht unmittelbar davor stehendes Spatium für das Austreiben nicht verfügbar. {1130}

#.^ ^ Zirkumflex, Dach (nicht als Akzent)  
 ^& bei LAU 3. Wert=1: &  
 ^< bei LAU 3. Wert=1: <  
 ^> bei LAU 3. Wert=1: >  
 ^&apos; bei LAU 3. Wert=1: '  
 ^&quot; bei LAU 3. Wert=1: " (senkrecht stehend / Zeichen oktal 42 aus aktuellem PS-Font)

## Sonstige Zeichen

#[2012] (figure dash) wie \- (bisher: Rückweisung) {1131}  
 #[2015] (horizontal bar) wie ^-  
 #[201C] (left double quotation mark) " wie #.'  
 #[201D] (right double quotation mark) " wie #."  
 #[201E] (double low 9 quotation mark) „ wie #.,  
 #[2020] (dagger) † wie ^+  
 #[2022] (bullet) Aufzählungspunkt •  
 #[2026] (horizontal ellipsis) Auslassungspunkte ...  
 #[202F] Zwischenraum 125 Bildlinien (bisher: 200)

## Linie, Punktreihen

&!.(a,b,n) und &..(a,b,n) Punktlinie mit Sperrung der Punkte mit nn/50 Geviert {1136}

## Hardware-nahe Anweisungen

&!(##/nnn) Zeichen nnn aus aktuellem PS-Font {1156}

&!(#Gn/Hiiii) und &!(#Sn/Hiiii) {1157}  
 wie &!(#Gn/iii) und &!(#Sn/iii), jedoch Angabe der Grafik-Nummer als 4-stellige Hexadezimalzahl (insbes. f. Unicode-Zeichen, die als Grafiken eingebunden werden)

Makros für die Satzumgebung

## #\*AUMBRUCH

MODUS = 2 | U'2 wie modus=-STD- bzw. U, aber für {1183}  
zweispaltig gesetzte Fußnoten

FNA: für den Übergang auf 2. Spalte bei neuer Fußnote {1187}  
bzw. innerhalb einer Fußnote können jetzt zwei  
weitere Zeichenfolgen (Nr. 3 und 4) angegeben  
werden.

## #\*GRAFIK

UMFANG = + {1202}  
verhindert bei großen EPS-Dateien (mit mehr  
als ca. 500.000 Zeilen Code), dass kürzere Zeilen  
zusammengefasst werden.

## #\*PSAUS

{1214}

#\*PSAUS berücksichtigt ggf. Farbseparationsbefehle  
und baut entsprechende DSC-Comments wie  
"%DocumentCustomColors:" etc. ein.

Die Nummern der eingebundenen Grafiken und ihre  
Namen werden im Ablaufprotokoll aufgelistet, ebenso  
die Gesamtzahl der eingebundenen Grafiken.

RAHMEN=::x\*y+x\*y wie ::x\*y+x\*y, aber ohne Passkreuze {1219}

## #\*PSMONT

{1234}

Wenn erste Quelldatei leer ist, wird die zweite  
unverändert kopiert

## #\*TAGS

{1254}

Empty-element-tags werden in die mit MAH angegebenen  
hierarchischen Tags aufgenommen (bisher: MAC).

End-Tags zu über Parameter angegebenen Anfangs-Tags  
werden automatisch ergänzt, dürfen also nicht  
angegeben werden.

\* \* \* \* \*

Corrigenda zum gedruckten Handbuch 2008

Seite 205, letzte Zeile:

Statt "ggf. nach den Daten" muss es heißen:  
"ggf. vor den Daten"

Seite 336, Zeile 20:

Statt "werden die Modi VARIABLE und DATA (s.o.)"  
muss es heißen:  
"werden die Modi VARIABLE und STATEMENT (s.o.)"

Seite 464, Seitenmitte, erste Zeile nach "Beispiel:"

Statt |%<| muss es heißen: |%<<|  
also (die ganze Zeile):  
|<%|0|%/|1|%\|2|%<<|3|%:|4|

Seite 1045, 10 Zeile:

Statt "Wie \$\$-nnn\$\${" muss es heißen:  
"Wie \$\$nnn\$\${"