

Anweisung) in der Datei steht, in der zuletzt mit dem Editor ein Segment einer Segment-Datei bearbeitet wurde. Ist dies der Fall, so wird die Beschreibung des Makros aus dieser Datei statt aus der Makro-Datei gelesen. Dadurch wird erreicht, dass beim Überarbeiten eines Makros dieses nicht immer wieder (mit der Rette-Anweisung) in die Makro-Datei zurückkopiert werden muss.

(Beim Aufruf eines Makros galt eine entsprechende Regel schon in vorhergehenden Versionen, vgl. S. 408.)

#UMWANDLE

Umwandeln von TUSTEP nach UTF-16 oder UTF-8 und umgekehrt: {245}

Das für Zeichenfolgen der Form #[xxxx] Gesagte gilt jetzt auch für Zeichenfolgen der Form #[xxxxx] und #[10xxxx].

Editor

Austauschen {311}
wu

als Antwort auf die Frage, ob ausgetauscht werden soll:
Die Zeichenfolge wird ausgetauscht; es wird erst wieder nachgefragt, ob ausgetauscht werden soll, wenn eine Zeichenfolge nicht durch die selbe ausgetauscht würde (d.h. wenn durch das Austauschen eine Änderung vorgenommen würde).

Prüfen von Tags; Anzeigen von Tags (318-323) {318}

Die Behandlung von XML-Kommentaren ist verständlicher beschrieben; inhaltlich keine Änderung.

Makros

Makrofunktionen für sonstige Informationen

NAME () {507}
Aktuell mit dem Kommando #DEFINIERE (siehe Seite 130) eingestellter Name bzw. Inhalt der System-Variablen TUSTEP_NAM, falls noch kein anderer Name eingestellt wurde.

Dateizugriffe - Sätze, Zeilen, Seiten {604}

Diese Dateizugriffe erlauben das Beschreiben, Lesen und Durchsuchen von TUSTEP- und Fremd-Dateien sowie (bei den Zugriffsmodi RECORDS und STREAM) von Variablen. 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.

Neu ist, dass nicht nur von Dateien, sondern auch von Variablen gelesen werden kann.

Parameterart IX

Verweise in Suchzeichenfolgen

{720}

Neu hinzugekommen ist die Möglichkeit, bei Verweisen auf Zahlenwerte Bedingungen anzugeben, um wie viel sich die verglichenen Zahlen unterscheiden dürfen oder müssen.

- | | |
|-----------|---|
| {±n#+i} | Die Zeichen, die im zu durchsuchenden Text dem angegebenen Element der Suchzeichenfolge entsprechen, müssen eine arabische Zahl ohne Vorzeichen sein. Der Wert dieser Zahl wird mit dem Wert der an dieser Stelle im Text vorkommenden Zahl verglichen; letztere Zahl muss genau um i größer sein. |
| {±n#-i} | Wie "{±n#+i}<", jedoch muss die Zahl um i kleiner sein. |
| {±n#+i-k} | Die Zeichen, die im zu durchsuchenden Text dem Element der angegebenen Suchzeichenfolge entsprechen, müssen eine arabische Zahl ohne Vorzeichen sein. Der Wert dieser Zahl wird mit dem Wert der an dieser Stelle im Text vorkommenden Zahl verglichen; letztere Zahl muss mindestens um i und darf höchstens um k größer sein. |
| {±n#+i+k} | Ist gleichbedeutend mit {±n#+i-k} |
| {±n#-i-k} | Die Zeichen, die im zu durchsuchenden Text dem Element der angegebenen Suchzeichenfolge entsprechen, müssen eine arabische Zahl ohne Vorzeichen sein. Der Wert dieser Zahl wird mit dem Wert der an dieser Stelle im Text vorkommenden Zahl verglichen; letztere Zahl muss mindestens um i und darf höchstens um k kleiner sein. |
| {±n#-i+k} | Die Zeichen, die im zu durchsuchenden Text dem angegebenen Element der Suchzeichenfolge entsprechen, müssen eine arabische Zahl ohne Vorzeichen sein. Der Wert dieser Zahl wird mit dem Wert der an dieser Stelle im Text vorkommenden Zahl verglichen; letztere Zahl darf höchstens um i kleiner und höchstens um k größer sein. |

Beispiele

{721}

"|{#}:{+1#+0}|" bezeichnet eine Suchzeichenfolge, die rechts von einem Doppelpunkt eine Zahl mit dem gleichen Wert wie links von diesem Doppelpunkt enthält (z.B. 123:123, aber auch 0123:123 und 123:00123, jedoch nicht 123:123 von 123:1234); die Zeichenfolge besteht aus einer beliebigen Zahl ("#{#}"), einem Doppelpunkt (":") und einer dem Wert nach gleichen Zahl wie die Zahl vor dem Doppelpunkt ("#{+1#0}").

Parameterart X

Verweise (in Ersatzzeichenfolgen)

{725}

Wenn die beim Austauschen aus dem Text zu übernehmenden Zeichen einer arabischen Zahl ohne Vorzeichen entsprechen, kann diese Zahl auch um einen bestimmten Wert erhöht oder erniedrigt und an dieser Stelle eingesetzt werden.

{±n#+i}	Ist gleichbedeutend mit "{±n-n#+i}"
{±n-m#+i}	Die Zeichen, die im zu durchsuchenden Text den angegebenen Elementen der Suchzeichenfolge entsprechen, müssen eine arabische Zahl ohne Vorzeichen sein. Beim Austauschen wird der Wert dieser Zahl um i erhöht und die Zahl an dieser Stelle in den Text eingesetzt. Enthalten die angegebenen Elemente andere Zeichen als Ziffern, so werden die Zeichen, die den angegebenen Elementen entsprechen, an dieser Stelle unverändert in den Text eingesetzt.
{±n#-i}	Ist gleichbedeutend mit "{±n-n#-i}"
{±n-m#-i}	Wie "{±n-m#+i}", jedoch wird beim Austauschen die Zahl um <<i>> erniedrigt.
{=n#+i}	Ist gleichbedeutend mit "{=n-n#+i}"
{=n-m#+i}	Die Zeichen, die im zu durchsuchenden Text dem n-ten bis m-ten Elementbereich der Suchzeichenfolge entsprechen, müssen eine arabische Zahl ohne Vorzeichen sein. Beim Austauschen wird der Wert dieser Zahl um i erhöht und die Zahl an dieser Stelle in den Text eingesetzt. Enthalten die angegebenen Elemente andere Zeichen als Ziffern, so werden die Zeichen, die den angegebenen Elementen entsprechen, an dieser Stelle unverändert in den Text eingesetzt.
{=n#-i}	Ist gleichbedeutend mit "{±n-n#-i}"
{=n-m#-i}	Wie "{±n-m#+i}", jedoch wird beim Austauschen die Zahl um <<i>> erniedrigt.

Laufende Nummern (in Ersatzzeichenfolgen) {726}

In den Ersatzzeichenfolgen kann an Stelle eines einzelnen Zeichens eine laufende Nummer eingefügt werden:

{#}	Ist gleichbedeutend mit "{#1}"
{#n}	Laufende Nummer einfügen und danach um 1 erhöhen; falls der Wert der laufenden Nummer noch undefiniert ist, die laufende Nummer zuvor auf den Wert n setzen.
{#n=}	Die zuletzt eingefügte laufende Nummer nochmals einfügen; falls noch keine eingefügt wurde, den Wert n als laufende Nummer einfügen.
{#n!}	Laufende Nummer auf den Wert n setzen (ohne sie einzufügen).

Arabische Schrift

Anmerkungen zur Codierung in Unicode {787}

Im Unicode-Zeichensatz gibt es für die Arabische Schrift u.a. folgende drei Code-Blöcke:

- 0600-06FF Arabisch
- FB50-FDFF Arabische Präsentationsformen-A
- FE70-FEFF Arabische Präsentationsformen-B

Bei den im zweiten und im dritten Block ("Arabic Presentation Forms") enthaltenen Buchstaben ist vorgegeben, ob der jeweilige Buchstabe nach links, nach beiden Seiten, nach rechts oder nicht verbunden wird.

Der erste Code-Block enthält die arabischen Buchstaben in ihrer Grundform ("Basic Arabic characters" = Form für allein stehende Buchstaben). Es bleibt der jeweiligen Anwendung überlassen, eine für den jeweiligen Kontext erforderliche Form (alleinstehender Buchstabe, nach links verbundener Anfangsbuchstabe, beidseitig verbundener Buchstabe, nach rechts verbundener Endbuchstabe) auszuwählen.

Beim Umcodieren von Unicode nach TUSTEP werden die arabischen Buchstaben aus dem zweiten und dritten Block entsprechend der Tabelle auf Seite 810 umcodiert; die arabischen Buchstaben aus dem ersten Block werden in die Form #?x umcodiert, wobei x der lateinische Kleinbuchstabe ist, der dem entsprechenden beidseitig verbundenen arabischen Buchstaben entspricht. Ausnahmen bilden die beiden Buchstaben Dāl und Zāy; sie werden mit #?D und #?R codiert.

Beim Umcodieren von TUSTEP nach Unicode gelten die im vorhergehenden Absatz aufgeführten Regeln entsprechend umgekehrt.

Parametergesteuerte Programme

Fehlermeldung bei Parameterfehlern

Zusätzlich zur Fehlermeldung wird nun auch noch ein Hinweis angezeigt, in welcher Zeile und in welcher Spalte der Fehler in den Parametern zu finden ist.

Werden die Parameter von einer Datei eingelesen (z.B. bei #ko,quelle,ziel,,+,parameter=datei), wird die Zeilennummer angegeben, die die Zeile mit dem Fehler in der Datei hat.

Folgen die Parameter nach dem Kommando (z.B. bei #ko,quelle,ziel,,+,parameter=*), wird angegeben, in der wievielten Zeile nach dem Kommando der Fehler steckt. In diesem Fall kann mit der Anweisung z,nummer+anzahl die Fehlerstelle im Editor aufgefunden werden. Dabei ist nummer die Satznummer des Kommandos (z.B. des Kommandos #ko) und anzahl die Zahl, die angibt, in der wievielten Zeile nach dem Kommando der Fehler steckt.

(aus der mail von K. Schälkle vom 16.7.2021 an die TUSTEP-Liste)

#SATZParameter

SCH	32101 TusLibertineO = +	{1158}
	32102 TusLibertineOI (kursiv) = +	
	32103 TusLibertineOB (fett) = +	
	32104 TusLibertineO (Kapitälchen) * +	
	32105 TusLibertineOBI (fett-kursiv) = +	
	32111 TusBiolinumO = +	
	32112 TusBiolinumOI (kursiv) = +	
	32113 TusBiolinumOB (fett) = +	
	32114 TusBiolinumO (Kapitälchen) = +	

Bei diesen Fonts fehlten in der Beschreibung die Zeichen = (Ligaturen ff ffi ffl)
 * (echte Kapitälchen)
 + (Mediaevalziffern)

Makros für die Satzumgebung#*CASH

Zusätzlich zu den auf Seite 1293 im Handbuch (#SATZ, Kapitel 16: Arabische Schrift) genannten Zeichen akzeptiert das Makro *CASH jetzt auch die oben unter "Arabische Schrift - Anmerkungen zur Codierung in Unicode" angeführte Codierung #?x in den Eingabedaten für die Umwandlung in den Code für links-, beidseitig-, rechts- und unverbundene Buchstaben.

#PSFONTVOR, #*PSFONT, #*PSDICKTEN, #*PSAUS

In diesen Makros sind jetzt weitere Unterschiede in der Namensgebung von Zeichen berücksichtigt, die in OpenType-Fonts gefunden wurden.

Beispiel:

Als Name für die Mediaevalziffer 2 wird dort `two` oder `twooldstyle` oder `two.oldstyle` oder `two.osf` verwendet.

Der Name für fi-Ligatur lautet `/fi` oder `/f_i` oder `uniFB01`.

Das griechische Delta heißt `Delta` oder `uni0394`, das Omega heißt in manchen Fonts `uni03A9` (das Versalbuchstaben-Alphabet in solchen Fonts lautet dann z.B. `Alpha Beta Gamma uni0394 Epsilon... Chi Psi uni03A9`) gegenüber `alpha beta gamma... lambda uni03BC nu ... chi psi omega` für die Kleinbuchstaben:

das kleine μ heißt normalerweise `mu`, in manchen Fonts aber `uni03BC` (zwischen `lambda` und `nu`), das schluß-sigma hat den Namen `sigma1` oder `uni03C2`.

In vielen Fonts fehlen bestimmte Zeichen, z.B. die Spitzklammern (`uni2329` und `uni232A`; diese werden deshalb aus dem Symbol-Font übernommen) oder sogar (in Noto-Fonts) das Minus-Zeichen; für letzteres muss über Parameter ein Ersatz angegeben werden, falls der Font nicht entsprechend nachgebessert wird.

