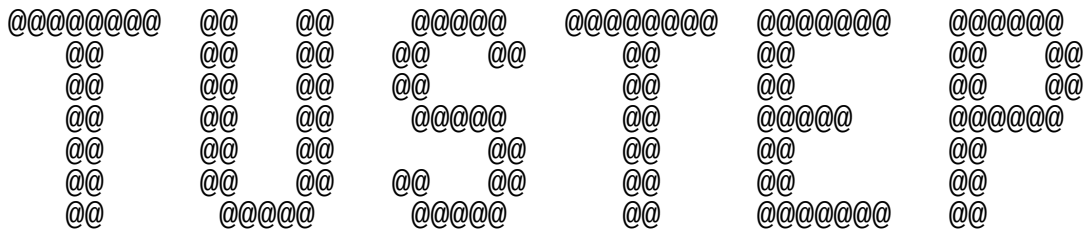


Tuebingen

System of Text Processing Programs

1993

==========

TÜBINGEN UNIVERSITY - CENTER FOR DATA PROCESSING

Department of Literary and Documentary Data Processing

Brunnenstrasse 27, D-72074 Tübingen

This manual provides a full description of TUSTEP's features. This description is intended as a reference work - not as a tutorial for beginners.

To get to know TUSTEP, we suggest attending an introductory course. The University of Tübingen offers a one-week introductory course during its winter and summer recesses. This course is followed up by a two-week main course starting in autumn.

TABLE OF CONTENTS

Introduction

Survey

What is TUSTEP?

The "TUebingen System of Text Processing Programs" TUSTEP was developed at the Computing Center of the University of Tuebingen by the Department of "Literary and Documentary Data Processing." The purpose of TUSTEP is to enable the user to solve problems of scientific textdata-processing with a minimum of short instructions which are closely related to the task, instead of being dependent on less appropriate tools like programming languages.

Therefore, programs have been provided for the most important basic operations of textdata processing. The functions of each program are controlled by parameters. The programs themselves may be combined in a variety of ways to solve problems of the most diverse kind.

The concept of "text*data* processing" used here is meant to distinguish the operational potential of TUSTEP from what is commonly understood by the term word processing. Naturally, TUSTEP also possesses all the features needed to create documents: input, correcting, formating, printing, etc., for these are employed for purposes of documentation and publication required in all scientific and academic fields. But TUSTEP was developed especially for those scientific fields in which texts represent the *main focus* of research: philology, linguistics, literature, history, library science, and other related areas. Scientific activity in all of these fields is not limited to the creation and publication of new texts. Here is where existing texts - whether traditional, already set down in writing or yet to be recorded (including literary texts and historical sources) - need to be saved in new critical editions where they can be the object of linquistic and contextual analyses, and further recorded in bibliographies.

These basic operations of textdata processing (and the corresponding TUSTEP programs) encompass functions that can be generally characterized as follows: *Collation* of different text versions; *correcting* not only in the interactive Editor mode, but also with the use of prepared (or automatically generated) correction instructions; *breakdown* of text into user-defined elements (z. B. semantic patterns); *sorting* textelements or lengthy units of text according to a number of different alphabets and other criteria; *Generating an index* by compiling sorted text-elements; *Processing* textdata according to user-defined selective criteria, replacing, moving, enhancing, concluding and comparing text, calculating numerical values contained in the text (e. g. calender dates) or those which can be determined from the text (e. g. the number of words in a sentence), and output in various formats, including those required by other operating systems (e. g. SPSS for statistical evaluation).

The tasks which can be processed with the help of TUSTEP range from typing a term paper to the preparation and automatic composition of extensive scientific works (bibliographies, encyclopedias, indices, concordances, dictionaries, special editions and of course monographs) in a quality to which one is accustomed in letterpress printing.

In addition to programs for the basic operations of textdata processing, TUSTEP is equipped with all the necessary organizational functions such as file handling and defining new commands, functions which are normally covered by the job control language (JCL) of the operating system (OS). Thus, an identical user interface independent of the computer and its OS is provided. This not only spares the user the additional effort of relearning new commands when he changes to a computer with a different operating system, but also enables him to use existing sequences of TUSTEP commands unchanged.

The implementation of a universal user interface requires sacrificing special features offered by certain computers and operating systems. For this reason, many graphic features common to a PC screen display are not available.

TUSTEP is permanently improved and expanded in order to facilitate solutions for new problems in scientific textdata processing. New developments in the field of hardware and operating systems are also taken into account.

TUSTEP owes many of its special features to the suggestions, criticism and cooperation of users from practically all areas of the liberal arts, including many users from a number of other universities besides the University of Tübingen. We at the TUSTEP programming staff remain grateful for any recommendations you may provide.

Although originally developed and implemented at the University of Tübingen, TUSTEP is now being used by a number of other universities. At present, it can be run under the following operating systems: DOS (IBM PCs and compatible), UNIX, VMS (DEC VAX) as well as BS2000 (SIEMENS), MVS (IBM), VM/CMS (IBM). The last three systems listed here are limited in the TUSTEP editor functions.

Basic Operations for Processing Textdata in TUSTEP

EDITING

Entering, modifying, replacing and searching textdata with the texteditor onscreen [#EDIT]

Automatic correction of textdata with user-defined correction instructions [#CORRECT]

COMPARE

Comparing different versions of a text; protocolling and storing of differences found [#COMPARE]

A line-by-line synoptic listing of the basic text and differences contained in the other versions [#COLLATE]

PROCESSING

Selecting, substituting, rearranging, supplementing, summarizing and comparing text parts on the basis of given rules and conditions; calculating with numerical values (including calendar dates), which are already present in the text or which can be derived from it; output in various formats (also for subsequent processing outside of TUSTEP) [#COPY]

Replacing text abbreviations with their complete text (including extended text units), which are located in their own file. [#INSERT]

Maintaining and updating cross-references [#NUMBER]

PREPARING INDEXES

Compiling index entries by breaking down text units into their respective components, or by extracting marked text parts. If necessary, text parts may be supplemented and modified. Defining the sort criteria and the sort alphabet, supplementing references; distinguishing between different types of entries [#PINDEX]

PREPARING FOR SORTING

Organizing logically related text parts into sort units; establishing user-defined sort criteria (selecting and ordering defined text parts of the sorting units), the sort values for any number of character strings as well as the sorting alphabet involved in the final sort to determine the order of the sort units. [#PRESORT]

SORTING

Arranging data records in alphabetically ascending or descending order as determined by the sort criteria. These are usually generated by the programs #PINDEX or #PRESORT, and are located in sorting fields [#SORT]

Merging a number of files containing sorted data into a single file [#MERGE]

GENERATING INDEXES

Organizing multiple and any hierarchically structured index entries or text units after sorting; supplementing and substituting text parts and references; distinguishing different types of entries; calculating absolute and relative frequencies [#GINDEX]

GENERATING LISTINGS

Preparing output for line printers, dot matrix printers and laser printers

- in the form in which the data are recorded in the file, i.e. control characters are not interpreted but printed along with the other characters. [#GLISTING]

- in a format and an arrangement which can be freely defined (using control characters contained in the text) by taking advantage of the entire inventory of typefaces and characters available for the selected printer, with automatic hyphenation, line division and page make-up, including page justification and placing of footnotes [#FORMAT]

- for forms (e.g. address labels, catalog cards, standardized letters, office forms) [#GFORMS]

GENERATING PHOTO COMPOSITION

Converting textdata into a form required by either a PostScript printer or a professional photo composition device (presently DIGISET, Monotype LASERCOMP, PostScript phototypesetters). Typographic features include automatic line division (with either justified or ragged right margins, tab settings) and automatic page make-up with running heads, titles, text, insertions in smaller type, side bar text, footnotes and up to nine critical apparatuses. An extensive selection of fonts and special characters is also available. [#COMPOSITION]

Organizational Features of TUSTEP

DATA TRANSFER

Converting textdata from files of the operating system into TUSTEP format; converting data from TUSTEP files into files of the operating system (can also be used for electronic data transfer for networks and E-mail) [#CONVERT]

FILE MANAGEMENT

Creating, cataloging, opening, closing, renaming and erasing files [#CREATE, #OPEN, #CLOSE, #RENAME, #ERASE]

MAGNETIC TAPE SERVICES

Data transfer to and from magnetic tape (can also be used to exchange data between different computers); View information concerning the contents of magnetic tapes [#MTWRITE, #MTREAD, #MTCOPY, #MTINFORM]

JOB CONTROL

Executing/controlling sequences of commands and programs; creating and executing user-defined commands (macros) [#EXECUTE, #MACRO]

F i l e s

Survey:

Project and Carrier Under the early operating systems on which TUSTEP was implemented, a file was identified by the user name and the name of the file itself. The valid user name was determined by the system (system manager). Such names were usually given for each task area, e.g. for an individual research project. For personal files, the user name did not have to be given explicitly, but it was mandatory for accessing files from other "projects".

Files were usually saved to hard disk, but could also be saved to removable disks. These were changed by the operator whenever necessary - thus every removable disk also required its own name, which was specified as the name given for the "data carrier". This name had to be included when requesting a file located on a removable disk.

These name specifications have been retained in the TUSTEP program as reflected in the the name categories of "project" and "carrier". These terms are still used in TUSTEP in order to enter the necessary file specifications (in addition to the actual file name) for clearly identifying a file.

Under newer operating systems, a file is identified by specifying its path and file name. A file is created by having its name placed in a directory. A directory is a special kind of file managed by the operating system. Except for the root directory, the name of each directory is listed in the directory at the next higher level. Thus, all directories branch off either directly or indirectly (via one or more other directories) from the root directory. The path specification for a file contains the names of all directories needed to find the file, starting with the root directory and ending with the directory where the name of the file is entered.

The directory (and any other directories above it) in which the name of a file is entered corresponds to the TUSTEP "Project" specification. The remaining part of the path is specified by "Carrier". The default settings for project and carrier have been selected in such a manner that in most ordinary cases merely specifying the file name is all that is necessary to address a file. More details concerning the project and carrier specifications are described in the chapter entitled "System environment" (see "System variables TUSTEP_DSK, TUSTEP_SCR, TUSTEP_PRJ" page 39).

Project names and file names

In TUSTEP, the complete file name (i.e. file name and its project) is written as "Project*Name". If the project specification and the following asterisk are omitted, the program will automatically use as "project" the project name most recently set with the command #DEFINE (see page 83). If only the project name is omitted, but the asterisk is specified before the name of the file, the active "project" will use the project name set during TUSTEP initialization (see page 48, "System variables TUSTEP_DSK, TUSTEP_SCR, TUSTEP_PRJ").

In the VMS version, the form "project*name(version)" is also possible. If the specification (version) is omitted, the file having the largest version number will be assumed.

The following list shows how the TUSTEP form of the file name is converted to the form used by the operating system, as well as the proper syntax for writing "project" and "name" according to the operating system being used. Here, "A" represents any letter from A to Z and "X" represents any letter from A to Z, any digit from 0 to 9 or the " " character. The maximum number of characters is given for each case.

BS2000: Project*Name --> $PROJECT.NAME

Project: AXXXXXXX
Name: AXXXXXXX oder AXXXXXXX.AXXXXXXX

DOS: Project*Name --> \PROJECT\NAME
-*Name --> \NAME

Project: AXXXXXXX or AXXXXXXX.XXX or AXXXXXXX\XXX
Name: AXXXXXXX or AXXXXXXX.XXX

MVS: Project*Name --> PROJECT.NAME

Project: AXXXXXXX
Name: AXXXXXXX or AXXXXXXX.XXXXXXXX

Due to organizational reasons, some computing centers may use different specifications. Please consult the information bulletin of the respective computing center for proper use.

UNIX:Project*Name --> /project/name
-*Name --> /name

Project: AXXXXXXX or AXXXXXXX.XXX or AXXXXXXX/XXX
Name: AXXXXXXX or AXXXXXXX.XXX

VM/CMS: Project*Name --> NAME - PROJECT
Project*Name1.Name2 --> NAME1 NAME2 PROJECT

Project: A
Name: AXXXXXXX

For permanent files, the project determines the file mode (the minidisk). For temporary files (scratch files), file mode T is always used. Thus, AXXXXXXX is a valid project syntax for these files. However, to ensure that file names remain unique for the operating system, a running number is used internally as the file type.

VMS: Project*Name --> [PROJECT]NAME
Project*Name(Version) --> [PROJECT]NAME;VERSION

Project: AXXXXXXXXXX or AXXXXXXX.XXX
Name: AXXXXXXXXXX or AXXXXXXX.XXX
Version: 1 to 999

File names having the form "TUSTEP.xxx" (where xxx stands for any combination of letters and digits) are reserved for TUSTEP and may not be used for naming files. But there are two exceptions to this:

The file name "TUSTEP.INI" is reserved for the TUSTEP start file, which contains commands that are automatically executed whenever TUSTEP is initialized (cf. page 48).

The file name "TUSTEP.USE" is reserved for the TUSTEP statistics file, where user statistics for the TUSTEP program are recorded. These statistics can be viewed and printed out with the command #STATISTICS (see page 170).

Standard files

In addition to files which are explicitly opened or created by the user, the following standard files are also available:

Standard DATA file
Standard EDITOR file
Standard LISTING file
Standard TEXT file

The names of these files depend on the operating system being used. Therefore, they should be addressed with the file name -STD- *only*. They are automatically created whenever they are needed (in the MS-DOS version during initialization). To find out when and where these files may be used, please refer to the respective TUSTEP commands.

TUSTEP files

TUSTEP programs assume that the data to be processed are recorded in TUSTEP files. TUSTEP files are files having a unique TUSTEP structure. They are created using the command #CREATE (cf. page 79) and are assigned either a SEQ or RAN type. They can be created only on a hard disk or diskette, but not on magnetic tape.

TUSTEP files can be classified into three groups: text files, program files, and segment files. They are all created in the same manner, differing only in the way they are employed and the type of data they contain.

System files

In order to transfer data to or from other programs TUSTEP is capable of working with "system files". System files are files having the usual structure common to the respective operating system. Such files may also be created with the command #CREATE (see page 70) and have the type specification of SDF. The following list shows what kind of files may be used for data transfer.

BS2000: SAM files with RECFORM=V

DOS: ASCII files

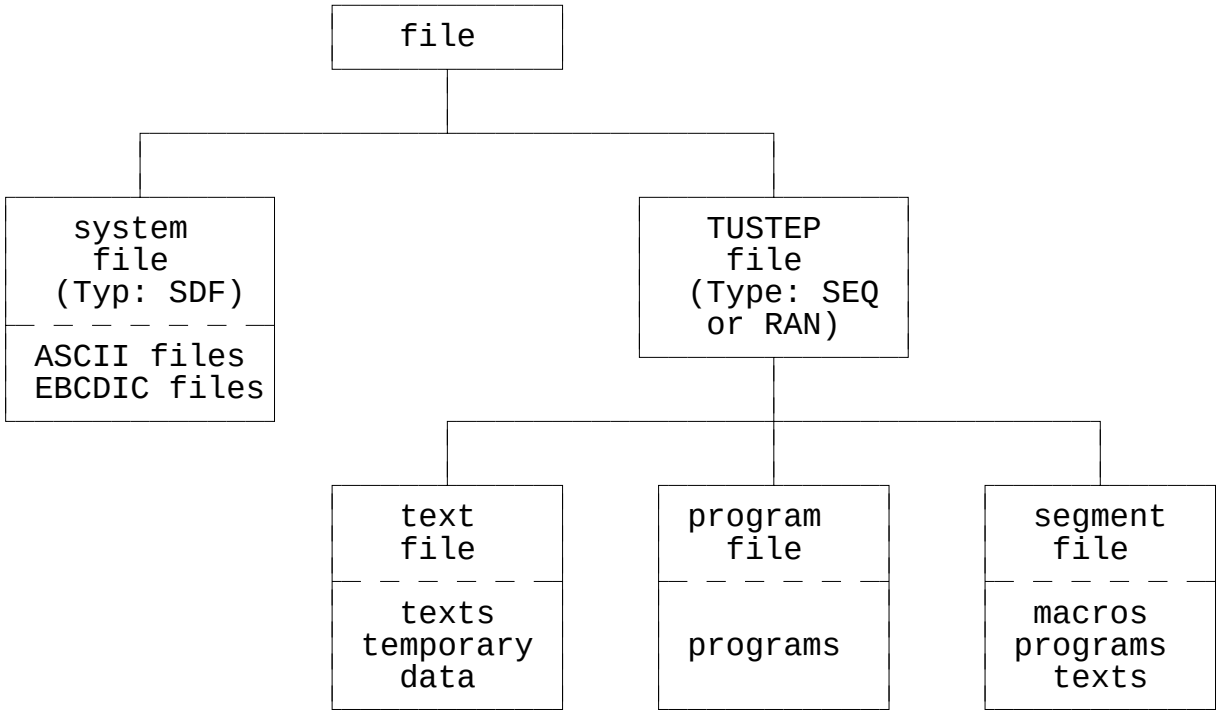
MVS: DSORG=PS with RECFM=FB or VB
DSORG=PO with RECFM=FB

UNIX: ASCII-Dateien

VM/CMS: CMS files where RECFM=F or V

VMS: RMS files

File types



Text files

A text file is a TUSTEP file which contains text or temporary data (e.g. sort keys).

If a text file contains text, its records are normally numbered in text mode. If the text being edited is to be saved as a segment file (e.g. an address or a text mask), its records must be numbered in program mode.

If a text file contains temporary data, the records are numbered by TUSTEP in the appropriate fashion and the record numbers will be interpreted as such if necessary. A file of this type usually cannot be edited with the editor, and - except for listing files - there is no reason to do so.

Program files

A program file is a TUSTEP file which contains a program (= a sequence of commands) or a command macro. Its records are numbered in program mode.

Segment files

A segment file is a TUSTEP file which is divided into segments. A segment may contain a program, a macro or text. Every segment has a name by which it can be addressed. At the beginning of a segment file is a table of contents showing the names of the segments contained in the file. This table of contents is created automatically whenever data edited in the EDITOR file are saved as a new segment to an empty TUSTEP file. This table of contents is also updated automatically whenever data from the EDITOR is saved to an existing segment file. In both cases, data are saved by using the Unload instruction in the Editor (see page 187).

A segment file should *never* be edited with the Editor directly. Instead, it should be created and processed only with the editing instructions "Load" and "Unload" (see instruction ʟ on page 186 and instruction ᴜ on page 187). Since the record numbers of a segment file serve as references for its table of contents, they should not be renumbered. The individual segments of a segment file can be sorted alphabetically by name with the standard macro #*SESO. This macro can also be used to renumber a segment file if its record numbers have been accidently altered. In addition, this macro also offers the possibility of compiling a new table of contents for a segment file. This is necessary when, for example, two segment files have been copied to a single file. Further information concerning this macro can be obtained with the command #INFORM,*SESO.

Macro files

A macro file is a segment file whose individual segments each contain a macro (user-defined commands; see chapter entitled "Macros" starting on page 266). The name of the segment is also the name of the macro. To gain access to these macros, the macro file must be defined as such with the command #DEFINE (see page 83).

Editor file

The editor file is the TUSTEP file currently being processed with the Editor, i.e. the file most recently given in the specification FILE when calling up the editor (cf. command #EDIT page 91), or the file last specified with the relevant editing instruction (see instruction F page 191). This file should not be confused with the standard EDITOR file which is automatically available to the user for editing purposes and only becomes an editor file when actually processed by the editor.

Print files

A print file is a TUSTEP file containing data which have been prepared for printing. Generally, such a file is set up by giving its name for the specification LISTING in TUSTEP programs. It can be sent to a printer using the command #PRINT (see page 160).

One essential characteristic of a print file is that the first character of each record is a feed character or an asterisk. The effect of the feed characters - 0 1 2 3 4 5 6 7 / is a new page, 0 to 7 line feeds and a one and a half line feed. A line beginning with an asterisk is a control line whose function is determined by the second character. If a slash ("/") follows the asterisk, printing is switched to one-and-a-half-line spacing. Thus, line feeds are increased by a factor of 1.5. A "1" after the asterisk switches back to single-space printing.

In addition to the usual data, such a file may also contain printer control information described in the table below.

#=nnn	absolute positioning to column nnn
#-nnn	relative positioning by nnn columns to the left
#+nnn	relative positioning by nnn columns to the right
#=000	mark current column position
#-000	positioning to marked column
#+000	positioning to the next available column to the right

Related files

Related files are two TUSTEP files where the records of the first file contain pointers to the records of the second file. Such files are generated by the commands #PINDEX and #PRESORT (see pages 156 and 158) by giving their names for the specifications DESTINATION and DATA. In this case, only those text parts needed for sorting (i.e. in most cases the reference and the sort key) are written along with a pointer to the file given in the specification DESTINATION. The remaining parts (i.e. the data to be sorted) are written to the file given in the specification DATA.

The file with pointers may only be sorted or merged. By no means should the record numbers of this file be altered, since they contain the pointers. The other file whose records are referred to, should not be altered in any way; otherwise, the pointers will not correspond to the intended records.

The data of related files may be recombined with the commands #SORT and #MERGE (see pages 167 and 131) and with the commands #GINDEX and #COPY (see pages 105 and 75). In this case, the file with the pointers must be given in the specification SOURCE and the other file in the specification DATA.

Start file

The start file is a file called *TUSTEP.INI, which contain commands that are automatically executed whenever TUSTEP is initialized (see page 48).

Statistics file

The statistics file is a file called *TUSTEP.USE, in which the TUSTEP user statistics are listed. These statistics can be listed with the command #STATISTICS (see page 170).

Scratch files

In TUSTEP a distinction is made between permanent (cataloged) files and temporary files. Permanent files remain so until they are explicitly erased. Temporary files are automatically erases at the end of a TUSTEP session, assuming that they were not explicitly erased beforehand.

Temporary files, also known as scratch files, are only used to store intermediate data, since scratch files are only used for the particular TUSTEP-Sitzung in which they were created. The usefulness of scratch files is that files used for intermediate

data can be automatically erased at the end of a TUSTEP session. Furthermore, a conflict of names with other files is avoided when a number of sessions are carried out at the same time.

A conflict of names is avoided in that the operating system converts the names of scratch files to internally standardized names. Furthermore, at the operating system level, all scratch files are created in the same project (directory). (see system variable TUSTEP_SCR page 39). Thus, it is not necessary that the project specified by the file name actually exists. However, at the TUSTEP level names are given to temporary files in the same manner as that for permanent files.

Magnetic tape files

TUSTEP programs cannot directly process files recorded on magnetic tape. For purposes of data protection and data exchange with other TUSTEP users, however, files may be copied from disk onto magnetic tape with the command #MTWRITE (see page 144). Such files may then be copied from magnetic tape back onto disk with the command #MTREAD (see page 141). The command #MTINFORM (page 137) is used to obtain a list of files which are recorded on such a tape. The command #MTCOPY (page 132) is used to make copies of such tapes.

Some computing centers provide standardized macros for reading and writing magnetic tapes in different data formats. These standard macros are outlined in the information bulletin of the respective computing center.

Structure of a TUSTEP file

Data in a TUSTEP file are organized into records. If, for example, a file contains the text of a book, each record of the file can contine a line of the text. But you are free to choose how you distribute your text as records. The text assigned to each record can be broken down into distinct and appropriate units. The only limit here is the length of the record. record length If a file is to be processed with the Editor, none of its records may contain more than 600 characters. In all other cases records may have a length of up to 32,000 characters.

Each record "records" not only the text it contains but also the record length, a record number and two reference points. When working with TUSTEP the only related organizational feature that is visible is the record number (see "record numbers" below). The reference points are used internally for working internally with larger files (in TUSTEP up to 7 GB per file)

One of the reference points of each record marks the record that precedes it, the other reference point marks the following record. Thus the logical sequence of records does not necessarily have to correspond to their physical sequence. This feature is especially relevant when, for example, a record is

inserted into a file during editing. At the TUSTEP level this record is written at the end of the file, but its reference points are established in a way that the record appears at its logical position in the file. Otherwise, all records following the inserted record would have to be shifted toward the end of the file, e.g. the rest of the file would have to be rewritten by the computer. The only disadvantage of using such reference points is that data access to a file declines in performance when significant alterations are made in the file. However, this can be corrected by reorganizing the file (see "Reorganizing a TUSTEP file").

Record numbers

TUSTEP automatically gives each record its own record number. Expressed in the form of a page number and a line number, record numbers can be easily used to orgainze a text in a manner analogous to its printed form. For example, if a file contains the text of a book, where each record contains a text line, the record numbers can be selected in such a way that the page number of the file's records correspond to the the page number in the book, and that the line number of file's records corresponds to the running number of the lines on each of the book's pages. Each text passage can thereby be quickly found, for example, when working with the Editor, since the printed and file references correspond to each other. Every line number can be expanded by a distinction number. The main purpose of a distinction number is its function of giving a unique number to lines inserted between line numbers without having to alter the the numbers of the following records. Thus, an inserted record is usally given the same page and line number of the records preceding it, but is given an additional distinction number.

In a file containing a program or macro there is no need to organize records into pages. In this case, a different numbering of records is employed, where the page number is always zero and thus not present. This type of numbering is known as "program mode numbering". But it can also be used for short texts where the page nubmer is of no relevance. The type of numbering described in the preceding paragraph is called "text mode numbering".

Record numbers in text mode:

This type of record numbering is generally used in TUSTEP unless specified otherwise (normally with the specification MODE=P).

For records numbered in text mode, the record number consists of a page number, a line number and a distinction number. The page number can have a maximum of 6 digits, the line and distinction numbers a maximum of 3 digits each. A record number thus has the

following syntax:

page number.line number/distinction number

The distinction number is usually 0. In this case the 0 and the preceding slash are omitted. The distinction number must be written with leading zeros; trailing zeros may be omitted. For example, the record numbers 1.2/3, 1.2/30 and 1.2/300 are identical.

Record numbers in [] program mode:

TUSTEP uses this type of record numbering for program files. It is normally activated by specifying MODE=P in the respective program.

For records numbered in program mode, the record number consists of a line number and a distinction number. The line number can have a maximum of 4 digits; the distinction number a maximum of 2 digits. Here a record number has the following syntax:

line number/distinction number

The distinction number is usually 0. In this case, the 0 and the preceding slash are omitted. The distinction number must be written with leading zeros; trailing zeros may be omitted. For example, the record numbers 1/2 and 1/20 are identical.

Reorganizing a TUSTEP file

When the data of a file are recorded, the file's records are organized in an unbroken and continous sequence. But when this file is altered by corrections made with the Editor, this tidy organization - from the computer's point of view - can become quite disarrayed for a number of reasons:

- whenever a record is erased, a gap is created.
- whenever a record is shortend, another gap is created.
- after a record is lengthened, it no longer fits into its old place, unless by chance a gap has been created immediately before or after the record, thus providing enough room for the extra space required. Thus, the record is written at the end of the file but logically placed at its proper location by means of record markers. The original space remains as a gap in the file. This case of where a record no longer fits into its old position is most common when a character string is automatically replaced by a longer one.
- When a record is inserted in a file, a check is made as to whether a sufficient gap already exists at the corresponding location. If yes, it is written there. If not, it is written at the end of the file and then assigned its logical position by means of record markers.

Extensive alterations made to a file with the Editor can therefore result in correspondingly large gaps in the file.

Eventually the logical sequence of its records (used for the actual processing of the file) has little or no correlation with the file's physical arrangement. This increases the amount of time needed to access individual records: first, because the ever-growing gaps in the file must still be read and second, because widely scattered parts of the file must be read. In addition, the resulting gaps are an unnecessary waste of disk space.

For this reason, files processed by the Editor should be reorganized from time to time. This is best carried out with the command #RESTORE (see page 165).

The problem just described about a file's records also applies to the process of saving segments of a segment file to disk. The gaps created here can be quite considerable, since the amount of data contained in individual segments is generally much larger. It is therefore strongly recommended to reorganize segment files when optimizing your disk space. In addition to the command #RESTORE, there is another way to reorganize a file when working with the Editor. After saving a segment to a segment file with the unload instruction, the load instruction "L,,-" (see page 186) can be used to load the entire segment file (with all of its segments), which is then immediately saved once more with the unload instruction "U!,,-" (see page 187).

Creating files

Except for its standard files, TUSTEP does not create any files automatically. Files that do not yet exist but are to be used in TUSTEP must be explicitly created with the command #CREATE (see page 79).

Opening files

To process existing files in TUSTEP, they must first be opened (= admitted for processing) with the command #OPEN (see page 151). An opened file is only valid for the current TUSTEP session.

When opening a file, you can determine whether it should be read only or also written to. This avoids the accidental writing or erasing of files that are meant to be read only.

Closing files

To prevent the accidental access to files created or opened in the same TUSTEP session, the command #CLOSE (see page 64) can be used to close access to files no longer needed.

Since only approximately 100 files can be opened at the same time in a single TUSTEP session (excluding TUSTEP's internal files), it may be necessary to close files that are presently not required in order to make space for opening or creating other files.

Please note that scratch files are automatically erased when they are closed. They cease to exist.

Erasing files

Files which are presently (or later) no longer needed can be erased with the command #ERASE (see page 93). Whenever a TUSTEP session is terminated (not just interrupted), the scratch files of this session will be erased automatically.

Printing a file

Unless it is a print file, a file must be prepared for printing in order to print it out. This can be done using TUSTEP's FORMAT program if the file contains the formatting instructions necessary for interpreting this program. Or it can be prepared for printing with the TUSTEP GLISTING program. This program establishes the file's page and line organization for printing and generates the corresponding listing, which is written to a LISTING file.

A LISTING file is printed with the command #PRINT (see page 160). Here the data will be converted into the control codes compatible for the selected printer.

When specifying certain types of printer (e.g. Postscript printers) the printer specification also includes whether the format of the printed output is to be portrait or in landscape. In sideways (landscape) printing two "pages" can be automatically printed on the same page. To obtain a list of defined printers, activate the command #LIST,PRINTERTYPES (see page 121).

Restoring files

In TUSTEP files are automatically terminated after being written to. Even after an interruption of a TUSTEP program files will be properly terminated if possible. If this is not possible (e.g. is the file is full and cannot be expanded; or if the system locks up), the data of such files will be blocked. This helps avoid unseen errors in programs which process these data. If there is a further need for such data, they must first be recopied with the command #RESTORE (see page 165).

Data transfer

Survey:

Data transfer to and from TUSTEP files

The TUSTEP command #CONVERT (see page 70) may be used to transfer data from system files to TUSTEP files or vice versa. Data may be recoded in the process. For example, the specification MODE may be used to convert "^a" to "ä" and vice versa. With the specification CODE, the coding of single characters may be altered if, for example, the data have been recorded in an EBCDIC code differing from the international EBCDIC code. If the coding does not need to be altered, the editor instructions "Load" and "Unload" (see instruction ʟ page 186 and instruction ᴜ page 187) may be also used to transfer data.

Data exchange / Data transfer [data exchange

If direct file access is not possible, there are two other ways of exchanging data with other TUSTEP users:

a) Data exchange by magnetic tape:

TUSTEP files may be recorded on magnetic tape with the command #MTWRITE (see page 144). Data on magnetic tape are copied into TUSTEP files with the command #MTREAD (cf. page 141). For writing, make sure that the selected recording density can be read by the computer of the data recipient.

b) Data exchange by data lines

In order to transmit data of a TUSTEP file by network, the data must first be copied into a system file using the command #CONVERT (see page 70), and then converted into a format appropriate for data exchange. For this purpose, MODE=TX" is normally specified when #CONVERT is called up.

The data recipient can reconvert the data into its original form and copy them into a TUSTEP file by using the same command #CONVERT. For this purpose, the data recipient must specify MODE=XT when #CONVERT is called up.

Data Backup

Survey

Data backup without using TUSTEP

TUSTEP files can be backed up like any other files by using the relevant backup command of the operating system (e.g. the DOS BACKUP program). They must then be retrieved with the corresponding programs (e.g. the DOS RESTORE program).

Data backup with TUSTEP to magnetic tape/cassette

TUSTEP files can be backed up on a magnetic tape or cassete with the command #MTWRITE (see page 144). Such a file can be restored with the command #MTREAD (see page 141).

A file can be repeatedly saved to tape or cassette (e.g. after making any considerable change to its contents) with the same file name. Files already located on the tape not be overwritten unless this has been expressedly specified in the command #MTWRITE. However, it is advisable to alternate tapes or cassettes when making backups instead of using the same tape or cassette. If, for example, the last tape or cassette used for a backup proves to be defect, there is always another backup available on the tape or cassette used for the backup prior to the most recent one.

If a magnetic tape or cassette is to be written for the first time with #MTWRITE, it must first be given a machine-readable label with the command #MTLABEL (see page 139) if this has not been already provided for (e.g. by the computing center).

Two other commands are used to manage files saved to magnetic tape and cassette. The command #MTINFORM (see page 137) is used to compile a list of all files on a tape or cassette. The command #MTCOPY (see page 132) is used to copy files from one tape or cassette to another.

Before a magnetic tape or cassette can be used, a system variable must first be defined. This tells TUSTEP the name (address) of the device on which the tape or cassette is located. Furthermore, on some computers certain operating system routines must be executed before and after the use of magnetic tapes and cassettes. A more detailed description of this can be found in the chapter entitled "System environment" (see "System variable TUSTEP_MT1, TUSTEP_MT2" page 43 and "Magnetic tape operation" page 44).

Data backup with TUSTEP to diskette

The DOS version of TUSTEP has no special commands for backing up data to diskette. If a backup of a file on a hard disk is to be made on a diskette, the command #CREATE is used to create a file on the diskette, to which the file to be backed up can then be written:

```
#create,-*text,seq-ap,carrier=a
#restore,text,-*text,,+
#close,-*text
```

If this file should be returned to the hard disk, it must first be opened and then copied to a file on the hard disk:

```
#open,-*text,carrier=a
#restore,-*text,text,+
#close,-*text
```

When saving a file to diskette with TUSTEP, the file must be copied in its entiriety to one diskette. TUSTEP cannot distribute a file to two or more diskettes. If a file is too large to fit onto one diskette only, its data must be distributed among a number of discrete files, when can then be backed up to diskette. The cumbersome process can be avoided by compressing the data when they are backed up. Compression practically doubles the amount of data that can be saved on a single diskette.

To backup a file in compressed mode, the line with the command #restore in the first command sequence above must be replaced by the following command:

```
#convert,text,-*text,tk,+
```

The corresponding command in the second command sequence above (for copying back to the hard disk) would be

```
#convert,-*text,text,kt,+
```

In contrast to file backup to magnetic tape or cassette using the command #MTWRITE, the following must be observed when backing up files to diskette: If a file is repeatedly backed up (e.g. after making any considerable change in its contents) using the same name on the same diskette, the backup copy of the file already on diskette will be overwritten each time. This can be avoided by assigning different names to the files on diskette (e.g. test.1, test.2 etc. when backing up a file called test). However, it is also wiser to alternate diskettes when backing up files instead of using the same diskette for all backups. If, for example, the last diskette used for a backup proves to be defect, there is always another backup available on the diskette used for the backup prior to the most recent one.

System environment

Survey

General information

In most cases you will not find it necessary to use the various ways of configuring TUSTEP for the respective computer environment as described in this chapter. TUSTEP's default settings have been either selected or set up during installation in such a way that all you have to do is call up TUSTEP in order to work with it.

System configuration is carried out by variables that are defined before the TUSTEP program is called up. Variables that are used by TUSTEP but not yet defined, are automatically defined and assigned a standard value. In the DOS version of TUSTEP, however, these variables are not automatically defined, since the amount of system memory required is often insufficient. However, they will still be treated as if they have been defined with a standard value.

These variables (usually called environment variables) should not be confused with the variables defined at the TUSTEP level. To avoid any confusion in this matter, the former type of variables will be called called "system variables" and the latter "TUSTEP variables".

System variables can be defined with the following commands at the operating system level:

```
DOS:   set NAME=value
UNIX:  setenv NAME value
VMS:   define NAME value
```

Note: UNIX distinguishes between uppercase and lowercase letters. The command name setenv must always be written in lowercase letters; the names of variables in uppercase letters. The latter applies only to system variables evaluated by TUSTEP.

If a TUSTEP session is temporarily interrupted and continued later, the system variables evaluated by TUSTEP may not be altered in the meantime. If they have been altered, they must be first be restored to the same values in effect when the TUSTEP session was interrupted before the session can be continued.

System variable TUSTEP_HST

To make sure that the journal also lists the name of the computer being used, the system variable TUSTEP_HST tells TUSTEP the name of the current (host) computer. Its name is then included in the start and end messages produced by the individual programs. The standard value here is the name provided by the respective operating system. An inquiry of the defined computer names can be made with command macros (see page 275).

System variable TUSTEP_USR

User identification (userid, loginid) is conveyed to TUSTEP by means of the system variable TUSTEP_USR. It is used to identify

the cover sheets of compter printouts. An inqiry of this user identification information can be made with command macros (see page 275).

System variable TUSTEP_NAM

Besides the user ID number, cover sheets in printouts also feature the user's name. The user name can be determined with the command #DEFINE. The default setting can be given with the system variable TUSTEP_NAM. The standard value here is taken from the contents of the system variable TUSTEP_USR.

System variables TUSTEP_DSK, TUSTEP_SCR, TUSTEP_PRJ

TUSTEP interprets these variables when identifying files. In TUSTEP a file is identified by three specifications:

1. Carrier: defines the first part of a file's path up to the last or next-to-last directory.

2. Project: Defines the rest of the file's path (the part not yet defined by the carrier). This is usually the name of the directory in which the file is listed. But it can also be the names of the last two directories when the name of the last directory consists of no more than three letters and the name of the next-to-last directory consists of no more than eight characters. The two names must be separated by a delimiter character. In DOS this is the backslash character, in UNIX the forward slash, and in VMS a period.

3. Name: name of the file with no path specifications.

Default values are possible for both the carrier as well as for the project. In most cases specifying just the file name is sufficient for it it to be correctly identified by TUSTEP.

The project can also be specified, but this is only necessary when it differs from the current setting. If necessary, the project is written before the file name and separated from it with an asterisk. The default project setting is defined by the system variable TUSTEP_PRJ which is read during TUSTEP initialization. This default value can be reset with the TUSTEP command #DEFINE; here the contents of the system variable remains unchanged. Supplying an asterisk in front of a file name (without specifying a project name) is the same as specifying the project established during TUSTEP initialization.

A carrier is specified only for commands which either provide file access for other TUSTEP commands (#OPEN, #CREATE), bar such access (#CLOSE, #ERASE) or supply information concerning existing files (#LIST). In all other commands, files are addressed by their project and file names only. (This also means that TUSTEP cannot access at the same time two files having the same project and file name and located on different carriers.) The carrier specification can be omitted if it corresponds to the default value. For permanent files, the default value for the carrier is specified in TUSTEP_DSK, for temporary files TUSTEP_SCR. The default value for the carrier cannot be changed

with TUSTEP commands. If necessary, a different system variable with the appropriate specification must be given.

The standard value for the system variables TUSTEP_DSK and TUSTEP_PRJ are taken from the corresponding path components of the "home directory" (usually the directory that is set after login). If, for example, this home directory is called /home/group/user, then /home/group is used as the standard value for the system variable TUSTEP_DSK, and user as the standard value for the system variable TUSTEP_PRJ. The standard value for the system variable TUSTEP_SCR uses the specification given for the system variable TUSTEP_DSK.

In this example and in the rest of this chapter, path specifications follow the UNIX conventions. If TUSTEP is to be used with a different operating system, please note the following differences:

- DOS: The path specification always begins with the letter of the drive (hard disk or diskette). This letter is separated from the rest of the path specification by a colon. A backslash is used to separate the individual directory names instead of a forward slash. For example, the path /home/group under DOS should read C:\home\group, where home and group stand for a directory name. - Under DOS the home directory is assumed to be C:\tustep. Thus C: is used as the standard value for the system variable TUSTEP_DSK, and tustep as the standard value for the system variable TUSTEP_PRJ.

- VMS: A path always begins with the name of a drive. A colon is placed between this name and the rest of the path specification, which is placed in square brackets. Furthermore, a period is used instead of a forward slash to separate directory names. For example, /home/group should be written as DISK:[home.group], where home and group stand for the name of a directory.

For a file in TUSTEP having the name file, the default settings described above are used to convert the three components carrier + project + name into the name /home/group/user/file.

If a file having the name /home/group/userx/file is to be addressed, this can be done by specifying userx as the project along with the filename file, i.e. userx*file. If, however, userex has been previously set as the project with the command #DEFINIERE, there is no need to include the project specification.

But if a file having the name /home/groupx/userx/file is to be addressed, a system variable having the value /home/groupx must first be defined. The name of the system variable can be freely chosen. This variable name must be specified as the carrier when creating or opening the file. The project specification is the same as described above.

If a file having the name /home/group/user/dir/file is to be addressed, this can be done by specifying user/dir as the project, as long as the actual directory name for user is no longer than 8 characters and the actual name used for dir is no longer than 3 characters. Otherwise, a system variable having

the value /home/group/user must be defined and then specified as the carrier. Here the project would be dir.

For file names whose paths contain more directories than shown in the sample paths here, system variables would have to be defined for the carrier specifications with the appropriate directory names being specified.

Naming files in TUSTEP is limited on one hand by the compatibility requirements of the various operating systems and on the other hand by syntactical reasons when entering commands and Editor instructions. As a result, TUSTEP may not always be able to address a file not created with TUSTEP if an invalid notation has been used in the file's name. This generally applies to files whose name contains special characters or umlauts. Under UNIX this also affects files whose names include uppercase letters, since in TUSTEP all letters of a file name (regardless of whether they are uppercase or lowercase) are evaluted and passed on to UNIX as lowercase letters. This can be corrected by changing the name of a file whose present name is invalid in TUSTEP at the operating system level until it is given a name that is also valid for TUSTEP.

The following operating system command can be used to rename files:

```
DOS:   rename oldname newname
UNIX:  mv oldname newname
VMS:   rename oldname newname
```

Under UNIX there is an additional command for giving a file an alternate name, which can be used when this file is addressed by TUSTEP. The command's syntax is:

```
UNIX:  ln name alternname
```

This has the advantage that the file's old name can still be used if it is to be addressed by programs other than TUSTEP.

Regardless of whether a system variable has been previously defined for the carrier specification or whether it is automatically defined by TUSTEP, the user should make sure that each of the corresponding directories actually exists. If not, it must first be created at the operating system level. If the operating system so permits, a directory specified by the project name can also be created with the command #CREATE (see page 79).

When working with temporary (scratch files a few additional points should be kept in mind. In TUSTEP, temporary files are named by the same conventions that apply to permanent files. However, at the system level the files are assigned a name different to that used within TUSTEP. To ensure that all scratch files are arranged in the same directory, the project name defined by the system variable TUSTEP_PRJ is always used in place of the project name specified or pre-set for this file when its name is converted. In addition, the names of scratch files are replaced by a standardized one in order to avoid conflicting names and to help identify scratch files as such at the operating system level. Under DOS this standardized name

takes the form mmmnnnnn.SCR, under UNIX and VMS the form SCR.mmmnnnnn, where mmm is the number of the TUSTEP session and nnnnn the running number assigned to the scratch file.

For example, if scratch files are to be located in a directory called /tmp, the system variable TUSTEP_SCR must be defined with the value /tmp. Then, all scratch files will be placed in the directory /tmp/user, assuming that the system variable TUSEP_PRJ has been defined as user, as shown in the above examples.

For the special case in DOS where only the letter of a drive is assigned to the system variable used to specify the carrier, with no other directories being specified, this process can be simplified. Here the letter can be directly specified as the carrier instead of being specified as a system variable.

System variable TUSTEP_LIB

The system variable TUSTEP_LIB must contain the carrier specification for all TUSTEP programs and related files. Under UNIX and VMS, the project assumed for these files and programs is always tustep, under DOS it is always TUSTEP.LIB. Therefore, if they are located in the directory /home/tustep or, for DOS, in the directory C:\TUSTEP.LIB, TUSTEP_LIB must be assigned the value /home or (for DOS) the value C:

System variable TUSTEP_MDS

The system variable TUSTEP_MDS must be given the value DIALOG when TUSTEP is used in dialogue (i.e. interactive) mode. The value BATCH is required for this variable when TUSTEP is used in batch mode.

System variables TUSTEP_SET, TUSTEP_RST

These two system variables are only used under UNIX. The contents of the system variable TUSTEP_SET is used to control the keyboard and screen settings. The system variable TUSTEP_RST records the keyboard and screen mode set before calling up TUSTEP so that this mode can be reactivated upon exiting TUSTEP. Both variables should only be defined by TUSTEP itself.

System variable TUSTEP_LPR

The system variable TUSTEP_LPR is only used under VMS and UNIX. It defines the operating system instructions used for printing in conjunction with the TUSTEP commands #MANUAL, #PRINT und #JOURNAL. Prior to executing the operating system instructions, the place keepers contained therein are replaced by TUSTEP with the corresponding values. The following place keepers are available for definition:

<DEVICE> for DEVICE secification
<COPIES> for COPIES specification
<USER> user name set with the command #DEFINE
<SUPPLEMENT> Contents of the system variables for the SUPPLEMENT

specification

<FILE> Name of the internally-created file containing printer control codes that are to be sent to the printer. In case this place keeper is not present in the operating system instruction, the name of the file will be inserted at the end of the operating system instruction.

System variable TUSTEP_SUB

The system variable TUSTEP_SUB is only used under VMS and UNIX. It defines the operating system instruction for executing a batch job with the command #EXECUTE. Before executing the operating system instruction, TUSTEP replaces the place keeper <SUPPLEMENT> with the contents of the system variables given in the SUPPLEMENT specification. This can be used, for example, to supplement a number of user options.

System variable TUSTEP_MEM

For each TUSTEP session a scratch file is created which records information, e.g. which files have been opened, that is relevant to the session in question. As with all scratch files, this file is located in the directory specified by the system variables TUSTEP_SCR and TUSTEP_PRJ. Since any number of sessions may exist at the same time, TUSTEP has to know which file contains the information pertaining to the session currently in progress. To do so, TUSTEP defines the system variable TUSTEP_MEM, which contains a reference number for this file which is written as an 8-digit number mmm00000, where mmm stands for the session number. This number is at the same time part of the file's standardized name given to it for use outside of TUSTEP. The standardized name of such files has under DOS the form mmm00000.SCR, under UNIX and VMS the form SCR.mmm00000. This gives the user the possibility of continuing a session that has not yet been ended with the command #RESET: here the system variable TUSTEP_MEM is set to the appropriate value before TUSTEP is called up. The user should make sure that the settings of the other system variables correspond to those in force at the time when the session to be continued was begun or interrupted.

System variables TUSTEP_MT1, TUSTEP_MT2

If access is to be made to a magnetic tape the system variable TUSTEP_MT1 must contain the name of the tape unit (e.g. under UNIX /dev/rmt0, under VMS mua0). The system variable TUSTEP_MT2 must contain the name of the second magnetic tape unit if a second magnetic tape unit is involved for purposes of copying from one tape to another.

The system variables TUSTEP_MT1 and TUSTEP_MT2 can also be replaced by system variables having another name. In this case, however, the names of these system variables must be specified in the commands that access the magnetic tape.

The names assigned to magnetic tape units depend upon the conventions established for each computing center or system administrator.

Magnetic tape operations

Under VMS a tape unit must first be assigned before a magnetic tape can be used. Under UNIX this is also necessary when working with some computers. The appropriate operating system instruction depends on the system being used. The relevant instruction might appear as follows:

```
UNIX:  tpmount -b -a /dev/rmt0 -s TAPE number
VMS:   mount/foreign mua0: number $TAPE$
```

The number specified here corresponds to the tape number. This reserves the tape unit for this purpose until it is slated for another use. This should be carried out as rapidly as possible, so that the tape unit is available for other users. Here the operating system instruction might appear as follows:

```
UNIX:  tpunmount -s TAPE
VMS:   dismount mua0:
```

Please consult the respective computing system or system administrator as to whether these two instructions are mandatory.

System variables and TUSTEP sessions

As described in the chapter "TUSTEP Startup" (page 48), a TUSTEP session starts with the callup of TUSTEP and ends at the completion of the commando #RESET. If a TUSTEP session is interrupted and later continued, it is of vital importance that the system variables used when continuing TUSTEP have the same values as those in force at the start or interruption of the TUSTEP session.

The system variables TUSTEP_SCR, TUSTEP_PRJ and TUSTEP_MEM are used to identify a TUSTEP session in that together they refer to the file containing the relevant information.

If not defined when TUSTEP is called up, the system variables TUSTEP_SCR and TUSTEP_PRJ will use the default settings.

For UNIX and VMS: if the system variable TUSTEP_MEM has not been defined upon startup, a new TUSTEP session will be started. Here a check is made as to which TUSTEP sessions, i.e. which files having the name SCR.mmm00000 are present in directory defined by the system variables TUSTEP_SCR und TUSTEP_PRJ. The new TUSTEP session is thus assigned the smallest possible number that is not yet used. If the system variable TUSTEP_MEM has already been defined (valid here are only those values consisting of an 8-digit number whose last 5 digits are 0), the corresponding TUSTEP session will be continued upon calling up TUSTEP. If, however, the file having the corresponding name does not exit, a new TUSTEP session will be started. This session will then be assigned the number contained in the system variable TUSTEP_MEM.

Since the system variable is lost upon each logoff of TUSTEP, the system variable TUSTEP_MEM is no longer defined. In this case, a new TUSTEP session will be started when TUSTEP is called up, unless the system variable has not been explicitly redefined.

If the system variable TUSTEP_MEM has not been defined at the DOS level, it will be given the standardized value of 00000000. This means that when calling up TUSTEP, the TUSTEP session will be continued with the number 000. If it does exist, a new session will also begin with the number 000.

A TUSTEP session can always be started or continued with a particular number by setting the system variable TUSTEP_MEM in the corresponding manner before calling up TUSTEP. A simpler possibility of continuing a TUSTEP session where the system variable TUSTEP_MEM is to be set to its corresponding value is described in the chapter "TUSTEP Startup" (page 49).

T U S T E P S t a r t u p

Survey:

Starting TUSTEP

TUSTEP is called up at the operating system level with the following command:
tustep

At some computing centers TUSTEP cannot be started as described above due to organizational reasons. In this case, the starting command may be obtained from the information bulletin of the respective computing center.

Before calling up TUSTEP you may wish to define the various system variables for configuring the necessary TUSTEP functions. The relevant information is detailed in the chapter "System environment" (page 36) However, the default settings have been set, or are set during installation, in such a way that TUSTEP is fully operative when it is called up.

After being called up, the TUSTEP program now awaits TUSTEP commands for the further processing or organization of data.

Initializing TUSTEP

TUSTEP is initialized at the start of every TUSTEP session. This means that permanent files are opened and the temporary files needed by TUSTEP for internal purposes are created.

In addition, TUSTEP user-information data TUSTEP user information are shown at the first callup. This information can also be obtained with the command #INFORM (cf. page 111).

TUSTEP then checks to see whether a start file already exists (see page 50). If so, the commands contained in this file are executed (any other type of data are not allowed in this file).

Starting a TUSTEP session

A new TUSTEP session is started when TUSTEP is called up and thereby initialized after a new login. Under DOS a new TUSTEP session is begun only if no other session exists.

Interrupting a TUSTEP session

Should it be necessary to interrupt TUSTEP in order to return temporarily to the operating system level, type "*EOF" at the TUSTEP command prompt.

To interrupt a session for an extended period of time under UNIX and VMS, the user can log out, under DOS the PC can be turned off.

Resuming a TUSTEP session

An interrupted TUSTEP session can be resumed by restarting TUSTEP in the same manner as described above.

If, however, a session can (or should) only be resumed after a new logon, the following operation system instruction can first be given:

tustep @

Specifying "@" after the TUSTEP command name produces a list of all existing TUSTEP sessions. Besides containing the number of each TUSTEP session, the list also displays the time when a session was begun and when it was last used. In addition the command #DEFINE can be used to specify a defined user name and the corresponding user-defined default settings.

TUSTEP is then called up with the following operating system command:

tustep n

where n is the number of the TUSTEP session that is to be resumed. If this TUSTEP session is interrupted once again, it can be resumed by calling up TUSTEP without any additional specifications. If this session is interrupted in oder to continue working with a different TUSTEP session, the number of the desired session should be specified. A new TUSTEP session is started by specifying a new number.

Terminating a TUSTEP session

A TUSTEP session is terminated by the command #RESET (see page 164).

Start file

The start file consists of any commands that are meant to be automatically executed at the start of a TUSTEP session (but not for resuming a session). This file may not contain any other data other than these commands. The start file has the name *TUSTEP.INI and can be created and edited just like any other TUSTEP file.

Some of TUSTEP's default settings have been established for working with a mainframe computer and may not be the ideal configuration in every case, for example, when working with a PC under DOS. The following example shows three basic commands that can be specified in the start file for a PC under DOS. In general, these commands require no additional alterations by the user.

```
#define,code=ibmpc
#wipe,off
#journal,portioned
```

These commands have the following effect:

- For screen output, it is assumed that the screen is capable of displaying not only the characters in the TUSTEP 7-bit character set, but special characters as well, e.g. German umlauts and the double ss character. This setting is mandatory under DOS when umlauts and a double s are used on a German keyboard.

  If accented letters are also to be displayed in the Editor, the code specification should not be "ibmpc", but rather "cp437" or "cp850", depending on which "code page" has been specified in the DOS configuration. If in doubt, both settings can be experimented with; the correct setting can be then easily ascertained by comparing the respective screen displays. However, never specify a code that does not correspond to the DOS configuration.

- When erasing files, their data are not to be erased beforehand. For reasons of data security, TUSTEP normally deletes the content of each file that is erased. This prevents the restoration of a file's data. If this type of data security is not required, it can be switched off with this specification; this also spares the user the extra time required to wipe files. This specification is also mandatory if the contents of an erased file are to be restored at the operating system level, e.g. with the DOS "undelete" command (or with a similar utility program).

- During execution of a command sequence, the execution listing is to be displayed screen by screen. This makes it easier for the user to examine it and keeps it from scrolling off the top of the screen before it can be read. This setting also lets the user make corrections in the command input line (similar to working with the Editor), as well as reshowing previously-entered commands on the screen, where they can be resent to the computer in either their old or newly-edited forms. (cf. Entering commands, page 57).

It is advisable to include Editor settings in the start file so that the function keys, Editor macros and any other user needs are automatically defined. This is done by including the following command in the start file. Since the exact specifications for this setting largely depend on the user's personal needs and habits, the example below is merely given to illustrate how such definitions may be written.

```
#edit,definitions=*

c --- character- and string-groups ---

>0z="()*,-./\>><<?
>0s=/%>0/%>0>0/
...

c --- Tabulator positions ---

t,,7 11 17

c --- Funktion keys ---

f21=x #for,<editor>,,lq,+ #pause #pri,,lq
...

c --- Parameters for search instructions ---

p1,aa=/*/
...

c --- Editor macros ---

y,d="Tübingen,", date_3
...

c --- Options (including color screen settings) ---

o=00501904 17 7E 1E 67 37 3E 17 7E 1E 67 6E 01 02 00 ...

*eof
```

For more details, see the command #EDIT (Seite 91) the chapter "Editor" (starting on page 174).

C o m m a n d s

Survey:

General information

Each command begins with a hash mark (#) followed by the command name. The command name can entered in abbreviated form (with or without a period), but must contain at least the first three letters of the command.

The command is usually followed by the specification values for the respective specifications. Each specification has its own specification name. Specifications are separated from each other and from the command name by a comma. For some specifications, more than one specification value is allowed, with each value separated from each other by an apostrophe.

A specification may take the form of a value assignment. In this case, the specification name is followed by an equals sign and the specification value. Specification names can also be abbreviated as long as the abbreviation is not ambiguous for the respective command.

Specifications for a command are entered in the same order as listed in the command's description. The specification name (and equals sign) may therefore be omitted if this sequence is adhered to. If the specification name (or its abbreviation) is given, the normal sequence may be ignored. These two ways of entering specifications may be combined within the same command.

If no value is given for a specification, the default setting will be used. Default values are marked by an asterisk (*) in the description of each command. Default settings may not be redefined by the user; however, user-defined commands (macros) may be defined with their own default settings.

A specification value may also be replaced by a previously defined TUSTEP variable (cf. command #DEFINE page 85). The variable's name must be placed in pointed brackets (<>).

A command can be written in one or in several lines (maximum: 80 characters per line). A new line may be started after each comma, apostrophe or equals sign. It is not necessary to mark continuation lines in commands.

Commands may be written in either lowercase or uppercase letters. Blanks are insignificant and may be freely inserted as desired.

The start of a comment is indicated by the number sign "#" combined with an equals "=", a plus "+" or a minus sign "-". The end of a comment is indicated by the next number sign located in the same line or a number sign at the beginning of a subsequent line. sign. Comments designated by a "=" will be recorded in the journal (i.e. shown interactively on screen assuming their output has not been suppressed (see #JOURNAL command on page 116). Comments designated with a "+" are always listed in the journal; those designated with a "-" will not appear in the journal.

Data to be processed or evaluated by commands are contained in files. In the case of some commands, however, the data may also be supplied directly after the command. In this case, an

asterisk is given for the corresponding specification instead of a file name. The data supplied after the command must be terminated by a line beginning with *EOF.

In some commands the situation may arise where data following the command contain lines starting with *EOF. To avoid having such lines interpreted as the line which marks the end of data input, such lines must be marked with an additional asterisk, i.e. *EOF* instead of *EOF. When the data are read, the marker (i.e. second asterisk) will be automatically removed. If such lines are part of nested commands, it may be necessary to mark an *EOF with more than one asterisk. A asterisk will be removed with each successive pass until an unmarked *EOF remains, which will be interpreted as the end of data input.

In the following example, a file with the name TXT will be created. Then the editor will be called up to edit the file TXT (in this case to add text to the file). In the next step, the contents of the file TXT will be formated for a HP Laserjet Plus printer (GER = printer output device, from the German word Gerät"). The page number will be centered between two minus signs at the top of each page (KT = "Kopftext" or header text). Finally, the formated data will be printed on a HP LaserJet Plus printer.

```
#CREATE,TXT,SEQ-P #EDIT,TXT,T
#FORMAT,TXT,ERASE=+,PARAMETER=*
GER       HP+
KT        / @z - XXX -/
*EOF
#PRINT,,HP+
```

To write the above command sequence into a file called CMD using the command #CONVERT, for example, the following commands would be given:

```
#CONVERT,*,CMD,0,+
#CREATE,TXT,SEQ-P #EDIT,TXT,T
#FORMAT,TXT,ERASE=+,PARAMETER=*
GER       HP+
KT        / @z - XXX -/
*EOF*
#PRINT,,HP+
*EOF
```

Here an asterisk must be added to the *EOF which terminates the parameters for the command #PRINT (in the line preceding #PRINT) so that the command #CONVERT does not interpret it as marking the end of data input. This marker asterisk will be removed by #CONVERT so that the file CMD will contain the line *EOF. The *EOF located after #PRINT terminates the data for the command #CONVERT and will not be written to the file CMD.

Entering commands When the prompt

Enter command >

appears, the TUSTEP program awaits commands which tell it which action it should perform. Once a command is typed in, it must then be sent to the computer by pressing the ENTER key, upon which the command will be interpreted and executed.

If a command line ends with a comma, an apostrophe or an equals sign, it will be assumed that the command is yet incomplete and will be continued. The prompt

Enter specifications >

means that the command's related specifications are being awaited. If at this point there is nothing to add to the command, an empty input line may be sent by pressing the ENTER key.

Some programs expect additional data (e.g. parameters) immediately after being called up. Depending on their specific function, these data are requested by various input prompts. Each line must be sent individually with the ENTER key. The end of data input is signaled to the program by sending a new line starting with *EOF.

The 20 most recently entered lines are stored in a buffer. They can shown on screen as a complete list for the user's information, or retrieved on screen individually where they can be sent to the computer after making any necessary alterations.

To aid the user in command input, there are a number of control commands (i.e. editing keys) at his disposal. Each control command has a short name which can be referred to in the tables listed on pages 230 to 239. Listed in these tables are the key or key combinations required to activate each control command for most major of keyboard types.

However, these control commands will not have the effects described below unless the journal mode has been set to either "portioned" or "continuous" with the command #JOURNAL (see page 116). If this has not been done, or if the mode has been set back to "system", the only control commands at the user's disposal are those available at the operating system level.

RESHOW "Reshow commands"

Lists the last 20 command lines entered.

CUR_UP Cursor up

Displays the most recent line entered. Can be used to scroll backwards through the list of the 20 most recent lines entered.

CUR_DN Cursor down

Displays the line that was entered after the line presently shown. After scrolling backwards with CUR_UP this key scrolls forward through the list of the 20 most recent lines entered.

CUR_RI Cursor right

Moves cursor one character to the right.

CUR_LE Cursor left

Moves cursor one character to the left.

TAB "Skip to next tabulator"

Moves cursor to the next tab stop. Tabulator positions are 11, 21, ..., 71.

SKP_BEG "Skip to start of line"

Moves cursor to the beginning of the command line.

SKP_END "Skip to end of line"

Moves cursor to the end of the command line.

SKP_WORD "Skip to next word / end of line"

Moves cursor to the beginning of the next word. If the cursor is already located at the last word, it moves to the end of the line. And if the cursor is already at the end of the line, this moves it to the beginning of the line. A "word" is any character string surrounded by a blank and/or comma.

TGL_INS "Toggle insert mode / replace mode"

Switches back and forth between replace mode and insert mode. In replace mode, existing characters will be overwritten; in insert mode, newly entered characters are inserted at the present cursor position, with any characters located at or to the right of the cursor being moved to the right.

DEL Delete: "Delete character"

Deletes the character at the present cursor position, moving all remaining characters in the line one position to the left.

BSP Backspace: "Delete character"

Deletes the character to the left of the current cursor position, moving all remaining characters in the line one position to the left. The cursor also moves one position to the left.

TGL_DEL "Toggle delete mode / backspace mode"

Switches back and forth between delete mode and backspace mode. In delete mode the DEL key has the effect described in DEL above. In backspace mode the DEL key has the same effect as BSP.

CLEAR "Clear line"

Deletes input line

ENTER "End of input"

The command or data entered in the command line are now sent to the computer.

CR Carriage return: "End of input"

Same as ENTER.

REFRESH "Refresh line"

Restores the contents of the input line that have been disrupted by system messages, transmission line interference, etc.

CANCEL "Cancel input" / "terminate TUSTEP"

- when entering data: terminates data entry; any data still located in the input line will be ignored.
- otherwise: ends (interrupts) TUSTEP session; any data still located in the input line will be ignored.

In addition to the keys or key combinations described here to carry out these control commands, the user may also use the fuction keys. For a list of default definitions and instructions on how to redefine function keys, refer to the command #DEFINE on page 83).

For repeatedly-used commands it is advisable to write them to a file with the help of the Editor. You can then execute the command sequence whenever necessary with the command #EXECUTE (see Page 97) without having to type them each time. In this case, you merely have to to specify the name of this file in the command #EXECUTE.

The same applies to commands which require the subsequent input of data, e.g. parameters. By having such commands placed in a file, any typing error which makes it necessary to repeat the command can be corrected in the file where it occurs (instead of having to reenter all parameters by hand) and the command #EXECUTE can then be repeated.

The most advantageous method of recording such command sequences is to save each command sequence as a segment in a single segment file instead of saving each one in a file of its own.

This is done in the Editor, where each command sequence, after a satisfactory trial run, is written as a segment to a segment file with the Unload instruction (see page 187). The Editor is also used to correct a segment, where it is copied to a temporary file with the Load instruction (see page 186), corrected, and then copied back to the segment file with the Unload instruction. A command sequence located in a segment can also be executed with the #EXECUTE command. Here the name of the segment file is followed by the name of the individual segment.

If a slight modification must be made to a command sequence before it is executed, it can be given place keepers (registers) which are then replaced by the desired specifications when #EXECUTE is called up. Advanced users, however, are encouraged to use command macros (see "Macros" chapter starting on page 266). When used in macros, command sequences can be modified in a variety of ways besides merely the specifications present when the macro is started. For instance, they can also be dependent on responses given to questions contained in the macro.

Interrupting command execution

It is often advisable to interrupt the execution of a program, command sequence or Editor instruction. (For example, if the wrong file has been specified for input.) This is carried out by the control command INTRPT. Please refer to the corresponding key combination for your particular keyboard as listed in the tables on pages 230 to 239.

Ater the operating system has informed TUSTEP of the interupt, TUSTEP responds with the prompt:

PROGRAMM INTERRUPTION - Enter instruction >

Five different specifications are possible here, each consisting of a single letter. They determine how the interrupted program is to proceed.

1) A (Abort) The interrupted program is to be ended (aborted) immediately. Any remaining commands are to be executed afterwards, providing that instruction C has not already been entered.

2) H (Hold) Upon completion of the interrupted program (but before performing any subsequent commands), priority commands priority commands shall be executed as entered individually (!). If an empty entry line is sent instead of a priority command, the interrupted command sequence will be continued.

3) C (Cancel) Any remaining commands will be canceled, i.e. not executed.

4) I (Interrupt) The Editor instruction which is presently being executed will be interrupted, with a prompt appearing for the next Editor instruction. In case no Editor instruction is being processed at the time the interruption is made, this instruction has the same effect as instruction G.

5) G (Go) The interrupted program will be resumed.

The instructions H and C require an additional instruction immediately. Thus these two instructions can be combined with one of the other remaining instructions.

There are other ways of interrupting a command sequence in progress (this option does not apply to a program or Editor instruction being executed) that are offered by the commands #PAUSE (see page 155), #ERROR STOP (see page 96) and #MACRO (see page 127 as well as the chapter "Macros" page 266).

Calling a program

CALL

Command:

#CALL

Specifications:

PROGRAM	= name		Name of the program to be called
READ	= n:file		Logical number n to be assigned to the specified file. The file must be a TUSTEP file and must be opened for reading or writing. More than one specification value is allowed.
	= -	*	No logical number is to be assigned to a file
WRITE	= n:file		Logical number n to be assigned to the specified file. The file must be a TUSTEP file and be opened for writing. More than one specification is allowed.
	= -	*	No logical number is to be assigned to a file
CARRIER	= -STD-	*	The program to be called is located on the carrier specified at the operating system level. Under DOS, UNIX and VMS, the carrier for programs that are called up with this program has been specified with the system variable TUSTEP_DSK.
	= name		Under DOS, UNIX and VMS: name of the system variables containing the path of the program to be called. Under DOS the drive letter can be specified here as long as the path does not contain any directory names.

Features:

With this program, FORTRAN programs may be called which use TUSTEP subroutines (at least the subroutines INIT and EXIT). If TUSTEP subroutines are used for input and output in such a program, a file name must be assigned to each logical number used in this program.

The syntax used for the program name depends on the operating system being used:

BS2000: filename

DOS: TUSTEP.PRG*Execname

MVS: filename(membername)

UNIX : filename

VM/CMS: modulename

--- VMS : filename

For the TUSTEP subroutines there is a special description with the name SR (cf. command #MANUAL on page 128).

Closing files

CLOSE

Command:

#CLOSE

Specifications:

FILE	= file		Name of the file to be closed. More than one file name is allowed.
	= -	*	No files to be closed.
	= +		Files selected with the specifications PROJECT/CARRIER/POSITIVE/NEGATIVE are to be closed.
PROJECT	= name		Name of the project whose files are to be closed.
	= +		Files from the current project are to be closed.
	= -STD-		Files of the project specified at TUSTEP initialization are to be closed.
	= -	*	No files of any project are to be closed.
CARRIER	= name		Under DOS, UNIX and VMS: name of the system variable containing the path specification of the file to be closed. Under DOS the drive letter can also be specified when the path does not contain any directory names.
	= -	*	No file of any carrier (disk or disk drive) is to be closed
POSITIVE	=	*	No positive selection by character strings occurring in file names
	= ...		Only those files are to be closed whose filenames contain at least one of the specified character strings.
NEGATIVE	=	*	No negative selection by character strings occuring in file names
	= ...		Only those files are to be closed whose filename does not contain any of the specified character strings.
WIPE	= +		Data contained in temporary files to be closed and thus later erased are to be erased beforehand.

= - Data contained in temporary files to be closed and then erased need not to be erased.

Features:

This command is used to close files. This prevents the inadvertent access of subsequent TUSTEP programs to these files (e.g. due to typing errors).

To close all files opened of a particular project, simply enter "+" for the specification FILE and the name of the project for the specification PROJECT. Likewise, all opened files of a data carrier can be closed by entering the name of the disk or disk drive in the specification CARRIER. If both specifications have been given, all opened files of the given project on the given data carrier will be closed.

To close all files whose name contains a particular character string, these character strings should be entered in the specification POSITIVE which must occur at least once in the file name for the file to be closed. The specification NEGATIVE can be used to specify character strings which must not be contained in the file name for the file to be closed. For a full description of this feature, refer to the command #LIST on page 121.

Temporary files (scratch files) will be automatically erased, and storage space will be made available for other files. The specification WIPE can be used to specify whether the data in such files are to be erased beforehand. If nothing is specified here, the data will be erased if no other mode has been selected with the command #WIPE (cf. page 173).

Special applications for the MVS version

Permament (cataloged) files which are closed become immediately accessible for other jobs. This is important when the same file is to be accessed by more than one job, since a permanent file can be opened for reading by any number of jobs but opened for writing by only one job.

Listing comparison results

COLLATE

Command:

#COLLATE

Specifications:

SOURCE	= file		Name of the input file containing the basic text
MODE	= -STD-	*	Each CORRECTION file contains the correcting instructions for a single text version.
	= CUMULATED		The CORRECTION file contains the correcting instructions for all text versions.
ERASE	= -	*	If the LISTING already contains data, they are to be retained.
	= +		If the LISTING already contains data, they are to be erased beforehand.
PARAMETER	= file		Name of the file with the parameters
	= *		Parameters follow the command and are ended by *EOF.
CORRECTION	= file		Name of the file containing the correcting instructions (with correction key). If MODE=-STD-, more than one file name is allowed.
m,2			
LISTING	= -STD-	*	The generated list is to be written to the standard LISTING.
	= file		Name of the file to which the generated list is to be written

Features:

With this command, a listing of the differences between one or several text versions and a basic text can be generated for printing in lines synoptic to the basic text. The differences must be provided in form of correcting instructions containing a correction key, as generated by the TUSTEP programs COMPARE or PRESORT. The deviant words (variants of the different text versions) are listed in synoptical lines under the respective words of the basic text. Identical text passages found between the basic text and the other text versions as well as between the variants themselves are marked as such.

Description:

For this command, there is a special description with the name CO (cf. command #MANUAL page 128).

Comparing versions of the same text

COMPARE

Command:

#COMPARE

Specifications:

VERSIONA	= file		Name of the input file containing text version A
VERSIONB	= file		Name of the input file containing text version B
MODE	= T		The differences between the texts are to be listed
	= C		The generated correcting instructions are to be listed
	= ...		Printer type used to print out the list of text differences. The type of printer available depends on the actual computer being used To obtain a list of these, use the command #LIST,PRINTERS (see page 121).
			The printer type can also be entered as a parameter specification.
ERASE	= -	*	If the CORRECTION file or the LISTING contain data, the data is to be retained.
	= +		If the CORRECTION file or the LISTING contain data, the data is to be erased beforehand.
PARAMETER	= -	*	No parameters
	= file		Name of the file containing parameters
	= *		Parameters follow the command and are ended with *EOF
CORRECTION	= -	*	Do not record correcting instructions
	= file		Name of the output file to which the correcting instructions are to be written
LISTING	= -	*	No printer output for listing differences
	= +		The generated listing is to be written to the journal (i.e. the screen when in interactive mode).

= -STD- The generated listing is to be written to the standard LISTING file.

= file Name of the file into which the generated listing is to be written.

Features:

This program is used to compare two text versions (A and B). The differences are listed in the file entered in the specification LISTING. In addition, these differences can be written to the file entered in the specification CORRECTION in the form of correcting instructions, using the same conventions for correcting instructions as required for the program CORRECT. If this file is used to correct version A, version B will be the result (the line division and the record numbers will be those of version A).

The line division of the versions may be entirely different. Omissions and insertions up to the length of one typed page can be identified by the program automatically (although this process consumes a disproportionately large amount of CPU time). The program can handle ommissions and insertions of any length, provided these are identified and indicated by the user. It is also possible to compare only certain ranges of the files.

Description:

For this command, there is a special description with the name CM (cf. command #MANUAL page 128).

Converting/encoding data/files

CONVERT

Command:

#CONVERT

Specifications:

SOURCE	= file		Name of the file containing the data to be converted
	= -STD-		The data to be converted are contained in the standard TEXT file
	= *		The data to be converted follow the command #CONVERT and are ended by *EOF.
DESTINATION	= file		Name of the file to which the converted data are to be written
	= -STD-		The converted data are to be written to the standard TEXT file
MODE	= ...		The data are to be converted using the mode specified here (see below).
	= -STD-	*	corresponds to the specification +1 if the SOURCE file is a system file and the DESTINATION file is a TUSTEP file. -1 if the SOURCE file is a TUSTEP file and the DESTINATION file is a system file. 0 in all other cases.
ERASE	= -	*	If the DESTINATION file already contains data, they are to be retained.
	= +		If the DESTINATION file already contains data, they are to be erased beforehand.
KEY	= -	*	The data are not to be encoded or decoded.
	= file		Name of the file containing the key for
	= *		The key for encoding or decoding follows the command #CONVERT and is ended by *EOF.
CODE	= -		No additional code conversion however, data from a system file is copied into a TUSTEP file or vice versa, the code of the data is to be converted in the same

way for input and output on the display device.

= xx:yy — The character with the hexadecimal code xx is to be converted to the character with the hexadecimal code yy. More than one pair of codes is allowed.

= TUSTEP — The data code is to be converted in the same manner used for input from the display device. Additional codes or exceptions to the conversion mode can be specified by additional hexadecimal code pairs (xx:yy). If the data contain unknown codes, these codes will be converted to character strings having the form "#[xx]".

= SYSTEM — The code of the data is converted in the same manner as used for output to the display device. Additional codes or exceptions to the current conversion mode can be specified by additional hexadecimal code pairs (xx:yy). Any character strings in the data having the form "#[xx]" will be converted into a single character.

= -STD- * corresponds to the specification
TUSTEP if the SOURCE file is a system file and the DESTINATION file is a TUSTEP file.
SYSTEM if the SOURCE file is a TUSTEP file and the DESTINATION file is a system file.
\- in all other cases.

= ASCII — The data are to be converted from EBCDIC to ASCII. The code conversion table (cf. page 354) may be modified by defining additional pairs of xx:yy codes.

= EBCDIC — The data are to be converted from ASCII to EBCDIC. The code conversion table (cf. page 354) may be modified by defining additional pairs of xx:yy codes.

NL = -STD- * The usual codes of the respective operating system (DOS: CR LF = 0D0A, UNIX: LF = 0A) are to be used as the delimiter character between data records in system files.

= xx — The character having the hexadecimal code xx is to be used as the delimiter character between data records in system files.

Warning:

Encoded data may not be saved to system files. Depending on the respective operating system, this could prevent the data from being decoded later, or could result in undefined errors. In addition, whenever encoding data, make sure that the destination file is either empty or that ERASE=+ has been specified in the command.

Features:

With this command data are converted or recoded according to the rules given for the specifications MODE and CODE. If so desired, they will be additionally encoded or decoded according to the key given in the specification KEY.

The files given for the specifications SOURCE and DESTINATION can be of the same type (system file or TUSTEP-file) or of different types. Thus it is possible to copy the contents of a system file to a TUSTEP file and vice versa.

If converting with CODE=TUSTEP (this is the default setting for copying a system file to a TUSTEP file) the data are transcribed in the same manner as defined by the command #DEFINE (see page 83) for input from the display device. Characters not defined for the code selected will be converted to the form "#[xx]", where xx stands for a character's hexadecimal code.

Conversely, when converting with CODE=SYSTEM (this is the default setting for copying a TUSTEP file to a system file) the data are transcribed for the output to the display device. Character strings having the form "#[xx]", where xx is a hexadecimal code from 00 to FF, will be converted into the character corresponding to this code.

An encoded file is decoded by reconverting the file with the same key used for encoding. The key is a character string at least 40 and at most 400 characters long, and is defined by the user.

The order in which the data are processed according to the specifications MODE, KEY and CODE depends on the value given for the specification MODE. The order is
CODE - MODE - KEY for modes +0, +1, +2, +4, XT, XB, TK, but
KEY - MODE - CODE for modes -0, -1, -2, -3, -4, TX, BX, and KT.
If MODE=0, only one of the specifications KEY and CODE may have a value other than "-" ("-STD-" may also be given for CODE if this corresponds to CODE=-).

Modes:

0 No conversion of data and no checking for illegal characters

+0 Data are not to be converted; check only whether individual characters (not character combinations) are accepted by TUSTEP.

-0 Data are not to be converted; check only whether individual characters (not character combinations) are accepted by TUSTEP.

+1 Characters encoded with the escape character "^" followed by a 7-bit ASCII character from the TUSTEP character set are converted into the respective TUSTEP characters (cf. code tables on pages 298 und 299).
Example:
"Er l^oste die TUSTEP-^Ubungsaufgabe gro^sartig"
is converted to
"Er löste die TUSTEP-Übungsaufgabe großartig".
("He solved the TUSTEP exercise excellently.")

-1 Reverses n=+1. Characters from the 8-bit TUSTEP character set are replaced by an input code consisting of the character "^" and a character from the 7-bit TUSTEP character set (cf. character tables on page 299).

+2 As in n=+1; in addition, if nothing else is indicated by the shift character "<", all alphabetic characters are converted to lowercase letters. If the data contain the character "<", alphabetic characters following this character are converted to uppercase characters. The character ">" cancels the effect of the character "<", i.e. alphabetic characters are converted back to lowercase letters. The characters "<" and ">" are not transfered. If the characters "<" or ">" are to be represented, they must be encoded by "<<" or ">>".

Example: The sentence
"<E>R L^OSTE DIE <TUSTEP>-<^U>BUNGSAUFGABE GRO^SARTIG"
is converted to
"Er löste die TUSTEP-Übungsaufgabe großartig".

-2 Reverses n=+2. The shift-characters "<" and ">" are inserted before and after one or more uppercase characters. If the characters "<" and ">" are encountered here, they will be doubled. In addition, characters from the 8-bit TUSTEP character set will be replaced by the input code consisting of the character "^" and a character from the 7-bit TUSTEP character set (cf. character table on page 298).

-3 "<" is inserted before each upper case character. If the characters "<" and ">" are encountered, they will be doubled. In addition, characters from the 8-bit TUSTEP character set will be replaced by the input code consisting of the character "^" and a character from the 7-bit TUSTEP character set (cf. character table on page 298).

-4 Data are to be converted to their hexadecimal form (each character is given its hexadecimal code specification from 00 to FF).

+4 Reverses -4. Every 2 characters will be interpreted as a hexadecimal code specification and converted into the character having the corresponding code. The data must be written as hexadecimal pairs.

TX The text data from the SOURCE file are to be written to the DESTINATION file in data exchange format.

XT The data in the SOURCE file are written in data exchange format and are to be written to the DESTINATION file as text data.

BX The binary data in the SOURCE file are to be written to the DESTINATION file in data exchange format.

XB The data in the SOURCE file are written in data exchange format and are to be written to the DESTINATION file as binary data.

TK The text data in the SOURCE file are to be written to the DESTINATION file in compressed form.

KT The data in the SOURCE file are written in compressed form and are to be written to the DESTINATION as (decompressed) text data.

Restrictions

If a system file has been entered for the specification DESTINATION, it must be empty or ERASE=+ must have been given so that the data in this file are erased. Data may be appended to the end of a file for TUSTEP files only.

Due to the structure of system file, the specification NL can only be used under DOS and UNIX.

COPY

Command:

#COPY

Specifications:

SOURCE	= file		Name of the file containing the data to be copied
	= -STD-		The standard TEXT file contains the data to be copied.
DESTINATION	= file		Name of the file to which the data are to be copied. More than one file name is allowed.
	= -STD-		The data are to be copied to the standard TEXT file.
MODE	= -STD-	*	Retain numbering of records if possible
	= +		Renumber records
	= -		Retain numbering of records in all cases
	= S		DATA file contains sort units
ERASE	= -	*	If the DESTINATION file already contains data, they are to be retained.
	= +		If the DESTINATION file already contains data, they are to be erased beforehand.
PARAMETER	= -	*	No parameters
	= file		Name of the file containing parameters
	= *		The parameters follow the command and are ended by *EOF.
DATA	= -	*	SOURCE file contains records in their entirety.
	= file		Name of the file containing the text part of each record
	= -STD-		The standard DATA file contains the text part of each record
LISTING	= -	*	No output of trace (test protocol)

= +	Trace (test protocol) is to be written to the journal
= -STD-	Trace (test protocol) is to be written to the standard LISTING.
= file	Name of the file to which the trace (test protocol) is to be written

Features:

With this program, files can not only be copied, but also processed in a variety of ways.

Parameters can be used to:

- examine data
- select data
- move and expand individual text parts
- replace character strings
- process calendar dates
- calculate numerical values contained in the text
- control the form of output

Description:

For this command, there is a special description with the name CP (cf. #MANUAL, page 128).

Executing correcting instructions

CORRECT

Command:

#CORRECT

Specifications:

SOURCE	= file		Name of the file containing the data to be corrected
	= -STD-		The standard TEXT file contains the data to be corrected.
DESTINATION	= file		Name of the file to which the corrected data are to be written
	= -STD-		The corrected data are to be written to the standard TEXT file.
MODE	= -STD-	*	Retain numbering of records if possible
	= +		Renumber records (in text mode)
ERASE	= -	*	If the DESTINATION file or LISTING already contain data, these data are to be retained.
	= +		If the DESTINATION file or LISTING already contain data, these data are to be erased beforehand.
PARAMETER	= file		Name of the file containing parameters
	= *		The parameters follow the command and are ended by *EOF.
CORRECTION	= file		Name of the file containing the correcting instructions
LISTING	= -	*	No protocol is to be recorded; only error messages are to be written into the journal.
	= -STD-		The correction protocol is to be written into the standard LISTING.
	= file		Name of the file to which the correction protocol is to be written

Features:

This program is used to correct texts without using the editor. Correcting instructions contained in a file or supplied after the command #CORRECT are used to replace, erase or insert

- lines (e.g. line 2 on page 3)

- words (e.g. the second word in line 3 on page 4)

- characters (e.g. the second character in line 3 on page 4 or the second character of the third word in line 4 on page 5).

A list of the executed corrections is provided if so desired.

Note

The correcting instructions for this program are generally created with the command #COMPARE (see page 68) and then adjusted accordingly.

Description

For this command, there is a special description with the name CR (cf. #MANUAL, page 128).

Creating files and projects

CREATE

Command:

#CREATE

Specifications:

NAME	= name		Name of the file or project (directory) to be created. More than one file or project name can be specified.
TYPE	= type		Type of file to be created.
	= type-opt		Type of the file to be created, with additional options
			The following values can be given for type:
		* SEQ	TUSTEP file with sequential access
		RAN	TUSTEP file with random access (not yet implemented)
		SDF	system file in system-data format
			The following can be given for opt:
		A	Open file if it already exists
		* T	Temporary file
		P	Permanent file
		F	Fixed record length
			For additional information concerning options, see below.
	= PROJECT		Creates a project (directory) having the name given for the specification NAME
CARRIER	= -STD-	*	The file or project is to be created on a data carrier (disk or disk drive) set by the operating system. Unter DOS, UNIX and VMS, the carrier for projects and permanent files is preset by the system variable TUSTEP_DSK. For temporary (scratch) files the system variable TUSTEP_SCR is used.
	= name		Under DOS, UNIX und VMS: name of the system variables which contain the path specification for the file or project to

			be created. Under DOS, merely specifying the letter of the drive is sufficient here as long as the path specification contains no directory names.
LENGTH	= n		The average length of records is n characters.
	= 60	*	The average length of records is 60 characters.
	= +n/-n		The average record length is n characters longer/shorter than the average record length of the file entered for the specification FILE.
	= +n%/-n%		The average record length is n% longer/shorter than the average record length of the file entered for the specification FILE.
RECORDS	= n		Number of records is n.
	= 100	*	Number of records is 100.
	= +n/-n		The number of records is n records greater (+n) or less (-n) than the number of records in the file given in the specification FILE.
	= +n%/-n%		The number of records is n% greater (+n%) or smaller (-n%) than the number of records in the file given in the specification FILE.
FILE	= file		Name of the file referred to by the relative values (+n/ -n and +n%/-n%) given in the specifications LENGTH and RECORDS.
	= -	*	The values given for the specifications LENGTH and RECORDS are absolute numbers.

Features:

This command is used to create files. New files are time automatically opened for writing (cf. command #OPEN).

TUSTEP programs accept only TUSTEP files (TYPE=SEQ or TYPE=RAN). However, system files (TYPE=SDF) can also be specified in the the program #CONVERT and in the Editor instructions "Load" and "Unload".

The type of file to be created can be defined in more detail by specifying one of the options listed below. These options (single letters) are placed after the type name and separated from it by a hyphen "-" (e.g. SEQ-T).

This command can also be used to create projects (directories) for managing your files. By organizing files under various project names, they can be compiled as logical groups (e.g. all files relating to each task area), giving the user a much clearer overview of all existing files.

Options for all versions:

If no option is entered, the default option T is used. If options are specified, either T or P must be included. Multiple options must be given in the order listed below. No delimiter character is required between options.

A The program will try to open the specified file for writing. If the file does not yet exist, it will be created in accordance with the additional option specified.

T Temporary file (scratch file)

P Permanent (cataloged) file

Additional option for the MVS and VM/CMS versions:

F Fixed record length (for system files only)

Notes

Up to 100 files (excluding those files used by TUSTEP internally) can be opened in a TUSTEP session. This also includes files created with the command #CREATE in the same session. If this limit is reached, existing files must either be closed with the command #CLOSE (cf. page 64) or erased with the command #ERASE (cf. page 93) before any new files can be created.

Hints:

The specifications concerning file size (specifications LENGTH, RECORDS and FILE) are irrelevant for the creation of projects.

When creating files these specifications are required in the MVS version only. Under this operating system, a file will be created according to the specified size and cannot be enlarged later. Should the file become too small, the following steps may be of assistance:

- create a new, larger file with another name
- copy the data from the old file to the new one
  (using the command #RESTORE, see page 165)
- erase the old file
  (using the command #ERASE, cf. page 93)
- change the name of the new file to that of the old file
  (using the command #RENAME, cf. page 163)

In the BS2000 and the VMS version, file size specifications are not required since files are automatically enlarged whenever

necessary. However, it is recommended to specify the anticipated file size, thus causing a physically contiguous area to be allocated to the file. Accessing a file occupying a contiguous area is much more efficient than accessing one which must be enlarged and therefore becomes increasingly fragmented.

File size specifications have no effect in the MS-DOS and VM versions of TUSTEP. Here files are automatically enlarged when necessary.

Restrictions:

A file and a project having identical names cannot be located on the same carrier.

File names having the notation "TUSTEP.xxx" (where xxx stands for any combination of letters or digits) are reserved for TUSTEP and may not be used in creating files. But there are two exceptions to this:

The file name "TUSTEP.INI" is reserved for the start file containing commands to be executed whenever TUSTEP is initialized (cf. page 48). The file name "TUSTEP.USE" is reserved for a file in which user statistics about TUSTEP are recorded (cf. command #STATISTICS page 170).

Defining a macro file, variables, u.a.

DEFINE

Command:

#DEFINE

Specifications:

MACROS	= file		Name of the macro file containing the command macros to be executed later.
	= -	*	Do not switch to another macro file
VARIABLES	= file		Name of the file containing the definitions of TUSTEP variables
	= *		Definitions of TUSTEP variables follow the command #DEFINE and are ended by *EOF.
	= -	*	No TUSTEP variables to be redefined.
PROJECT	= name		Project name assigned to the file names.
	= -STD-		The project name used at TUSTEP initialization is to be assigned to the file names.
	= -	*	Retain defined project name.
USER	= name		User name to be printed on the cover sheets of printouts
	= -	*	Retain defined user name
CODE	= -	*	Retain defined code for input and output on the display device
	= ASCII		Display device uses the international ASCII character set (see table on page 345); it is not necessary to alter the coding.
	= EBCDIC		Display device uses the US-EBCDIC character set (see table on page 353); the coding is to be altered as described below.
	= GERMAN		Display device uses the German ASCII or EBCDIC character set (see table on page 346 and page 353, respectively); the coding is to be altered as described below.

	= DECMCS		Display device uses the DEC multinational character set.
	= IBMPC		Display device uses the IBM-PC character set (see table on page 347)
	= CP437		Display device uses the character set defined by code page 437 (see table on page 348)
	= CP850		Display device uses the character set defined by code page 850 (see table on page 349)
	= ISO8859		Display device uses the character set corresponding to the ISO standard 8859 (see table on page 351)
	= xx:yy		On input from the display device, a character with hexadecimal code xx is to be converted to a character with hexadecimal code yy; before being displayed, a character with hexadecimal code yy is to be converted to a character with hexadecimal code xx. More than one pair of codes may be specified.
FUNCTIONS	= file		Name of the file containing the definitions for function keys
	= *		The definitions for function keys follow the #DEFINE command and are ended with *EOF.
	= -	*	Retain present function key definitions

Features:

This command can be used to:

- establish which file contains the command macros (see page 55) which (in addition to standard macros) are to be subsequently used.

- define TUSTEP variables to be used in commands (see page 55) and in command macros (see the chapter entitled "Macros", starting on page 266).

- redefine the name of the project assigned to the file. redefined. This project name is used unless a different project name is given with a file name in a TUSTEP command.

- redefine the user name printed on the cover sheet of printouts.

- establish rules for altering the character coding for the input and output on the display device. After initialization, TUSTEP assumes that the display device uses the international ASCII character set.

- define function keys as used at the command level (not for the Editor).

Note:

The current settings/definitions can be displayed with the command #INFORM (see page 111).

German keyboard code

If a German keyboard (with umlauts) is used, this command must be given to set the code required for the respective operating system:

BS2000:	CODE=GERMAN	UNIX:	CODE=GERMAN
DOS:	CODE=IBMPC	VMS:	CODE=DECMCS
MVS:	CODE=GERMAN	VM/CMS:	CODE=GERMAN

The DOS page codes CP437 or CP850, which also contain German umlauts, can also be set as an alternative to the IBMPC code setting. If TUSTEP is not accessed on a local PC but rather on a remote computer via a terminal emulation program, the same code can be set as if working with a local computer (depending on the emulation being employed). However, it may be necessary to set CODE=ASCII and to refrain from using umlaut characters on the keyboard.

Should a particular code be set whenever TUSTEP is initialized, the corresponding command can be written in the start file. However, if different screens are employed, this is only appropriate when the same code has been set for all display devices.

If an interrupted TUSTEP session is to be resumed on another screen, it may be necessary to set the proper code for this screen.

Altering codes:

The following table is read as follows: the character on the left is the character whose TUSTEP code is sent to the computer from the display device; the character on the right is the intended character entered at the display device and whose code must therefore be generated from the incoming code.

- For CODE=EBCDIC the following characters are recoded as shown below:

  ! <---> | | <--->]] <---> !

  For a computer with EBCDIC CODE this corresponds to the specification

  CODE=4F:6A'6A:5A'5A:4F

For display devices using the US-EBCDIC character set (which in contrast to the international EBCDIC character set features no square brackets) substitute characters must be used for the following TUSTEP characters:

[= Cent] = broken vertical bar ^ = logical NOT

- For CODE=GERMAN coding of the characters below is altered as follows:

[<---> Ä \ <---> Ö] <---> Ü
{ <---> ä | <---> ö } <---> ü tilde <---> ß

For computers with ASCII CODE this corresponds to the specification

CODE=5B:C1'5C:CF'5D:D5'7B:E1'7C:EF'7D:F5'7E:F3

and for a computer with EBCDIC CODE this corresponds to the specification

CODE=4A:77'E0:9E'5A:AD'C0:B9'6A:DB'D0:EB'A1:DF

For display devices with the German character set, substitution codes must be used for the following TUSTEP characters:

[= ^< \ = ^/] = ^>
{ = ^(| = ^: } = ^)

Restrictions for the specification USER

The user name defined with this command is printed on the cover sheet of printout whenever the cover sheet is generated by TUSTEP. If the cover sheet is generated by the operating system, this user name is not used for this purpose under the operating systems MVS and VMS.

Instructions for defining TUSTEP variables:

Instructions must contain two identical markers in the first two columns. The default marker is the dollar sign ("$"). Besides instructions for defining variables, instructions for comments and messages as well as instructions for redefining markers are also allowed here.

Names of TUSTEP variables must consist of 1 to 12 alphanumeric characters and must begin with an alphabetic character.

There are three ways to define a variable and to assign it a value, or a new value if it has already been defined.

With the instruction

```
$$: variable name = "character string"
```

the character string enclosed in quotation marks is assigned to the TUSTEP variable having the specified name. The character itself may not contain quotation marks.

When the following instruction is used in an interactive session, a message will appear and a response will be expected:

```
$$? "message", variable name = "character string"
```

The response (no longer than one line) will be assigned to the specified TUSTEP variable. If a null response is given or the command #DEFINE is called in a batch job, the character character string given in the instruction will be assigned to the TUSTEP variable.

If more than one character string is to be assigned to a TUSTEP variable, the following instruction can be used:

```
$$: variable name = *
```

The character strings following this instruction and preceding the next instruction (=next line beginning with $$) will be assigned to the TUSTEP variable having the specified name. In this case each line represents one character string.

The instruction

```
$$- comment
```

can be used to insert comment lines between the definitions of TUSTEP variables. The instruction

```
$$+ message
```

is used to list messages in the journal (i.e. the display device in interactive mode).

If messages are to be listed in interactive mode only, the following macro instruction can be used:

```
$$* message
```

This can be used to display information about possible responses and their effects before they are actually activated.

The default instruction-marker "$" located in the first two columns of a line can be redefined with the following instruction:

```
$$= *
```

After this instruction an asterisk is used as the instruction-marker. Any other special character may be used instead of an asterisk (used here as an example). After being changed with this instruction, the marker can be reset to its default value with the following instruction:

```
**= $
```

The use of a marker other than "$" may become necessary when more than one character string is assigned to a TUSTEP variable and one of these character strings begins with a $$.

Instructions for defining function keys

The following description for defining default values for function keys is valid only at the command level (after the "Enter command" prompt). Function key settings in the Editor (after the "Enter instruction" prompt) must be set individually (see page 191).

Function keys can be used at the command level only if the command #JOURNAL has been previously used to set the mode to either PORTIONED or CONTINUOUS (see page 116).

For input at the command level, each of the 20 most recently lines will be stored to buffer. Function keys can be used to recall these lines to screen, where they may be reentered at the command prompt

However, the function keys cannot be defined arbitrarily. At present, only 4 different definitions are possible.

Fn=CUR_UP (default setting for F9)

defines function key n for scrolling backwards in the list of lines contain in the buffer.

Fn=CUR_DN (default setting for F10)

defines function key n for scrolling forwards in the list of lines contained in the buffer.

Fn=RESHOW (default setting for F4)

defines function key n for displaying a list of all lines saved to the buffer.

Fn=CANCEL (default setting for F3)

defines function key n for supplying an end-of-file marker, which is also used to interrupt a TUSTEP session.

Analyzing files

DUMP

Command

#DUMP

Specifications

FILE = file Name of the file to analyzed.

Features

This command is used to analyze and alter the contents of files (e.g. for determining the structure of a system file). However, the contents of files should not be altered unless the exact structure of each file is known. Specifications are to be provided upon calling up this command. A list of specifications possible can be shown on screen by entering a question mark. This program is terminated by entering "*eof" in place of an instruction.

Editing data files

EDIT

Command:

#EDIT

Specifications:

FILE	= file		Name of the file whose data are to be edited
	= -STD-		The standard EDITOR file is to be edited
MODE	= T		Record numbering in text mode
	= P		Record numbering in program mode
	= -STD-		Set mode automatically (see below)
DEFINITIONS	= -	*	No new definitions; those last used remain in effect unaltered
	= file		Name of the file containing the new definitions for the Editor
	= *		New definitions for the Editor follow the command #EDIT and are ended by *EOF.
WAIT	= +		When the Editors is started, confirmation is required (by pressing the ENTER key) before the clearing the screen.
	= -	*	When the Editor is started, no confirmation is required before clearing the screen.
MACRO	= name		Name of the Editor macro that is automatically carried out after starting the Editor
	= -	*	No automatic execution of macros after starting the Editor.

Features:

This command is used to start the Editor. The Editor can be used to enter and edit text as well as programs at the display device.

If the editor is called up without giving any specification values, the file last edited and the mode last used are automatically assumed. When the editor is called up for the first time with no specifications being given, the Standard-EDITOR-file is edited in mode P.

The MODE is automatically set according to the following criteria:

- If the file has already been processed by the Editor and has not been changed since then, the mode last used remains in effect.
- If the records of the file are numbered in text mode (this is assumed if the record number of the last record is greater than 999999), MODE=T is set.
- If the first record starts with "#= ", and it's number is less than 1000000, MODE T will be used.
- In all other cases, MODE P will be used.

Restrictions:

Because each record in the editor is referred to by its record number, the record numbers in the file to be edited must be in ascending order, with each record having a unique number. Moreover, no record may exceed 600 characters.

Notes:

After the Editor has been started, Editor instructions are entered at the prompt

```
Enter instruction >
```

to tell the Editor what it is to do. After typing in an instruction, it is then sent to the computer by pressing the ENTER key. Only then will it be interpreted and executed. Editor instructions are described in detail in the chapter entitled "Editor" (starting on page 174).

The data given in the specification DEFINITIONS are used to define or preset tabulators, functions, character groups and string groups, parameters, Editor macros as well as various options (cf. chapter "Editor" pages 190, 191, 193, 202, 192, 211).

If a definition needs more than one line (i.e. more than 80 characters), it can be continued in the following line. These continuation lines have to start with a blank, which is ignored when the line is processed by other programs. Lines beginning with a "C" are treated as comment lines, and are therefore skipped over.

It is advisable to record permanently-used Editor settings in the start file (see page 21) so that with each new TUSTEP session the function keys, Editor macros, etc are automatically defined to conform to the user's particular needs.

Erasing data/deleting files

ERASE

Command:

#ERASE

Specifications:

DATA	= file		Name of the file containing the data to be erased. More than one file name is allowed.
	= -	*	No data contained in a specific file are to be erased
	= +		The data in the files selected with the specifications PROJECT/CARRIER/POSITIVE/ NEGATIVE are to be erased.
FILE	= file		Name of the file which is to be deleted. More than one file name is allowed.
	= -	*	No file with a specific name is to be deleted
	= +		Files selected with the specifications PROJECT/CARRIER/POSITIVE/NEGATIVE are to be deleted.
PROJECT	= name		Name of the project whose files are to be deleted; name of the project to be deleted. If a project is to be deleted, no entries are to be made for the specifications DATA and FILE (i.e. besides using the default setting: "-"); the carrier on which the project to be deleted is located must be given in the specification CARRIER.
	= +		Files of the current project are to be deleted.
	= -STD-		Files of the project set at initialization are to be deleted.
	= -	*	No file of any specific project is to be deleted.
CARRIER	= -STD-		The files (or project) to be deleted are (is) located on a data carrier specified by the operating system. Under DOS, UNIX

			and VMS the carrier for permanent files is set by the system variable TUSTEP_DSK.
	= name		Under DOS, UNIX and VMS: name of the system variable containing the path specification for the files (project) to be deleted. Under DOS the letter of the appropriate drive can be specified here alone if the path specification contains no directory names.
	= -	*	No file of any specific carrier (disk or disk drive) is to be deleted.
POSITIVE	=	*	No positive selection of file names by character strings
	= ...		Only those files are to be deleted whose file names contain at least one of the given character strings.
NEGATIVE	=	*	No negative selection of file names by character strings
	= ...		Only those files are to be deleted whose file names do not contain any of the given character strings.
WIPE	= +		The files to be deleted (or the data contained therein) are to be overwritten.
	= -		The files to be deleted (or the data contained therein) are not to be overwritten.

Features:

This command is used to erase the data in files as well as files themselves. If only a file's data are erased, the file the file continues to exist as an empty file. After a file is deleted, it no longer exists. In both cases, the files involved must be opened for writing.

In addition, this command is used to delete projects (directories). A project to be deleted may not contain any files. If so, they must first be deleted or renamed (with the command #RENAME 163) in such a way that they are entered under another directory. Projects specified by the system (system manager) may not be deleted.

To delete all files opened for writing in a particular project, simply enter a "+" for the specification DATA or FILE and the name of the project for the specification PROJECT. Likewise, all files on a data carrier which are opened for writing can be deleted by giving the name of the disk or disk drive for the specification CARRIER. If both specifications are used, all opened files of the given project which are located on the given data carrier will be deleted.

To erase data in all files or to delete all files whose file name contains a particular character string, enter those character strings in the specification POSITIVE which must occur at least once in the file name for the file to be erased. The specification NEGATIVE can be used to specify character strings which must not occur in the file name for the file to be deleted or the data contained to be erased. For a description of how to define the character strings, refer to the command #LIST on page 121.

The specification WIPE can be used to specify whether the data contained in the files are to be overwritten (data security!). If no entry is made for this specification, the data will be overwritten unless another mode has been set with the command #WIPE (cf. page 173).

Error stop toggle

ERROR STOP

Command:

#ERROR STOP

Specifications:

MODE	= ON	Switch on error stop
	= OFF	Switch off error stop

Features:

With this command the error stop can be switched on and off. If no entry is made for the specification MODE, the present mode of the error stop will be displayed.

If a TUSTEP program is stopped due to errors (e.g. if the file given in a command's specification does not exist, or if there are errors in the parameters), there is no point in executing the commands which follow.

If the error stop is switched off, the execution of commands will continue even after an error occurs.

If the error stop is switched on when an error occurs, the remaining commands will be canceled if in batch mode; in interactive mode the user is given a choice of three possibilities:

1) H (Hold) Before the remaining commands are carried out, priority commands shall be executed. Priority commands are subsequently entered one by one (!). If an empty input line is sent to the computer (ENTER) instead of a priority command, the interrupted command sequence will continue.

2) C (Cancel) Cancel remaining commands

3) G (Go) The remaining commands are to be executed.

When TUSTEP is initialized, the error stop is switched on (active).

Executing a sequence of commands

EXECUTE

Command:

#EXECUTE

Specifications:

COMMAND	= file		Name of the file (program or segment file) containing the command sequence to be executed.
	= -STD-		The commands to be executed are recorded in the standard EDITOR file.
	= *		The commands to be executed follow the command #EXECUTE and are ended by *EOF. In this case, only the value "-" is allowed for the specification RANGE.
RANGE	= -	*	Executes the entire command sequence given in the specification COMMAND.
	= pos		Executes only the command sequence starting at record position pos in the program file given in the specification COMMAND.
	= pos1-pos2		Executes only the command sequence located between record position pos1 to record position pos2 (inclusive) in the program file given in the COMMAND specification.
	= name		Executes only the command sequence contained in the segment name given here and whose segment file has been given in the specification COMMAND
	= name-pos		Execute only the command sequence contained in the segment name specified here starting with the record-position pos.
	= name-pos1-pos2		Execute only the command sequence contained in the segment name specified here from record position pos1 to record position pos2 (inclusively).
PARAMETER	= -	*	No parameters
	= param		Parameters to be inserted into the command sequence at the correspondingly marked positions.

LOOP	= n		Execute command sequence n times; the values 1, 2, 3, ... to n will be assigned to the loop counter, which is inserted into the command sequence at the correspondingly marked positions.
	= n-m		Execute command sequence m-n+1 times, with the values n to m being assigned to the the loop counter. If n is zero, the first pass (where the loop counter has a value of 0) will insert an empty character string instead of a 0 at the correspondingly marked positions.
	= 0-0	*	The command sequence is executed one time, with an empty string being inserted at the correspondingly marked positions for the loop counter.
	= file		The command sequence is executed for each record of the given file, with the contents of the record (instead of the loop counter) being inserted at the correspondingly marked positions.
MARKER	= -		The command sequence contained no marked positions where parameters or loop counters are to be inserted.
	= x		The character specified here has been used to mark positions in the command sequence where parameters or loop counters are to be inserted.
	= ?	*	The character "?" has been used as the marker.
LISTING	= -	*	No additional protocol of the command sequence.
	= +		The command sequence is to be written to the journal (i.e. in interactive mode, the screen) after the parameters and the loop counters have been inserted.
	= file		Name of the file to which a protocol of the command sequence is to be written.
EXECUTION	= +	*	Execute command sequence
	= -		Do not execute command sequence executed (This can be specified if, for example, the command sequence is only to be listed in a test run).
	= *		Execute command sequence in batch mode. The two sequences of operating system instructions required before and after the TUSTEP command will be entered after

	the #EXECUTE command, with each sequence ending with *EOF.
SUPPLEMENT = ...	Name of a system variable containing supplementary specifications for the operating system for starting the command sequence in batch mode (cf. system variable TUSTEP_SUB page 43). Check with the respective computing center or system administrator for which specifications are recommended or necessary in each case.

Feaatures:

With this command, a TUSTEP command sequence (normally located in a file) can be executed once or repeatedly. It is possible to modify the commands with parameters and/or a running number (= loop counter).

If only a certain part of a command sequence is to be executed, the specification RANGE can be used to select this part. A description of how to select specific parts of a command sequence is provided above, where "pos" stands for a record number whose numbering method corresponds to that used in program mode (cf. page 24). "name" stands for the name of a segment in a segment file.

For the specification PARAMETER, up to 9 parameters can be given, each separated by an apostrophe. The 1st parameter will be inserted into the command sequence at all locations marked with "?1" (where "?" is the marker); the 2nd, 3rd etc. parameters will be inserted at locations marked with "?2", "?3" etc. If fewer parameters are given for the specification PARAMETER than provided for in the command sequence, an empty string will be inserted for the missing parameters.

If the command sequence is to be repeated with other parameters, these sets of (1 to 9) parameters can be given one after the other and separated from the preceding parameter set by a semicolon. If a set of parameters contains fewer parameters than are provided for in the repeated command sequence, the same sequence of characters used in the previous set will be inserted for the missing parameters. The same is true if an empty string (i.e. a final apostrophe only) is given as a parameter. The parameters given in each parameter set thus become the default values for the subsequent set of parameters.

Commands can also be modified with a running number. It corresponds to the value of the loop counter and is inserted into the command sequence at locations marked with "?0" (where "?" is the marker). The command sequence is executed for each value in the loop counter. The values to be assigned to the loop counter can be given in the specification LOOP. If the command sequence is also to be repeated based on the values given in the specification PARAMETER, then each of these repetitions will executed with the frequency given in the specification LOOP.

Instead of using a loop counter, a character string can also be inserted at positions marked with "?0" (where "?" is the marker). These character strings must be located in the file given in the specification LOOP, where each record of this file contains one character string. Leading and trailing blanks will be ignored. The command sequence will be executed for every character string. If the command sequence is to be additionally repeated due to PARAMETER specifications, each of these repetitions will be executed with every record of the file given in the specification LOOP.

For the specification MARKER, any character except "#", "=", "'", "," and " " can be used to mark locations in the command sequence where parameters or loop counters are to be inserted. However, the specified character will effectively mark such a location in the command sequence only if it is directly followed by a digit. This ensures that the marker and the digit will be replaced in any case (if necessary, by an empty string), even if no value has been given in the specifications PARAMETER and/or LOOP. If a marker is not followed by a digit in the command sequence, the character used as a marker remains unchanged.

Note:

If, for example, a command sequence is to be executed for all files whose names contain (or do not contain) a certain character string, these file names can be written to file with the command #LIST (cf. page 121). This file can be given for the specification LOOP after making any necessary corrections in this list with the Editor.

If a command sequence is to be executed in batch mode, the related operating system instructions must be given for the specfication EXECUTION. To simplify this procedure, a standard macro called #*EXECUTE can be used, which automatically supplies the necessary operating system instructions and executes the commands in batch mode. This macro has the same specifications as the command #EXECUTE. The specification SUPLEMENT may also have to be specified as a system variable.

Restriction

Batch operations are not permitted under DOS.

Formating texts (auto. layout)

FORMAT

Command:

#FORMAT

Specifications:

SOURCE	= file		Name of the file containing the data (with formating instructions) to be formated
	= -STD-		The standard TEXT file contains the data (with formating instructions) to be formated.
DESTINATION	= -	*	Output to the LISTING only
	= file		Name of the file to which the data (with formating instructions) are to be written observing the new page-line division.
	= -STD-		The formated data (with formating instructions) are to be written to the standard TEXT file observing the new page-line division.
MODE	=		Type of printer for which the data are to be prepared. The types of printers available depends on the computer being used. To obtain a list of these, use the command #LIST,PRINTERS (cf. page 121) The printer can also be specified with parameters.
ERASE	= -	*	If the DESTINATION file or the LISTING already contains data, their data are to be retained.
	= +		If the DESTINATION file or the LISTING already contains data, their data are to be erased beforehand.
PARAMETERS	= -	*	No parameters
	= file		Name of the file containing parameters
	= *		The parameters follow the command and are ended by *EOF.
PREFIX	= -	*	No user-defined default settings

	= file		Name of the file containing formating instructions for user-defined default settings.
	= *		The formating instructions for user-defined default values follow the command and are ended by *EOF.
LISTING	= -STD-	*	The formated data are to be written into the standard LISTING.
	= file		Name of the file into which the formatted data are to be written
	= -		No output of formated data to the LISTING

Features:

With this command texts can be prepared for printing. The text is automatically laid out into lines and pages (including hyphenation, line justification and setting footnotes). The layout can be controlled by instructions which are included in the text.

Notes:

Data in the SOURCE file are never altered. The results of this program (data prepared for printing) are written to the LISTING. file. This can be subsequently printed (sent to a printer) with the command #PRINT (see page 160).

The data are written to the DESTINATION file as they are found in the SOURCE file (except for character strings which are replaced using the corresponding parameters). The line division and the page-line numbers, however, will be adjusted to the formated result.

The DESTINATION file may thus serve as a basis for further corrections and has the advantage over the SOURCE file in that the page numbering of the printed result corresponds to the page and line numbering as used in the Editor for finding the places where the text must be corrected.

If an index is to be compiled from the formatted text, it must be based on the data as formatted in the DESTINATION file to ensure that the page numbers appearing in the index reflect the current layout of the document.

Description

For this command, there is a special description with the name FO (cf. #MANUAL on page 128).

Generating forms for printing

GFORMS

Command:

#GFORMS

Specifications:

SOURCE	= file		Name of the file containing the data from which forms are to be generated.
	= -STD-		The standard TEXT file contains the data from which forms are to be generated.
MODE	= -	*	The sequence of data in the LISTING shall correspond to that of the input file.
	= -STD-		The input data are to be sorted in such a way that the forms can be sorted in stacks after being cut.
ERASE	= -	*	If the LISTING already contains data, they are to be retained.
	= +		If the LISTING already contains data, they are to be erased beforehand.
PARAMETER	= file		Name of the file containing parameters
	= *		The parameters follow the command and are ended by *EOF.
FORM	= -	*	No form to be used as a mask
	= file		Name of the file containing a form to be used as a mask
LISTING	= -STD-	*	The generated forms are to be written to the standard LISTING.
	= file		Name of the file to which the generated forms are to be written.

Features:

This command can be used to prepare data for printing in a preset format (e.g. address labels, office forms, catalogue cards). The user may specify:

- the size of the form
- standard text to appear on each form (which can also vary depending on the occurrence of specific text parts in each form).

- the line and character positions for individual text parts

- which text parts are to be repeated on continuation forms if there is not enough space for the text on a single form.

Description

For this command, there is a special description with the name GF (cf. #MANUAL on page 128).

Generating an index after sorting

GINDEX

Command:

#GINDEX

Specifications:

SOURCE	= file		Name of the file containing the index entries to be edited or containing the entries for the KWIC index.
	= -STD-		The standard TEXT file contains the index entries to be edited or the entries for the KWIC index.
DESTINATION	= -	*	Output to the LISTING only
	= file		Name of the file to which the generated index is to be written
	= -STD-		The generated index is to be written to the standard TEXT file.
MODE	= +	*	Generate index; the input data contain a reference field
	= -		Generate index; the input data do not contain a reference field
	= KWIC		Generate KWIC index
ERASE	= -	*	If the DESTINATION file or the LISTING already contains data, the data are to be retained.
	= +		If the DESTINATION file or the LISTING already contains data, the data are to be erased beforehand.
PARAMETER	= file		Name of the file containing parameters
	= *		The parameters follow the command and are ended by *EOF.
DATA	= -	*	SOURCE file contains the index entries in their entirety.
	= file		Name of the file containing that part of each index entry or of each KWIC index entry which was not required for sorting
	= -STD-		The standard DATA file contains that part of each index entry or of each KWIC index entry which was not required for sorting.

LISTING	= -STD-	*	The formated index is to be written to the standard LISTING
!	= file		Name of the file to which the formated index is to be written.
	= -		No output of the formated index

Features:

With this command, index entries and text units can be condensed and combined into an index (e.g. an index of word forms or a KWIC index, or a special list (e.g. bibliographies). The SOURCE file contains the index entries or text units which have been generated and prepared for sorting with PINDEX or PRESORT and have then been sorted with SORT. Other data can be processed also, provided each input record contains a text unit.

With this program it is possible

- to add control characters and markers
- to choose any format for the print output
- to generate titles and running heads
- to select index entries and text units
- to structure index entries hierarchically into entries and subentries
- to calculate absolute and relative frequencies

Notes:

For printing the generated index or the directory with the command #PRINT, the LISTING file must be used, *not* the DESTINATION file. The DESTINATION file is used for further processing of the index or list (e.g. for photo composition); in this case, this program cannot generate a LISTING simultaneously.

Description

For this command, there is a special description with the name GI (cf. #MANUAL on page 128).

Generating a listing of a text file

GLISTING

Command:

#GLISTING

Specifications:

SOURCE	= file	Name of the file containing the data to be listed (i.e. of which a protocol is to be generated), or (for MODE=A and MODE=U) name of the file containing the protocol from which specific parts are to be copied.
	= -STD-	The standard TEXT file contains the data from which a protocol is to be prepared, or contains the protocol from which parts are to be copied.
MODE	= T	Insert page-line number in front of each record (for data numbered in text mode).
	= -STD-	Begin a new page when the page number of the record changes; insert the line number in front of each record.
	= S	Begin a new page when the page number of the record changes; records appear with no page-line numbers.
	= O	Ignore record numbers
	= P	Insert line number in front of each record (for data numbered in program mode, and for segment files).
	= A	The SOURCE file is a listing file; page selection based on the page number in the record number.
	= U	The SOURCE file is a listing file; page selection based on the page number in the header line.
	= ...	Type of printer for which the data are to be prepared. The type of printer available depends on the computer being used. For a list of these, use the command #LIST,PRINTERS (siehe Seite 121). The printer type can also be specifed in parameters.

ERASE	= -	*	If the LISTING already contains data, they are to be retained.
	= +		If the LISTING already contains data, they are to be erased beforehand.
PPARAMETER	= file		Name of the file containing parameters
	= *		The parameters follow the command and are ended by *EOF.
LISTING	= -STD-	*	The generated listing is to be written into the standard LISTING.
	= +		Listing should be outputed to the journal (i.e. written to screen)
	= file		Name of the file into which the generated listing is to be written

Features:

With this command a listing of a file can be generated, i.e. a list file of its data can be prepared for subsequent printing. Control characters contained in the data are not interpreted but treated as printable characters. To determine the format of the listing, parameters can be used to specify the number of columns, headers for pages and columns, type of numbering and line spacing.

Furthermore, specific pages of a listing file given in the specification SOURCE can be selected and copied to the file given in the specification LISTING.

Note

Data in the SOURCE file always remain in their original form. The results of the program (the data listing prepared for printing) are written to the LISTING file. This file can then be sent to a printer with the command #PRINT (see page 160).

Description:

For this command, there is a special description with the name GL (cf. #MANUAL on page 128).

Online help

HELP

Command

#HELP

Specifications none

Features

This program can be used interactively for displaying TUSTEP help texts on screen. Texts are selected by choosing the desired topic from a hierarchical table of contents. After the online help command is given, a menu will appear with the main help topics. If one of these topics is selected, a new window will be opened which displays either a submenu and its topics or the help text of the selected topic. Control codes (described below) are used to select topics and to scroll through the help text.

Control codes

As a quick reference aid, each control code has a short form by which it is listed in the tables on pages 230 to 239. These tables show which key or key combination must be pressed to carry out each control code for the most common computer keyboards.

CUR_DN Cursor down

Moves the cursor down one line. If the cursor is already in the last line of the active window and if pertinent data are located after this line, the contents of the window will be moved up one line in order to display the following line.

CUR_UP Cursor up

Moves the cursor up one line. If the cursor is already in the first line of the active window and if pertinent data are located before this line, the contents of the window will be moved down one line in order to display the preceding line.

CUR_RI Cursor right: "Select"

In a menu, this key displays the submenu or related text located in the same line as the cursor.

CUR_LE Cursor left: "Return"

In the main menu, this key exits the online help program; in submenus and in help texts this key will display the next higher menu.

ENTER Enter: "Select"

Same as CUR_RI

DEL Delete: "Return"

Same as CUR_LE

SHW_DN "Show next screen of data"

Scrolls forward in the current menu or help text (i.e. toward the end of the text).

SHW_UP "Show preceding screen of data"

Scrolls backwards in the current menu or help text (i.e. toward the start of text).

SKP_BEG "Skip to start of data"

Jumps to the beginning of the current menu or help text.

SKP_END "Skip to end of data"

Jumps to the end of the current menu or help text.

REFRESH "Refresh screen"

Restores screen display disrupted by system messages, line interference, etc.

CANCEL "Terminate"

Exits the online help program.

Note

Online help can also be called up from the Editor with the instruction HELP or the control command HELP.

Information about macro files, variables, etc.

INFORM

Command:

#INFORM

Specifications:

MACROS	= $name		The description of the user macro having the name specified here and contained in the current macro file is to be listed in the journal (i.e. in interactive mode, the screen). More than one name is allowed.
	= *name		The description of the standard macro having the name specified here is to be listed in the journal (i.e. in interactive mode, the screen).
	= name		The description of the user macro having the name specified here, or - if there is no user macro with this name - the description of the standard macro - is to be listed in the journal (i.e. in interactive mode, the screen). More than one name is allowed.
	= +		The name of the current macro file and a list of the macros contained therein is to be listed in the journal (i.e. in interactive mode, the screen).
	= -STD-		The names of the available standard macros are to be listed in the journal (i.e. in interactive mode, the screen).
	= -	*	No listing of information about the current macro file, user-defined macros or standard macros
VARIABLES	= name		The definition of the variable having the name specified here is to be listed in the journal (i.e. in interactive mode, the screen). More than one name is allowed.
	= +		The definitions of all defined TUSTEP variables are to be listed in the journal (i.e. in interactive mode, the screen).
	= -	*	No listing of information concerning variables

PROJECT	= +		The current project name is to be listed in the journal (i.e. in interactive mode, the screen).
	= -	*	No listing of current project name
USER	= +		The current user name is to be listed to the journal (i.e. in interactive mode, the screen).
	= -	*	No listing of the current user name
CODE	= +		The current code table for the display device is to be listed in the journal (i.e. in interactive mode, the screen).
	= -	*	No listing of current code table
FUNCTIONS	= name		The definitions of all defined function keys are to be listed in the journal (in interactive mode, the screen)
	= -	*	No listing of information concerning function keys

Features:

With this command the following information can be listed:

- descriptions of user macros from the current macro file
- descriptions of standard macros
- name of the current macro file and the names of the macros it contains
- names of standard macros
- contents of specified TUSTEP variables
- names of all defined TUSTEP variables and their values
- current project name used to complete the file names. This project name is used if in a TUSTEP command a file name does not explicitly contain a project name.
- current user name to be used on the cover sheet of print output
- current code table being used for input and output at the display device
- defining function keys at the command level.

If this command is given without any specifications, TUSTEP user information will be listed if available.

Notes:

The current macro file is the file most recently defined as such with the command #DEFINE (cf. page 83).

The description of a macro consists of the comment lines ("$$-" in column 1 to 3), located immediately at the beginning of a macro in the macro file.

TUSTEP variables can be defined with the command #DEFINE (see page 83) or in macros (see macro instruction DEFINE on page 272); their values can be used in commands and in macros. This command provides no information about variables defined within macros.

The project name and the user name can be redefined with the command #DEFINE (cf. page 83). This command can also be used to set the code table and define function keys.

Inserting text parts

INSERT

Command:

#INSERT

Specifications:

SOURCE	= file		Name of the file containing the data where text parts are to be inserted
	= -STD-		The data where text parts are to be inserted are located in the standard TEXT file.
DESTINATION	= file		Name of the file to which the data with the inserted text parts are to be written
	= -STD-		The data with the inserted text parts are to be written to the standard TEXT file.
MODE	= -STD-	*	Unique short form, normal case
	= SHORT		Unique short form, short text part to be inserted
	= LONG		Unique short form, long text part to be inserted
	= PARALLEL		Short forms are located in the SHORT file and in the SOURCE file in parallel fashion.
	= SORTED		Short forms are sorted alphabetically.
	= MORE		The SHORT file contains more short forms than the SOURCE file.
	= LESS		The SHORT file contains less short forms than the SOURCE file.
	= GROUPS		The SHORT file contains groups, each having the same short forms. The SOURCE file is to be copied for each group.
ERASE	= -	*	If the DESTINATION file already contains data, they are to be retained.
	= +		If the DESTINATION file already contains data, they are to be erased beforehand.
PARAMETERS	= file		Name of the file containing parameters
	= *		The parameters follow the command and are ended by *EOF.

SHORT	= file	Name of the file containing the text parts to be inserted

Features:

With this program, text parts which are located in a file and are indentifiable by a short form (e.g. a running number) can be inserted (merged) into the data of another file. Each text part is inserted at the position in the data where its corresponding short form (abbreviation for the text part) is located.

Practical examples (the type of data located in the SHORT file is shown in parentheses):

- Merge footnotes with the text (footnotes)

- Replace short forms with the full text (full text)

- Compile form letters (form mask)

- Create series letters (one address per group)

Description

For this command there is a special description with the name IS (cf. #MANUAL page 128).

Activating/deactivating journal file

JOURNAL

Command:

#JOURNAL

Specifications:

MODE =	ON	Activate journal file
	OFF	Deactivate journal file
	NEW_PAGE	Start a new page in journal file, if activated.
	SHOW	Deactivate journal file and write it to the journal (i.e. in interactive mode, the screen).
	EDIT	Deactivate journal file and start Editor for editing journal file.
	COPY	Deactivate journal file and copy it to the file given in the specification FILE.
	ERASE	Erase journal file.
	PRINT	Deactivate journal file and send it to the printer given in the specification DEVICE.
	-	Suppress output of journal file
	+	Do not suppress output of journal file
	CONTINUOUS	Uninterrupted journal output to screen
	PORTIONED	Portioned journal output to screen, next screen portion not displayed until ENTER key is pressed.
	SYSTEM	Journal output to screen in the conventional manner used by the respective operating system.
TYPE =	...	Type of printer to be used. The type of printer available depends on the actual computer being used. To obtain a list of these, use the command #LIST, PRINTERS (see page 121).
DEVICE =	...	If MODE=PRINT, name of the printer used. The names of the available printers may be obtained from the information bulletin of the respective computing center.

COPIES	= 1	*	If MODE=PRINT, the journal file is to be printed once.
	= n		If MODE=PRINT, the journal file is to be printed n times.
PREFIX	= -	*	No prefix
	= *		The prefix follows the command #JOURNAL and is ended by *EOF.
	= file		Name of the file containing the prefix
PAGES	= -	*	The journal file is to be printed in its entirety.
	= n / n-m		Print only page n / pages n-m. More than one page or range of pages may be specified here.
COVER	= -STD-	*	Whether TUSTEP supplies a cover for the printed output (cover and end sheets with names) depends on the computer and printer being used.
	= -		No cover to be supplied by TUSTEP. This specification will be ignored if the computing center does not allow this for certain printers.
	= +		TUSTEP is to supply a cover for the printed output.
FILE	= -	*	Normal case
	= file		If MODE=COPY, name of the TUSTEP file to which the journal file is to be copied. If MODE=PRINT, name of the system file to which the printer control codes are to be written. More than one file name may be specified.
ERASE	= -	*	If the file given in the specification FILE already contains data, they are to be retained.
	= +		If the file given in the specification FILE already contains data, they are to be erased beforehand.
OPTIONS	= ...	*	Specifications for modifying the generated printer control codes.
SUPPLEMENT	= ...		Name of the system file that contains supplementary specifications for the operating system (cf. system variable TUSTEP_LPR page 42). The appropriate specifications may be obtained from the

		information bulletin or system manager of the respective comptuting center.
PORTION	= -STD-	The printout should be subdivided into portions of 500 pages each (for line printers), or portions of 200 pages each (for dot matrix and laser printers).
	= n	The printout should be divided into portions of n pages each.

Features:

This command can be used to log TUSTEP messages and print output (in addition to the journal (i.e. in interactive mode, the screen) into a journal file.

When the journal file is active, the output generated by giving the value "+" for the specification LISTING of TUSTEP commands (LISTING =+) as well as the list of parameters will be written into the journal file only. The start and end messages as well as any error messages will be written into both the journal and the journal file.

When TUSTEP is initialized, the journal file is deactivated.

If the command is called up with no specifications, only the current mode of the journal file will be displayed (i.e. whether the journal file is active or not).

With this command, the journal file can be
- displayed on screen,
- processed with the Editor,
- copied to a file,
- erased,
- printed.

If only certain "pages" are to be printed, they can be selected with the specification PAGES. Pages are selected based on the record numbers in the journal file irrespective of the actual printed page number. Page selection should generally not be done until the journal file has been viewed in the Editor, where the file's page organization can be ascertained.

If the journal file is to be printed out on a remote printer, (e.g. one connected to another PC), a permanent system file (TYP=SDF) can be entered for the specification FILE. The printer control codes for the specified printer will then be written to this file. The appropriate operating system command can then be used to send this system file to the desired printer (e.g. the DOS PRINT command). If more than one file has been given in the specification FILE, only the number of pages given in the specification PORTION will be written to each file. Output will then proceed to the next file.

For printing on EPSON compatible matrix printers, the following options can be specified:

BD bidirectional printing
UD unidirectional printing (default)

H11 page length 11"
H12 page length 12"

LQ letter quality printing (default)
DRAFT draft quality printing

When executing "debugged" command sequences, is may be advisable to reduce journal recording to a minimum. By setting the mode to "-", the journal will record only comments beginning with "#+", error messages and any messages that are part of an input prompt. This mode can be cancelled by setting the mode back to "+". If this mode has not been cancelled at the end of a command sequence in interactive mode, it will be automatically cancelled so that the journal is able to record newly entered commands in normal fashion.

In interactive mode, this command can also be used to set the type of screen display and keyboard input:

MODE = SYSTEM

Output and input conform to the usual method common to the operating system being used. The function keys and control commands are not (or only partially) available for input, correction and repetition of commands at the command line. This is the mode set after TUSTEP is initialized.

MODE = CONTINUOUS

Output proceeds uninterrupted until the next input prompt. Function keys and control commands may be used to input, correct and repeat commands at the command line. (cf. page 57).

MODE = PORTIONED

Corresponds to MODE = CONTINUOUS, except that output pauses, when the screen is filled (usually after 22 lines of output). Upon confirmation, the next screen portion of the journal will then be outputted. The size of the next portion can be specified by entering the number of desired lines.

After initialization of TUSTEP, output and input operations proceed in the manner common to the operating system being used (i.e. as in MODE=SYSTEM).

Note:

If the journal file is printed with MODE=PRINT, it will not be erased at the same time. This must be done with the specification MODE=ERASE.

Restrictions in DEVICE specifications

Under DOS, only the "+" specification for DEVICE has any effect; all other specifications do not. Output is directed to the printer previously set with the DOS PRINT command (i.e. at the operating system level) assuming no file has been given in the FILE specification.

Listing file names etc.

LIST

Command:

#LIST

Specifications:

MODE = -STD-		*	If no file name has been given for the specification FILE, all files currently opened will be listed. Otherwise, the beginning and end of the specified file will be listed.
	= n		Lists the first n records of the file given in the specification FILE.
	= n;m		Lists the first n and the last m records of the file given in the specification FILE.
	= PRINTERS		Lists all printers supported by the TUSTEP version being used.
	= FILENAMES		Lists the names of all existing files (whether opened or not) of a project. In case a file name has been given in the specification FILE, the list of file names will be written to this file.
	= PROJECTNAMES		Lists the names of all existing projects of a data carrier. In case a file has been given in the specification FILE, the list of project names will be written to this file.
FILE	= -	*	Normal case
	= name		If MODE=-STD-, "n" or "n;m": name of the file whose beginning and end are to be listed. If MODE=FILENAMES or PRINTERS: name of the file to which the file names are to be written.
ERASE	= -	*	If the file given for the specification FILE already contains data, they are to be retained.
	= +		If the file given for the specification FILE already contains data (and when MODE=FILENAMES or PRINTERS), they are to be erased beforehand.
PROJECT	= name		If MODE=FILENAMES: lists the names of all files in this project.

	= +	*	If MODE=FILENAMES, the names of all files in the current project are to be listed.
	= -STD-		If MODE=FILENAMES: lists the names of all files in the project set at initialization.
CARRIER	= -STD-	*	The files whose names are to be listed are located on a data carrier (disk or disk drive) set by the operating system. Under DOS, UNIX and VMS: the carrier for permanent files is set by the system variable TUSTEP_DSK.
	= name		Under DOS, UNIX and VMS: name of the system variable containing the path specification for the files or project whose names are to be listed. Under DOS, the drive letter alone can be specified here if the path specification contains no directory names.
POSITIVE	=	*	Normal case
	= ...		If MODE=FILENAMES or MODE=PROJECTNAMES, only those file names are to be listed which contain at least one of the specified character strings.
NEGATIVE	=	*	Normal case
	= ...		If MODES=FILENAMES or MODE=PROJECTNAMES, file names or project names containing one of the specified character strings are not to be listed.
VERSION	= -STD-	*	Normal case
	= +n		Under VMS: If MODE=FILENAMES, only file names having the n largest version number are to be listed.
	= -n		Under VMS: If MODE=FILENAMES, file names having the n largest version number are not to be listed.
REPLACE	= -	*	Normal case
	= ...		If MODE=FILENAMES or MODE=PROJECTNAMES the specified replacements should be made in the list of file names or project names before it is written to the file given in the FILE specification.

Features:

With this command, the following information can be listed. If not specified otherwise, it will be listed into the journal (i.e. in interactive mode, the screen). Only those

specifications relevant to the desired action will be described in the following.

Names of all opened files

#LIST

Opened files also include files that have been created but not yet closed or erased. In addition to the file names, the following information will also be listed for TUSTEP files:

- file type
- whether the file has been opened for reading READ (R) or writing WRITE (W); a scratch file can be recognized as such in that neither (R) nor (W) appears.
- average record length
- number of records in the file
- file size in KB
- Percent of file occupied by data
- Name of the data carrier (for system variables)

Names of all existing files

#LIST, FILENAMES, PROJECT=..., CARRIER=...,
POSITIVE=..., NEGATIVE=...

This lists all files which have been created under the specified project on the specified carrier and which have a TUSTEP-compatible name.

To produce a list of only those file names which contain (or do not contain) certain character strings, these character strings can be entered in the specification POSITIVE (or NEGATIVE): for a file name to be listed, at least one of the character strings given in the specification POSITIVE must occur in its name. Conversely, character strings given in the specification NEGATIVE can be used to exclude file names from the list which contain one of the specified character strings. For a description of how to define character strings when selecting file names, see below.

Names of all available projects

#LIST, PROJECTNAMES, CARRIER=...,
POSITIVE=..., NEGATIVE=...

This produces a list of all projects located on the specified carrier which have a TUSTEP compatible name.

To produce a list of only those projects which contain (or do not contain) certain character strings, these character strings can be entered for the specification POSITIVE (or NEGATIVE); for a project name to be listed, at least one of the character strings given in the specification POSITIVE must occur in its name. Conversely, character strings given in the specification NEGATIVE can be used to exclude project names from the list which contain one of the specified

character strings. For a description of how to define character strings when selecting project names, see below.

Beginning and end of a TUSTEP file

#LIST, FILE=..., CARRIER=...

The file given for the specification FILE must be a TUSTEP file. The first 5 records and the last 3 records of this file will be listed.

If the file with the specified name on the specified carrier has not been opened, it will be automatically opened and then immediately closed.

Supported printers

#LIST, PRINTERS

This will list all printers supported by the TUSTEP version being used. The types of printers supported depends on the version and computer being used.

The printer used for printing must be specified in the parameter GER when data are prepared for printing and also in the specification TYPE when printing with the command #PRINT.

Notes:

If, for example, a command sequence is to be executed for all files whose names contain (or do not contain) a certain character string, the command #LIST can be used to select and then write these file names to a file. After making any necessary corrections to the list with the Editor, this file can be given for the specification LOOP in the command #EXECUTE (see page 97).

The command #INFORM (cf. page 111) can also be used to list additional information concerning macro files, variables, etc.

Character selection strings for file names

The specifications POSITIVE and NEGATIVE can be given character strings which must either occur or not occur in file names, and, if necessary, character strings (character exception strings), which are to be ignored when this check is carried out. If character strings of different lengths have been given in a specification, the one having the longer length is given precedence. For character strings of equal length, the order of priority corresponds to the order they were given in the specification. Character strings must be separated from each other by a delimiter character of the user's choice. Due to syntactical reasons, however, the characters "#", "=", "'" and "," may not be used as delimiters. The delimiter is the first character given in the specification. A double delimiter is used to separate character strings to be searched from those which

are to be ignored during the search. The last character given in the specification must again be the delimiter.

In a character string, one of the character combinations listed below may be given instead of a single character. They are used to represent a group of characters.

>/	all digits in the 7-bit and 8-bit TUSTEP character sets
</	all letters in the 7-bit and 8-bit TUSTEP character sets
>%	all characters in the 7-bit TUSTEP character set
<%	all characters of the 7-bit and 8-bit TUSTEP character set

A frequency condition given in a character string applies to the character which directly follows it:

><n	character must occur at least n times
<>n	character may occur up to n times
><0	character may be absent
<>0	character may occur any number of times

Here any single-digit number from 1 to 9 may be given for "n". For "><0" and "<>0", the zero may be omitted if no digit follows. Thus, "<>0>/" has the same effect as "<>>/" and stands for "any number of digits". The frequency conditions "><n" and "><0" may be combined with "<>n" and "<>0". For example, "x><<>yz" means that between x and z the y may be absent, or that any number of y's may be located between x and z.

If the given character string is to match the beginning of a file name, "><x" must be added to the beginning of the character string, where "x" stands for the selected delimiter. Likewise, "<>x" must be added to the end of a character string if the given character string is to match the end of the file name. Thus, "/></TXT/" stands for all file names which start with TXT, and "/TXT<>//" for all file names which end with TXT (here / is the delimiter; the character " is not included in the specification).

In addition, pointers and surrounding conditions may also be specified. For a more detailed description of various character strings definitions used in the specifications POSITIVE and NEGATIVE please refer to the chapter entitled "Parameters" (page 241 ff.); These specifications correspond to the character search string table described for parameter type IX.

Character replacement strings for file names

The specification REPLACE is used to specify a pair of character strings, where the the first character string is the search string, which is to be replaced by the pair's second character string, the replacement string. In addition, the user can also specify character search strings which are not to be replaced (exception strings).

Character search strings are treated in the same manner as described above for "Character selection strings for file names".

The character search and replacement strings may be of any length. However, they must be separated from each other by a delimiter character, which can be any character chosen by the user. Due to syntactical reasons, however, the characters "#", "=", "'" and "," may not be used as delimiters. The delimiter is always the first (and last) character given in the specification. To switch from character string pairs to exception strings, the delimiter is written where a search string would normally be expected (i.e. a double delimiter). Please note once more that the last character of the specification must be the delimiter character.

Character replacement strings can also contain pointers to the related search string. For a more detailed description of the various character strings definitions used in the specification REPLACE, please refer to the chapter entitled "Parameters" (page 241 ff.). These specifications correspond to the character string replacement table described under parameter type X.

Creating command sequences

MACRO

Command:

#MAKRO

Specifications:

SOURCE	= file		Name of the file containing macro instructions and commands
	= *	*	The macro instructions and commands follow the command #MACRO and are ended by *EOF.
LISTING	= -	*	No additional listing of the generated command sequence
	= +		The generated command sequence is to be written to the journal (i.e. in interactive mode, the screen).
	= file		Name of the file to which the generated command sequence is to be written.
EXECUTION	= +	*	The generated command sequence is to be executed.
	= -		The created command sequence is not to be executed.

Features:

With this command, a command sequence can be created and executed in the form of a macro. In this case, however, specification values cannot be further conveyed; the macro instruction $$! is therefore not allowed. This command also permits user interaction during the execution of a command sequence without having to execute a command macro. Examples pertaining to the use of this command are given at the end of the chapter entitled "Macros" (page 290).

Printing descriptions

MANUAL

Command:

#MANUAL

Specifications:

NAME	= name		Name of the description to be listed. More than one name is allowed.
TYPE	=		Printer to be used. The type of available printers depends on the actual computer being used. To obtain a list of these, use the command #LIST, PRINTERS (cf. page 121).
DEVICE	= ...		Name of the printer for printing descriptions. A list of available printers may be obtained from the information bulletin or system manager of the respective computing center.
	= +		Description to be sent to the journal (i.e. in interactive mode, the screen).
COPIES	= 1	*	1 copy of the description is to be printed.
	= n		n copies of the description are to be printed.
PREFIX	= -	*	No prefix
	= *		The prefix follows the command #PRINT and is ended by *EOF.
	= file		Name of the file containing the prefix.
PAGES	= -	*	The entire description is to be printed.
	= n / n-m		Print only page n / pages n to m. More than one page or range specification possible.
COVER	= -STD-	*	Whether TUSTEP supplies a cover for printed output (cover and end sheets) depends on the computer and printer being used.
	= -		No cover to be supplied by TUSTEP. This specification will be ignored if not applicable for certain printers as defined by the computing center or system manager.

	= +		TUSTEP is to supply a cover for the printed output.
DATEI	= -	*	Output to the printer specified in the specification DEVICE.
	= file		Name of the system file to which the printer control codes are to be written. More than one file name may be specified.
ERASE	= -	*	If the file given for the specification FILE already contains data, they are to be retained.
	= +		If the file given in the specification FILE already contains data, they are to be erased beforehand.
OPTIONS	= ...	*	Specifications for modifying the generated printer control codes.
SUPPLEMENT	= ...		Name of a system variable containing supplementary specifications for the the operating system (cf. system variable TUSTEP_LPR page 42). The appropriate specifications should be obtained from the respective computing center or system manager.
PORTION	= -STD-		The printout should be subdivided into portions of 500 pages each (for line printers), or portions of 200 pages each (for dot matrix and laser printers).
	= n		The printout should be divided into portions of n pages each.

Features:

This command is used to print TUSTEP descriptions.

Descriptions currently available and the latest version of each description have been compiled in a list, which can be printed or viewed by entering "AB" (German for "Aktuelle Beschreibungen" (current descriptions) in the specification NAME.

If only certain pages are to be printed, they can be selected with the specification PAGES.

If a description is to be printed out on a remote printer, (e.g. one connected to another PC), a permanent system file (TYP=SDF) can be entered for the specification FILE. The printer control codes for the specified printer will then be written to this file. The appropriate operating system command can then be used to send this system file to the desired printer (e.g. the DOS PRINT command). If more than one file has been given in the specification FILE, only the number of pages given in the specification PORTION will be written to each file. Output will then proceed to the next file.

For printing on EPSON compatible matrix printers, the following options can be specified:

BD	bidirectional printing
UD	unidirectional printing (default)
H11	page length 11"
H12	page length 12"
LQ	letter quality printing (default)
DRAFT	draft quality printing

If the command is called with no specifications, the on-line help will be automatically activated (see #HELP command on page 109).

Restrictions

Under DOS, only the "+" specification for DEVICE has any effect; all other specifications do not. Output is directed to the printer previously set with the DOS PRINT command (i.e. at the operating system level) assuming no file has been given in the FILE specification.

Note:

Giving the value ZV for the specification NAME results in list of the TUSTEP character set. This can be used to determine which TUSTEP characters can be printed on a given printer.

Merging data/files

MERGE

Command:

#MERGE

Specifications:

SOURCE	= file		Name of the files containing the data to be merged.
DESTINATION	= file		Name of the file to which the merged data are to be written.
	= -STD-		The merged data are to be written into the standard TEXT file.
SORTFIELD	= sf-A		The input data have been sorted according to the sort field sf, in ascending order.
	= sf-D		The input data have been sorted according to the sort field sf, in descending order.
	= sf		as in sf-A
			One of the following can be specified for sf:
			0 Sorting according to record numbers
			n-m The sort field starts with character position n and ends with character position m.
			n+m The sort field starts with character position n and is m characters long.
			More than one sort field is allowed.
ERASE	= -	*	If the DESTINATION file already contains data, they are to be retained.
	= +		If the DESTINATION file already contains data, they are to be erased beforehand.
DELETE	= -	*	The records are to be written into the DESTINATION file unaltered.
	= n-m		Before the data are written into the DESTINATION file, the characters from position n to position m are to be eliminated from each record.

	= n+m		Before the data are written into the DESTINATION file, m characters starting with position n are to be eliminated in each record.
	= +		Only the data in the DATA file are to be written to the DESTINATION file in the sorted sequence.
DATA	= -	*	No DATA file (normal case).
	= file		On merging, the data are read from the given DATA file and written in the same order to the DESTINATION file.
	= -STD-		On merging, the data are read from the standard DATA file and written in the same order to the DESTINATION file.

Features:

With this command, files containing sorted data can be merged.

The specifications given for SORTFIELD must be identical to those given for this specification in the command #SORT when sorting the files to be merged.

This program does not check whether the data have actually been sorted according to the specified sort field. The sort field is only used to determine which record read from the various files is to be the next record written to the DESTINATION file.

If the sort fields of records contained in different SOURCE files are identical, the order in which the records are written to the DESTINATION file is the same as the sequence in which the corresponding files are given for the specification SOURCE. The specification DELETE can be used to delete sort fields which are no longer needed after sorting.

Copying from magnetic tape to magnetic tape

MTCOPY

Command:

#MTCOPY

Specifications:

STAPE	=	Reel (ID) number of the magnetic tape from which files are to be copied (source tape). The reel number may consist of letters and digits only.

SNUMBER	= -STD-	*	The last file (i.e. the most recent version) having the corresponding name is to be copied.
	= n		The first file having the corresponding name is to be copied, with the search starting at the nth file on the source tape.
SLABEL	= -STD-	*	All files starting from file position n given in the specification SNUMBER are to be copied. If the value for the specification SNUMBER is -STD- (default value), only the last (i.e. the most recent) file of all files with the same file name will be copied.
	= name		Name of the file to be copied. More than one file name is allowed.
SCODE	= ASCII		The data on the source tape have been written in ASCII code (with ANSI-label).
	= EBCDIC		The data on the source tape have been written in EBCDIC code (with IBM-label).
SDEVICE	= -STD-	*	A tape unit specified by the system is to be used for the source tape. Under UNIX and VMS the tape unit is set by the system variable TUSTEP_MT1.
	= name		Under UNIX and VMS: Name of the system variable specifying the name of the tape unit for the source tape.
DTAPE	=		Reel (ID) number of the magnetic tape onto which files are to be copied (destination tape). The reel number may consist of letters and digits only.
DNUMBER	= -STD-	*	The file(s) shall be written after the last file on the destination tape.
	= n		The file(s) shall be written onto the tape starting at the nth file position.
DLABEL	= -STD-	*	The name(s) of the file(s) on the destination tape shall be the same as the name(s) on the source tape.
	= name		The file(s) on the destination tape shall have the name(s) given here. The number of names must agree with the number of files given for the specification SLABEL. If the value -STD- is given for the specification SLABEL, the value -STD- must also be given for the specification DLABEL.

DCODE	= ASCII		The tape is to be written in ASCII code (with ANSI-label).
	= EBCDIC		The destination tape is to be written in EBCDIC code (with IBM-label).
DDEVICE	= -STD-	*	A tape unit specified by the sytem is to be used for the destination tape. Under UNIX and VMS the tape unit is set by the system variable TUSTEP_MT2.
	= name		Under UNIX and VMS: name of the system variable containing the name of the tape unit.
DENSITY	= 1600		Under BS2000 and MVS: The tape is to be written with 1600 bpi.
	= 6250	*	The tape is to be written with 6250 bpi.
LISTING	= +	*	The list of all files on the destination tape are to written in the journal.
	= -STD-		The list of all files on the destination tape are to be written to the standard LISTING file.
	= file		Name of the file to which the list of of all files on the destination tape are to be written.
ERASE	= -	*	If the LISTING file already contains data, they are to be retained.
	= +		If the LISTING file already contains data, they are to be erased beforehand.

Default values:

The default value for each specification is marked with an asterisk (*). The default values for the specifications SCODE and DCODE depend on the operating system:

ASCII: UNIX, VMS
EBCDIC: BS2000, MVS, VM/CMS

Features:

With this command, files can be copied directly from one magnetic tape to another. The files to be copied can be specified by the user.

After the files are copied, a list of the files located on the destination tape can be created. This list corresponds to that generated with the command #INFORM (see page 111).

Warning:

If the files are not copied after the last file on the tape, the file which is thus overwritten and all following files already on the tape will be lost.

If the specifications SNUMBER=-STD- (default value) and SLABEL=-STD- (default value) are given, not all files will be copied if the source tape contains files with the same name. In this case, of all files with identical names, only the *last* one (i.e. the most recent version) will be copied. If all files are to be copied, the specification SNUMBER=1 must be given.

Note:

If the tape being written to is used for the first time or is written to at its beginning, the specification DNUMBER must be 1 (DNUMBER=1).

Restrictions:

All files recorded on the source tape must have been written with the command #MTWRITE or #MTCOPY.

Files must be written to the destination tape either from the beginning of the tape, or all files already present on the destination tape must have been written with the commands #MTWRITE or #MTCOPY.

The destination tape must be labeled. Unlabeled tapes will be rejected for reasons of data security. They can be labeled with the command #MTLABEL (see page 139).

Because standard labels are used, all files on a tape must be written onto the tape in the same code.

When giving a value for the specifications SCODE and DCODE, the following points should be considered:

- BS2000, UNIX, VM/CMS and VMS versions: If the wrong code is given for the source tape, it will be automatically adjusted to the code in which the tape has been written. When the first file is written onto the destination tape, the code can be freely chosen; if necessary, the tape will be relabeled. If a file is not to be written to the beginning of the tape and the wrong code is given, the code is automatically adjusted to the code used on the tape up to this point.

- MVS version: only the code used by the computing center to label the source tape and the destination tape is allowed. If a different code is given, the operator will not be able to assign the tape properly.

Under the operating system MVS, some computing centers can only use magnetic tapes in batch mode. In this case, the error message "No magnetic tape permission" will appear. Batch jobs can be started with the command #EXECUTE (cf. page 97). At some computing centers, a standard macro called #*MTC is available

which will start such a batch job. It has the same specifications as the command #MTCOPY. Additional information relating to this can be obtained with the command #INFORM,*MTC.

Warning:

When using ASCII code for writing or reading magnetic tapes under the operating system MVS, most computing centers use 7-bit mode (instead of 8-bit mode). In this case, characters encoded with the escape character "^" (e.g. the German umlauts ä, ö and ü) will be lost.

If a file on the tape contains such characters and if only a 7-bit data transfer is possible, the tape cannot be read without losing these characters. The loss of these characters can be prevented by using a tape to which the files are written in EBCDIC code or which contains only 7-bit ASCII-characters.

The latter can be achieved by using the command #CONVERT,...,MODE=-1CODE=- before the data are written onto magnetic tape. This command converts the data into 7-bit ASCII characters. After these data have been read from the magnetic tape, they can be converted back into TUSTEP code with the command #CONVERT,...,MODE=+1,CODE=-.

Information about a magnetic tape

MTINFORM

Command:

#MTINFORM

Specifications:

TAPE	=		Reel (tape ID) number of the magnetic tape for which file information is requested. The reel number may consist of letters and digits only.
LISTING	= +	*	The list of files is to be written into the journal.
	= -STD-		The list of files is to be written to the standard LISTING file.
	= file		Name of the file to which the list of files is to be written.
ERASE	= -	*	If the file given for the specification LISTING file already contains data, they are to be retained.
	= +		If the file given for the specification LISTING file already contains data, they are to be erased beforehand.
CODE	= ASCII		The data have been written to the tape in ASCII-code (with ANSI label).
	= EBCDIC		The data have been written to the tape in EBCDIC code (with IBM label).
DEVICE	= -STD-	*	The system tape unit is to be used; under UNIX and VMS the tape unit is set with the system variable TUSTEP_MT1.
	= name		Under UNIX and VMS: name of the system variable containing the name of the tape unit.

Default values:

The default values for the specifications are marked with an asterisk (*). The default value for the specification CODE depends on the operating system.

ASCII: UNIX, VMS
EBCDIC: BS2000, MVS, VM/CMS

Features:

This command provides information about files written to magnetic tape. The names of the files, the date they were written onto the tape with #MTWRITE and the number of records they contain will be listed.

Notes:

If files are copied to magnetic tape with the commands #MTWRITE or #MTCOPY, these commands can also be used at the same time to create a list of files currently on the magnetic tape. The format of these file lists correspond to that of lists created with the command #INFORMIERE.

If such a file list is compiled from different magnet tapes and is to be written to the same file (without erasing any data contained therein), the standard macro #*MTISORT can be used to sort this file. Here only the most recent file list for each tape will be used to list files; the file list is then sorted according to tape number. An additional listing of files can also be created where files are sorted by file name. This lets the user see which files are located on which magnetic tape as well as the date they were copied from disk to magnetic tape. Further information can be obtained with the command #INFORM,*MTISORT.

Restrictions:

All files recorded on the magnetic tape must have been written with the commands #MTWRITE or #MTCOPY.

When giving a value for the specification CODE, the following points should be considered:

- BS2000, UNIX, VM/CMS and VMS versions: If the wrong code is given, it will be automatically adjusted to the code in which the files were written onto the tape.

- MVS version: Only the code in which the magnetic tape was labeled by the computing center is allowed. If another code is given for the specification CODE, the operator cannot assign the tape properly.

MVS operating system: At some computing centers magnetic tapes can only be used in batch jobs. In this case, the error message "No magnetic tape permission" will appear. Batch jobs can be started with the command #EXECUTE (cf. page 97). Some computing centers (e.g. in Tübingen) offer a standard macro called #*MTI which will start such a batch job. It has the same specifications as the command #MTINFORM. Further information relating to this macro can be obtained with the command #INFORM,*MTI .

Adding/removing label of a magnetic tape

MTLABEL

Command

#MTLABEL

Specifications

TAPE	=		Reel (ID) number of the tape to be labelled, or whose label is to be removed. This ID may only consist of alphanumeric characters.
DENSITY	= 1600		Under BS2000 and MVS: the tape is to be labeled with 1600 bpi.
	= 6250	*	Under BS2000 and MVS: the tape is to be labeled with 6250 bpi.
CODE	= ASCII		The tape is to be labeled/is already labeled in ASCII-Code (with an ANSI label).
	= EBCDIC		The tape is to be labeled/is already labeled in EBCDIC-Code (with an IBM-Label).
DEVICE	= -STD-	*	A tape unit set by the operating system is to be used; under UNIX and VMS the tape unit is set with the system variable TUSTEP_MT1.
	= name		Under UNIX and VMS: name of the system variable which sets the name of the tape unit.
OWNER	=		Name of the owner which is to appear on the tape's label. The name may be up to 14 characters long.
	= -		The tape is already labelled - remove label.

Default settings

An asterisk (*) marks the default setting for a specification. The default setting for the specification CODE depends on the operating system being used.

ASCII: UNIX, VMS
EBCDIC: BS2000, MVS, VM/CMS

Features

This command can be used to label unlabelled magnetic tapes or unlabel tapes already labelled. When a tape is labelled, a volume label is written at the beginning of the tape so that it can be written with the commands #MTWRITE and #MTCOPY. When a tape is unlabelled, its volume label will be overwritted with zeros, so that the tape can be later written with programs that require unlabelled tapes.

Warning

Whenever a label is either added or removed, all data on the tape will be lost.

Restrictions

This command has not been implemented for the MVS operating system.

Reading files from magnetic tape

MTREAD

Command:

#MTREAD

Specifications:

TAPE	=		Reel (ID) number of the magnetic tape from which the file is to be read. The reel number may consist of alphanumeric characters only.
NUMBER	= -STD-	*	The last file (i.e. the most recent version) having the corresponding name is to be read.
	= n		The first file having the corresponding name is to be read, with the search starting at file position n on the tape.
FILE	= file		Name of the file to which the data read from the tape are to be written. More than one file name is allowed.
LABEL	= -STD-	*	The names of the files read from the tape are identical to those given for the specification FILE.
	= name		The files to be read from tape have the names entered here. The number of names must agree with the number of files given in the specification FILE.
ERASE	= -	*	If the files given for the specification FILE already contain data, their data are to be retained.
	= +		If the files given for the specification FILE already contain data, their data are to be erased beforehand.
CODE	= ASCII		The data have been written to tape in ASCII-code (with an ANSI label).
	= EBCDIC		The data have been written to tape in EBCDIC-code (with an IBM label).
DEVICE	= -STD-	*	A tape unit set by the operating system is to be used; under UNIX and VMS the tape unit is set with the system variable TUSTEP_MT1.

= name Under UNIX and VMS: name of the system variable which specifies the name of the tape unit.

Default settings

An asterisk (*) marks the default setting for a specification. The default setting for the specification CODE depends on the operating system being used.

ASCII: UNIX, VMS
EBCDIC: BS2000, MVS, VM/CMS

Features:

With this command, files can be copied from tape to disk.

Restrictions:

All files recorded on the magnetic tape must have been written with the commands #MTWRITE or #MTCOPY.

When giving a value for the specification CODE, the following points should be considered:

- BS2000, UNIX, VM/CMS and VMS versions: if the wrong code is given for the tape, it is automatically adjusted to the code with which the tape has been written.

- MVS version: only the code used by the computing center to label the magnetic tape is allowed. If a different code is specified here, the operator will not be able to assign the tape properly.

Under the operating system MVS, some computing centers can only use magnetic tapes in batch jobs. In this case, the error message "No magnetic tape permission" will appear. Batch jobs can be started with the command #EXECUTE (cf. page 97). At some computing centers, a standard macro called #*MTR is available which will start such a batch job. It has the same specifications as the command #MTREAD. Additional information relating to this can be obtained with the command #INFORM,*MTR .

Warning:

When using the ASCII code for writing or reading magnetic tapes under the operating system MVS, most computing centers use 7-bit mode (instead of 8-bit mode). In this case, characters encoded with the escape character "^" (e.g. the German umlauts ä, ö and ü) will be lost.

If a file on the tape contains such characters and if only a 7-bit data transfer is possible, the tape cannot be read without losing these characters. The loss of these characters can be prevented by using a tape to which the files are written in EBCDIC-code or which contains only 7-bit ASCII-characters.

The latter can be achieved by using the command #CONVERT,..., MODE=-1,CODE=- before the data are written onto magnetic tape. This command converts the data into 7-bit ASCII-characters. After these data have been read from the magnetic tape, they can be converted back into TUSTEP code with the command #CONVERT,...,MODE=+1,CODE=-.

Writing files to magnetic tape

MTWRITE

Command:

#MTWRITE

Specifications:

TAPE	=		Reel (ID) number of the magnetic tape to which the file is to be written. The reel number may consist of alphanumeric characters only.
NUMBER	= -STD-	*	The files are to be written after the last file on the tape.
	= n		The file is to be written onto the tape starting at file position n.
FILE	= name		Name of the file to be written onto the tape. More than one file name is allowed.
LABEL	= -STD-	*	The name of the file on the tape shall be the same as the name given for the specification FILE.
	= name		The file on the tape shall be given the name specified here. The number of names must agree with the number of files given for the specification FILE.
DENSITY	= 1600		Under BS200 and MVS: the data are to be written to the tape with 1600 bpi.
	= 6250	*	Under BS2000 and MVS: the data are to be written to the tape with 6250 bpi.
CODE	= ASCII		The data are to be written to the tape in ASCII code (with an ANSI label).
	= EBCDIC		The data are to be written to the tape in EBCDIC code (with an IBM-label).
DEVICE	= -STD-	*	A tape unit set by the operating system is to be used; under UNIX and VMS the tape unit is set with the system variable TUSTEP_MT1.
	= name		Under UNIX and VMS: name of the system variable which specifies the name of the tape unit.
LISTING	= +	*	The list of all files on the tape are to written to the journal.

	= -STD-		The list of all files on the tape are to be written to the standard LISTING file.
	= file		Name of the file to which the list of all files on the tape are to be written.
ERASE	= -	*	If the LISTING file already contains data, they are to be retained.
	= +		If the LISTING file already contains data, they are to be erased beforehand.

Default values:

The default values for the specifications are marked with an asterisk (*). The default value for the specification CODE depends on the operating system being used:

ASCII: UNIX, VMS
EBCDIC: BS2000, MVS, VM/CMS

Features:

With this command, files can be copied from disk to tape (for data backup and data exchange).

After the files are copied, a list of the files located on the tape can be created. This list corresponds to that generated with the command #INFORM (see page 111).

Warning:

If the files are not copied after the last file on the tape, the file which is thus overwritten and all following files already on the tape will be lost.

to be copied, the specification SNUMBER=1 must be given.

Note:

If the tape being written to is used for the first time or is written to at its beginning, the specification NUMBER must be 1 (NUMBER=1).

The files can be copied back from tape to disk with the command #MTREAD (see page 141)

Restrictions:

Files must be written to the magnetic tape either from the beginning of the tape, or all files already present on the tape must have been written with the command #MTWRITE or #MTCOPY.

The magnetic tape must be labeled. Unlabeled tapes will be rejected for reasons of data security. They can be labeled with the command #MTLABEL (see page 139).

Because standard labels are used, all files on a tape must be written to the tape in the same code.

When giving a value for the specification CODE, the following points should be considered:

- BS2000, UNIX, VM/CMS and VMS versions: When the first file is written to the destination tape, the code can be freely chosen; if necessary, the tape will be relabeled. If a file is not to be written to the beginning of the tape and the wrong code has been given by mistake, the code is automatically adjusted to the code used on the tape up to this point.

- MVS version: only the code used by the computing center to label the source tape and the destination tape is allowed. If a different code is given, the operator will not be able to assign the tape properly.

Under the operating system MVS, some computing centers can only use magnetic tapes in batch mode. In this case, the error message "No magnetic tape permission" will appear. Batch jobs can be started with the command #EXECUTE (cf. page 97). At some computing centers, a standard macro called #*MTW is available which will start such a batch job. It has the same specifications as the command #MTWRITE. Additional information relating to this can be obtained with the command #INFORM,*MTW.

Warning:

When using ASCII code for writing or reading magnetic tapes under the operating system MVS, most computing centers use 7-bit mode (instead of 8-bit mode). In this case, characters encoded with the escape character "^" (e.g. the German umlauts ä, ö and ü) will be lost.

If a file on the tape contains such characters and if only a 7-bit data transfer is possible, the tape cannot be read without losing these characters. The loss of these characters can be prevented by using a tape to which the files are written in EBCDIC code or which contains only 7-bit ASCII-characters.

The latter can be achieved by using the command #CONVERT,..., MODE=-1CODE=- before the data are written onto magnetic tape. This command converts the data into 7-bit ASCII characters. After these data have been read from the magnetic tape, they can be converted back into TUSTEP code with the command #CONVERT,...,MODE=+1,CODE=-.

Warning:

If the files are not copied after the last file on the tape, the file which is thus overwritten and all following files already on this tape will be lost.

Note:

If the tape being written to is used for the first time or is written to at its beginning, NUMBER=1 must be specified.

The files can be copied back from tape to disk with the command #MTREAD (cf. page 141)

Restrictions:

Files must be written onto the magnetic tape either from the beginning of the tape, or all files already present on the tape must have been written with the commands #MTWRITE or #MTCOPY.

The magnetic tape must be labeled. Unlabeled tapes are rejected for reasons of data security.

Because standard labels are used, all files on a tape must be written onto the tape in the same code.

When giving a value for the specification CODE, the following points have to be considered:

- VM/CMS and VMS versions: When the first file is written onto tape, the code can freely be chosen; if necessary, the tape is relabeled. If a file is not to be written at the beginning of the tape, the code is adjusted to the code of the files already present on the tape, if a different code has been specified.

- MVS version: Only the code in which the magnetic tape was labeled by the computing center is allowed. If another code is given for the specification CODE, the operator cannot assign the tape.

When using the operating system MVS, at some computing centers magnetic tapes can only be used in batch jobs. In this case, the error message "No magnetic tape permission" will appear. Batch jobs can be started with the command #EXECUTE (cf. page 97). At some computing centers (e.g. in Tübingen), a standard macro called #*MTW is available which will start such a batch job. It has the same specifications as the command #MTWRITE. Further information relating to this can be obtained with the command #INFORM,*MTW.

Warning:

When using the ASCII code for writing or reading magnetic tapes under the operating system MVS, most computing centers use 7-bit mode (instead of 8-bit mode) for data transfer. In this case, the characters of the 8-bit TUSTEP character set (e.g. the German umlauts ä, ö and ü) will be lost.

To avoid this, the data can be converted into 7-bit ASCII-characters with the command #CONVERT,...,
,0 MODE=-1,CODE=- before they are written to magnetic tape. After these data have been read from magnetic tape, they can be

converted back to their TUSTEP code with the command #CONVERT,...,MODE=+1,CODE=-.

Renumbering references

NUMBER

Command:

#NUMBER

Specifications:

SOURCE	= file		Name of the file containing the data whose numbers and/or references are to be updated
	= -STD-		The standard TEXT file contains the data into whose numbers and/or references are to be updated.
DESTINATION	= file		Name of the file to which the data with the updated numbers and/or references are to be written
	= -STD-		The data with the updated numbers and/or references are to be written to the standard TEXT file.
MODE	= -	*	Normal case
	= PUT		The concordance (list of concordant numbers) of the old and the new numbering is to be written to the CONCORDANCE file. Any data already in this file will be erased regardless of what has been given for the ERASE specification.
	= GET		The concordance of the old and the new numbering is to be read from the CONCORDANCE file.
	= ADD		The concordance of the old and the new numbering in the CONCORDANCE file is to be supplemented.
ERASE	= -	*	If the DESTINATION file or the LISTING file already contains data, they are to be retained.
	= +		If the DESTINATION file or the LISTING file already contains data, they are to be erased beforehand.
PARAMETER	= file		Name of the file containing parameters
	= *		The parameters follow the command and are ended by *EOF.
CONCORDANCE	= -	*	Normal case

	= file		Name of the file to which the concordance of the old and the new numbering is to be written or which contains the concordance.
	= -STD-		The concordance of the old and the new numbering is to be written to the standard DATA file or is to be read from the standard DATA file.
LISTING	= -		No protocol of the concordance of the old and the new numbering.
	= +		A protocol of the concordance of the old and the new numbering is to be written into the journal.
	= -STD-	*	A protocol of the concordance of the old and the new numbering is to be written into the standard LISTING file.
	= file		Name of the file to which a protocol of the concordance of the old and the new numbering is to be written.

Features:

This program is used to update (running) numbers and their respective references. this purpose, running numbers can be inserted at appropriately marked positions in the text. (Old) numbers already present at these positions will be replaced. At the same time, any appropriately marked references that refer to old numbers can be updated.

In addition, this command can also be used to update references referring to a page-line number at appropriately marked positions after the page-line makeup has been altered.

Description

For this program there is a special description with the name NU (cf. #MANUAL on page 128).

Opening files

OPEN

Command:

#OPEN

Specifications:

READ	= file		Name of the file to be opened for reading. More than one file name is allowed.
	= -	*	No files to be opened for reading.
	= +	*	Files selected with the specifications PROJECT/CARRIER/POSITIVE/NEGATIVE are to be opened for reading.
WRITE	= file		Name of the file to be opened for writing. More than one file name is allowed.
	= -	*	No files to be opened for writing.
	= +	*	Files selected with the specifications PROJECT/CARRIER/POSITIVE/NEGATIVE are to be opened for writing.
SCRATCH	= -	*	(no longer defined)
PROJECT	= name		Name of the project from which files are to be opened.
	= +		Files from the current project are to be opened.
	= -STD-		Files of the project set at TUSTEP initialization are to be opened.
	= -	*	No file belonging to any particular project is to be opened.
CARRIER	= -STD-		Files to be opened are located on a data carrier (disk or disk drive) set by the system. Under DOS, UNIX and VMS, the carrier for permanent files is set with the system variable TUSTEP_DSK.
	= name		Under DOS, UNIX and VMS: name of the system variable which sets the path specification for the files to be opened. Under DOS, specifying merely the letter of the drive is sufficient if the path specificaion contains no directory names.

POSITIVE	=	* No positive selection by character strings occuring in file names
	= ...	Only those files are to be opened whose filenames contain at least one of the character strings specified here.
NEGATIVE	=	* No negative selection by character strings occuring in file names
	= ...	Only those files are to be opened whose filenames do not contain any of the character strings specified here.

Features:

In order to access permanent (cataloged) files, they must first be opened with this command. Files opened for reading can be read only; files opened for writing can be both read and written to.

To open all files of a particular project, simply give "+" for the specification READ or WRITE, and the name of the project for the specification PROJECT.

To open all files whose name contains a particular character string, enter the character strings in the specification POSITIVE which must occur at least once in the file name for the file to be opened. The specification NEGATIVE can be used to specify character strings which must not be contained in the file name for the file to be opened. For a full description of these specifications, refer to the command #LIST on page 121.

Notes:

Any files having a name not recognized by TUSTEP will be ignored when files are opened by specifying "+" for READ or WRITE.

The specification PROJECT is only of significance if a "+" has been given for the specification READ or WRITE. If a file name has been supplied for these two specifications instead, the project name must be also specified along with the file name if the file is not part of the current project. In this case, giving the project name for the specification PROJECT will have no effect.

Up to 100 files (excluding those files used by TUSTEP internally) can be open in a single TUSTEP session. This also includes files created with the command #CREATE during the same session. If this limit is reached, existing files must either be closed with the command #CLOSE (cf. page 64) or erased with the command #ERASE (cf. page 93) before additional files can be opened.

Special note for the MVS version

Files opened for writing cannot be accessed by other jobs or runs.

Activating/deactivating parameter log

PARAMETER

Command:

#PARAMETER

Specifications:

MODE	= OFF	Deactivate logging of parameters
	= ON	Activate logging of parameters
	= NEU	Set new parameter interpretation
	= ALT	Set old parameter interpretation

Features:

This command is used to select whether the TUSTEP programs should
- list parameters into the journal (MODE=ON) or not (MODE=OFF).
- interpret parameters according to either the new or old conventions.

If no value is given for the specification MODE, only the current mode setting will be displayed. When TUSTEP is initialized, the parameter log is activated.

Parameters can be interpreted according to either "old" or "new" conventions. The old method of interpreting parameters is the one first used in TUSTEP and whose concept was based on the punch card, then the standard medium for the input of data and programs. The notation used here took into account the fact that punch cards ordinarily did not distinguish between uppercase and lowercase letters, and that umlauts were not capable of being interpreted. The "new" method of interpreting parameters has been designed with more modern input technology in mind; the limitations imposed by the "old" method therefore no longer apply.

Note

The old method of interpreting parameters is, however, still in effect as the default setting. Future versions of TUSTEP will use the new method of parameter interpretation as the default setting. It is therefore recommended that this command be used in every command sequence for specifying which convention is to be used to interpret parameters.

For a detailed description of the differences between the old and new parameter conventions, see the chapter entitled "Parameters" starting on page 241).

Pause before executing remaining commands

PAUSE

Command

#PAUSE

Specifications none

Features

This command is used to interrupt the processing of a command sequence (at the position given in this command sequence). This can be useful when, for example, a different diskette is to be placed in the drive before the next command in a command sequence. Once activated, the command will inquire how to treat the remaining comands. The user can answer with one of three valid responses:

1) H (Hold) Before the remaining commands are carried out, priority commands shall be executed. Priority commands are then entered one by one (!). If an empty input line is sent to the computer (ENTER) instead of a priority command, the interrupted command sequence will continue.

2) C (Cancel) Cancel remaining commands

3) G (Go) The remaining commands are to be executed.

Preparing an index before sorting

PINDEX

Command:

#PINDEX

Specifications:

SOURCE	= file		Name of the file containing the data from which index entries are to be extracted or from which a KWIC index is to be prepared
	= -STD-		The standard TEXT file contains the data from which index entries are to be extracted or from which a KWIC index is to be prepared.
DESTINATION	= file		Name of the file to which the index entries or the entries for the KWIC index are to be written; more than one file name is allowed
	= -STD-		The index entries or the entries for the KWIC index are to be written into the standard TEXT file.
MODE	= +	*	Prepare index entries with references
	= -		Prepare index entries without references
	= KWIC		Prepare index entries for the KWIC index
ERASE	= -	*	If the DESTINATION file, the DATA file or the LISTING file already contains data, they are to be retained.
	= +		If the DESTINATION file, the DATA file or the LISTING file already contains data, they are to be erased beforehand.
PARAMETER	= file		Name of the file containing parameters.
	= *		The parameters follow the command and are ended by *EOF.
DATA	= -	*	The data are to be written in their entirety to the DESTINATION file
	= file		Name of the file to which the text part (i.e. text not required for sorting) of each record is to be written.
	= -STD-		The text part of each record is to be written to the standard DATA file.

LISTING	= -	*	No trace listing
	= +		Trace listing is to be written to the journal.
	= -STD-		Trace listing is to be written into the standard LISTING file.
	= file		Name of the file to which the trace listing is to be written.

Features:

This program can be used to

- decompose texts into appropriate entities for an index (e.g. for an index of word forms or a KWIC index).

- extract appropriately marked text parts to be used as index entries (e.g. for preparing an author index, subject index or geographical index). These entities are then prepared for subsequent sorting.

Description:

For this command, there is a special description with the name PI (cf. MANUAL, page 128).

Preparing data for sorting

PRESORT

Command:

#PRESORT

Specifications:

SOURCE	= file		Name of the file containing the data to be prepared for sorting
	= -STD-		The standard TEXT file contains the data to be prepared for sorting.
DESTINATION	= file		Name of the file to which the data prepared for sorting are to be written. More than one file name is allowed.
	= -STD-		The data prepared for sorting are to be written to the standard TEXT file.
MODE	= -		Generate sort key only.
	= +	*	Generate sort key and add REF/TYPE/STB.
	= R		Input data contain REF/TYPE/STB; add sort key
	= K		Input data are correcting instructions; add correction key and, if indicated by parameter, sort key.
	= S		Input data are correcting instructions with correction key; add sort key.
ERASE	= -	*	If the DESTINATION file, DATA file or LISTING file already contains data, they are to be retained.
	= +		If the DESTINATION file, DATA file or LISTING file already contains data, they are to be erased beforehand.
PARAMETER	= file		Name of the file containing parameters
	= *		The parameters follow the command and are ended by *EOF.
DATA	= -	*	The data are to be written to the DESTINATION file in their entirety.
	= file		Name of the file to which the text part of each record is to be written.

	= -STD-		The text part of each record is to be written into the standard DATA file.
LISTING	= -	*	No trace listing
	= +		The trace listing is to be written to the journal.
	= -STD-		The trace listing is to be written to the standard LISTING file.
	= file		Name of the file to which the trace listing is to be written.

Features:

With this command, text units (consisting of one or more input records, which may also be correcting instructions) can be prepared for sorting. It is possible to define the criteria for sorting.

Description:

For this command, there is a special description with the name PS (cf. the command #MANUAL, page 128).

Print output

PRINT

Command:

#PRINT

Specifications:

LISTING	= file		Name of the file containing the data to be printed.
	= -STD-	*	The standard LISTING file contains the data to be printed.
TYPE	=		Type of printer to be used. The types of printers available depends on the actual computer being used. To obtain a list of these, use the command #LIST, PRINTERS
DEVICE	= ...		Name of the printer to be used. The names of the available printers may be obtained from the information bulletin of the respective computing center.
	= +		Output to the journal (i.e. in interactive mode, the screen)
	= -		No output to a printer. If no file has been given for the specification FILE, only the data in the LISTING file will be syntactically checked.
COPIES	= 1	*	The data are to be printed once.
	= n		The data are to be printed n times.
	= n*m		The data are to be printed n*m times (see below)
PREFIX	= -	*	No prefix
	= *		The prefix follows the command #PRINT and is ended by *EOF.
	= file		Name of the file containing the prefix.
PAGES	= -	*	The file is to be printed in its entirety
	= n / n-m		Only page n / pages n to m are to be printed. More than one page/range may be specified here.
COVER	= -STD-	*	Whether TUSTEP supplies a cover for the printed output (cover and end sheets with

user's name) depends on the computer and printer being used.

= - No cover to be supplied by TUSTEP. This specification will be ignored if the computing center does not allow this for certain printer types.

= + TUSTEP is to supply a cover for the printed output.

FILE = - * Output to the printer given in the specification DEVICE.

= file Name of the system file to which the printer control codes are to be be written. More than one file name may be specified.

ERASE = - * If the file given in the specification FILE already contain data, they are to be retained.

= + If the file given in the specification FILE already contains data, they are to be erased beforehand.

OPTIONS = ... * Specifications for modifying the generated printer control codes.

SUPPLEMENT = ... Name of a system variable containing supplementary specifications for the operating system (cf. system variable TUSTEP LPR, page 42). The relevant and necessary supplements for each case may be obtained from the information bulletin of the respective computing center.

PORTION = -STD- The printout should be subdivided into portions of 500 pages each (for line printers), or portions of 200 pages each (for dot matrix and laser printers).

= n The printout should be divided into portions of n pages each.

Features:

This command is used to print files which have already been prepared for printing (listing files). Normally, these are files which have been written as listing files in a previously run TUSTEP program. To print a file which is not a LISTING file, its data must first be prepared for printing. This can be carried out with the TUSTEP program FORMAT if the file contains the necessary formatting instructions that are interpreted by this program. Otherwise, the TUSTEP program GLISTING can be used.

If only certain pages are to be printed, these can be selected with the specification PAGES. Page selection is based on the

record numbers in the LISTING file, not on any page number in the page header. If you wish to select pages based on the page header number, use the TUSTEP program GLISTING instead.

If more than one copy is to be made of a file (as specified in the specification COPIES), some operating systems insert a new cover which may consist of several pages) between each copy. To save paper, especially when printing many copies of a short file, it is advisable to use the formula n*m instead of n in the specification COPIES. In this case, the file specified as the LISTING file will be copied m times to an internal intermediate file, which is then printed out n times (with only n covers).

If the printer specified for output cannot be directly accessed (e.g. if it is connected to another PC), a system file (TYPE=SDF) can be specified for FILE. This file will contain all the printer control codes for specified printer type. This system file can then be sent to the printer using the appropriate commands of the operating system (for example, the command PRINT for a PC running under MS-DOS). If more than one file has been specified for FILE, only the number of pages specified for PORTION will be written into a file, after which the next file will be written to.

For printing on EPSON compatible matrix printers, the following options can be specified:

BD bidirectional printing
UD unidirectional printing (default)

H11 page length 11"
H12 page length 12"

LQ letter quality printing (default)
DRAFT draft quality printing

Restrictions concerning the specification DEVICE

Under DOS, only the "+" specification for DEVICE has any effect. All other specifications have no effect. If no other specification has been given for FILE, the print output will be directed to the printer previously set with the DOS command PRINT at the operating system level.

Renaming files and project names

RENAME

Command:

#RENAME

Specifications:

FILENAME = - * No file to be renamed

= name1:name2 Old and new names of the file to be renamed. More than one name pair may be specified.

PROJECTNAME= - * No renaming of a project

= name1:name2 Old and new name of the project to be renamed. More than one name pair may be specified.

CARRIER = -STD- * The projects to be renamed are located on the carrier preset by the system. Under DOS, UNIX and VMS the carrier for permanent files is set by the system variable TUSTEP_DSK

= name Under DOS, UNIX and VMS: name of the system variable containing the path specification for the projects to be renamed. Under DOS ll. Under DOS, merely specifying the relevant disk drive is sufficient if the path contains no other directory names.

Features:

With this command, the names of files can be altered (i.e. files can be renamed). Under DOS, UNIX and VMS: If the old and the new file name are each given a different project, the old file will be moved to the project of the new file (i.e. it will be deleted from the old project and entered in the new project). The "new" project must already exist (i.e. must be opened beforehand).

This command can also be used to rename projects (directories). Projects preset by the computing center's operating center (system manager) may not be renamed.

Resetting/terminating TUSTEP

RESET

Command:

#TRESET

Specifications:

TERMINATE	= -	* TUSTEP is not to be terminated.
	= +	TUSTEP is to be terminated.
WIPE	= +	The data in the temporary files are to be overwritten.
	= -	The data in the temporary files do not need to be overwritten.

Features:

This command is used to end a TUSTEP session and thus:

- delete all standard files
- delete all temporary files (scratch files)
- close all opened files

The specification WIPE can be used to specify whether the data contained in temporary (scratch) files are to be overwritten (data security!). If nothing is given for this specification, the data will be overwritten unless another mode has been set with the command #WIPE (cf. page 173).

In case TUSTEP is not to be terminated, a new TUSTEP session will be started and initialized after the files have been either deleted or closed (cf. "Starting TUSTEP", page 48).

Restoring data

RESTORE

Command:

#RESTORE

Specifications:

SOURCE	= file		Name of the file containing the data to be restored (copied).
	= -STD-		The data to be restored (copied) are located in the standard TEXT file.
DESTINATION	= file		Name of the file to which the restored (copied) data are to be written.
	= -STD-		The restored (copied) data are to be written to the standard TEXT file.
MODE	= -STD-		Normal case
	= n		Records having the nth ascending order are to be copied.
ERASE	= -	*	If the DESTINATION file already contains data, they are to be retained.
	= +		If the DESTINATION file already contains data, they are to be erased beforehand.

Features:

This command is used to

- copy data from an incomplete (improperly terminated) file to another file.
- copy records in ascending order to another file. An ascending record sequence either ends or begins with a record number that is equal or less than the record number immediately preceding it.
- copy data from one file to another in order to eliminate any gaps in the file structure that may have arisen during editing and thus restore records to their logical order.

Note:

If the restored data are to be made accessible in the SOURCE file, they must be copied back from the DESTINATION file into the SOURCE file with an additional command, where ERASE=+. To avoid the extra task of recopying (especially when working with large files) and also make available the disk space freed by the

automatic reorgainzation of data, the following steps are to be taken:

- create a new file with a new name
  (using the command #CREATE, see page 79),
- restore the data from the old file to the new one
- delete the old file
  (using the command #ERASE, see page 93),
- rename the new file with that of the old file
  (using the command #RENAME, see page 163).

Sorting data/files

SORT

Command:

#SORT

Specifications:

SOURCE	= file		Name of the file containing the data to be sorted.
	= -STD-		The data to be sorted are located in the standard TEXT file.
DESTINATION	= file		Name of the file to which the sorted data are to be written.
	= -STD-		The sorted data are to be written to the standard TEXT file.
SORTFIELD	= sf-A		The data are to be sorted according to the sort field sf, in ascending order.
	= sf-D		The data are to be sorted according to the sort field sf, in descending order.
	= sf		as sf-A
			One of the following can be specified for sf:
			0 Sorting according to record numbers
			n-m The sort field starts with character position n and ends with character position m.
			n+m The sort field starts with character position n and is m characters long.
			More than one sort field is allowed. More than one sort field is possible.
ERASE	= -	*	If the DESTINATION file already contains data, they are to be retained.
	= +		If the DESTINATION file already contains data, they are to be erased beforehand.
DELETE	= -	*	After sorting, the records are to be written to the DESTINATION file unchanged.
	= n-m		Before the data are written to the DESTINATION file, the characters from

			position n to position m are to be deleted from each record.
	= n+m		Before the data are written into the DESTINATION file, m characters starting with position n are to be deleted from each record.
	= +		Only the data of the DATA file are to be written into the DESTINATION file in the sorted sequence.
DATA	= -	*	Normal case).
	= file		After sorting, the data are read from the given DATA file and written in the same order to the DESTINATION file.
	= -STD-		After sorting, the data are read from the standard DATA file and written in the same order to the DESTINATION file.
WIPE	= +		The data in the sort key files are to be overwritten before these files are deleted
	= -		The data in the sort key files do not need to be overwritten before these files are deleted.

Features:

This command is used to sort files. Here the order in which records are processed is determined by the content of the specified sort fields. If the contents of a sort field is identical for two or more records, their order is determined by the next respective sort field. If no other sort field has been specified, the order of such records will remain unaltered.

Data to be sorted must have been previously prepared with the program PRESORT or PINDEX. Exceptions to this rule: only when a sort of record numbers is desired. The arrangement of records resulting from the preparation of data is described in the respective chapters of these programs. This arrangement and the sequence in which the individual record fields are allocated during sorting are expressed in the specification for the SORTFIELD.

In most cases the sort field corresponds to the sort key generated by the programs PRESORT and PINDEX. Here the length of this sort field is the result of the overall length of the individual sort key set by the parameter SSL. It ordinarily begins at position 1 (if the records do not start with a reference marker; if so, then it begins at the first position after the reference marker, which is usually position 17 unless otherwise specified by parameter IRL).

If the resorted data are to be located in their original file, the same file must be specified for SOURCE and DESTINATION. In this case a "+" must be entered for the specification ERASE in order to erase the unsorted data before the sorted data are written to file.

The specification DELETE can be used to eliminate sort keys no longer needed after sorting is completed.

The sorting process requires intermediate files automatically created by TUSTEP, which are then erased after sorting is completed. Should the user wish to overwrite the data in these files before they are erased for reasons of data security, the corresponding specification in WIPE can be selected. If no specification is made, the data will be automatically overwritten unless a different mode has been set for the command #WIPE (see page 173).

Listing TUSTEP statistics

STATISTICS

Command:

#STATISTICS

Specifications:

LISTING	= +	*	Statistics are to be written to the journal
	= -STD-		Statistics are to be written to the standard LISTING file
	= file		Name of the file to which the statistics are to be written
ERASE	= -		If the LISTING file already contains data, they are to be retained.
	= +		If the LISTING file already contains data, they are to be erased beforehand.

Features:

With this command TUSTEP user-statistics can be listed or prepared for printing.

However, statistics are kept for the user only if a statistics file having the name "TUSTEP.USE" has been previously created with the command
#CREATE,*TUSTEP.USE,SEQ-P (cf. page 79).

The statistics data can be erased with the command:
#ERASE,DATA=*TUSTEP.USE (cf. page 93).
In this case, a new user statistics log will be started. If the statistics file is deleted with the command
#ERASE,FILE=*TUSTEP.USE (cf. page 93),
statistics will no longer be recorded for the user.

Setting/clearing sense switches

SWITCH

Command:

#SWITCH

Specifications:

SET	= SWn		Set sense switch n. The letters SW may be omitted. More than one sense switch can be specified.
	= -	*	Do not set a sense switch
CLEAR	= SWn		Clear sense switch n. The letters SW may be omitted. More than one sense switch is allowed.
	= -	*	Do not clear a sense switch

Features:

With this command, the sense switches can be set or cleared. In TUSTEP there are seven sense switches, numbered from 1 to 7. When TUSTEP is initialized, all sense switches are cleared.

If the command is called up without any specifications, only the current mode of the sense switches will be displayed.

Note:

The sense switches can be used for controling TUSTEP programs.

If +n or -n is given in column 4 and 5 of a parameter, the parameter will be interpreted only if the sense switch n has the same setting (+n for set; -n for cleared).

- These sense switches can be inquired in command macros; which can be used to determine the further processing of the macro.
- For the program COPY, these sense switches can be activated (with the appropriate parameters) at the beginning of the internal program sense switch and either set or erased at the end of the internal program sense switch.

Time

TIME

Command:

#TIME

Specifications: none

Features

This command displays the current time and (except in the MS-DOS version) the total amount of computing time that has been used.

Wipe (overwrite) data - on/off

WIPE

Command:

#WIPE

Specifications:

MODE	= OFF	Turn off data wipe function
	= ON	Turn on data wipe finction

Features:

With this command, the data wipe function can be set, i.e. whether the data contained in files to be deleted are to be overwritten (MODE=ON) or not (MODE=OFF). If nothing is given for the specification MODE, only the current mode will be displayed. When TUSTEP is initialized, the wipe function is turned on.

Overwriting data concerns not only the explicit deletion of files with the command #ERASE, but also the automatic deletion of files with the commands #CLOSE, #RESET and #SORT. Yet each of these commands also allows the user to decide in each case whether the data are to be overwritten or not, regardless of the current setting made with the command #WIPE.

Note:

With some operating systems, it is possible to re-access data from deleted files. While this may be advantageous for the PC user, it is not desirable for mainframe operations for reasons of data security, since other users would have potential access to private data. For this reason, TUSTEP has a standard feature which overwrites the data in such files with zeros before they are deleted.

E d i t o r

Survey:

Starting the Editor The procedure for starting the Editor is described in the chapter "Commands" on page 91.

Character codes:

Coding instructions for individual characters is described in the chapter "Character sets" (see page 294).

Numbering records in the Editor:

Each record in a TUSTEP file has its own record number. For a description of how records are numbered in program mode and text mode, and how these record numbers are interpreted, consult the sections "Record numbering in program mode" (page 24) and "Record numbering in text mode" (page 23).

In order for a file to be processed by the Editor, the following conditions must be met:

- The record number 0 or 0.0 may not be used.
- The records must be numbered in ascending order.
- A record number must not occur more than once (i.e must be unique).

Normally, record numbers start with 1 or 1.1. Smaller record numbers may be used (e.g. 0/1 or 0.1) if, for example, a new record is to be written in front of the first record.

Screen display of records

TUSTEP will try to fit each record into its own line on screen. Records that do no fit into a single line will be extended as a continuation line.

A continuation line will start at a blank space between words whenever possible. If no such blank occurs in the data concerned, the line will be written to its fullest extent and a continuation line will be automatically generated. A gravis character (`) is inserted at the end of the line indicates that no blank space is present in the data entered when it is wrapped to the next line on screen.

Assuming that record number display has not been suppressed (according to the relevant modus setting), the record number of each record will be displayed in the first line of each record. A continuation line can be recognized by the fact that it is preceded by no number. E.g. the following example:

```
`  1.1  | Short record within one record. Followed by a
`  1.2  | longer record which cannot fit into a single
`       | line and which requires two continuation
`       | lines. This record is followed by a brief
`  1.3  | record which fits into a single line.
```

Calling up online help

Online help (see command #HELP page 109) can be accessed with the Editor instruction HELP or with the control command HELP. If online help is accessed with the control command HELP, the Editor will be in the same condition after the HELP command as it was before the command was given (online help is effectively ignored for Editor purposes).

Interrupting an Editor instruction

Extended instructions and search instructions may be interrupted in the Editor with the control command INTRPT (cf. page 61) and then entering the instruction I at the prompt:

```
PROGRAM INTERRUPT- Enter instruction>
```

Reorganizing files

Files which are continually processed by the Editor should be reorganized from time to time as described in the section "Reorganizing a TUSTEP file" (page 24).

Note:

The abbreviations POS, RAN, LIMI, CSST, CSCT, and CSPT are explained starting on page 207.

If changes are made on screen to a file which has been opened for reading only, the corresponding error message will appear. Afterwards, any new instruction will not be accepted until the screen alteration has been either declared invalid with the control command IGNORE (see page 228) or deleted with the control command CLEAR (see page 225). If the Editor is to be exited (e.g. in order to open the file for writing), this can be accomplished with the control command CANCEL (see page 228); here too many screen changes will be ignored.

In all instructions the comma which immediately follows the instruction letter(s) may be omitted if it is not immediately followed by another comma.

Instructions which are longer than a single line on screen may be written in the screen portion where data are normally displayed. If data are already located here, they may be previously erased from the screen with CLEAR. If continuation lines are required for an instruction, these do not have to be marked as such. Please remember that blanks at the end of an instruction line will be ignored.

Basic instructions:

Ending:

B Ends the Editor.

The Editor must always be ended with this instruction or with the control command CANCEL. Otherwise a file opened for writing will not be properly terminated.

Data from incomplete files (i.e. those not properly terminated, however, can usually be made accessible by copying them to another file with the command #RESTORE (see page 165).

Inserting, entering:

IE Enters records to an empty file or at the end of the file.

Each line on the screen results in a record in the file. Entering can be finished by entering an empty record with the ENTER key or a function key.

Note: During data input the control command CR has the same effect as the control command LF. To send data to the computer, the control command ENTER must be used.

I,POS#N Inserts N records starting at record position POS (if a record with the record number POS does not yet exist) or after the record position POS (if there is already a record with record number POS).

N can be an estimated maximum. If N=1, #N can be omitted. Each line on the screen results in one file record. If less records than specified are to be inserted, inserting can be finished by entering an empty entry with ENTER or a function key.

Note: During data input the control command CR has the same effect as the control command LF. To send data to the computer, the control command ENTER must be used.

The way in which the inserted records are numbered depends on how many record numbers are available at this position. The numbering increment selected will be the largest possible of the increments 1/0, 0/1, 0/01, 0/001 (the last mentioned is only possible in text mode). If a different increment is to be used for numbering the records, it can be specified after POS. POS and desired increment must be separated by a semicolon (e.g. 23;0/2#30). The increment may be expressed as a line number and/or a distinction number. If not enough record numbers are available for the selected increment, the specified increment will be ignored and a smaller increment will be used.

II,TEXT,RAN:COL Inserts the character string TEXT into every record of the range RAN.

A delimiter character must be placed before and after the character string TEXT. This character can be freely chosen by the user but it must not occur in the character string itself. For coding the TEXT character string, the same rules apply those used to code data (see "Character codes" page 176).

:COL may be omitted. In this case, the character string TEXT is inserted at the end of each record. If the value N is given for COL, the character string is inserted starting at position N of each record (i.e. the characters from position N to the end of the record are moved to the right). If +N is given, the character string is inserted after the character position N (counting the characters from the beginning of the record). If -N is given, the characters are counted from the end of the record and the character string is inserted in front of the respective character position.

If the value N1-N2 is given for COL, the character string in the columns N1 to N2 is replaced by the character string TEXT. The characters after column N2 up to the end of the record are moved to the left if the character string TEXT is shorter than the character string N1-N2, or to the right if it is longer. If only the character in column N is to be replaced, the value N-N must be given. (Giving only N would insert the character string TEXT at position N instead of replacing the character N).

IB,RAN1,RAN2 Inserts the records of the range RAN1 before each record of the range RAN2.

IA,RAN1,RAN2 Inserts the records of the range RAN1 after each record of the range RAN2.

If the range into which the records are to be inserted contains more than 10 records, the user will be asked beforehand if records should be inserted at so many places. The user may respond with either **Y** (yes) or **N** (no), or a new instruction may be given (in which case the records are also not inserted). The number of places where insertion may occur before this message will appear can be specified after RAN:COL or RAN2 and separated from it by a number sign (e.g. (23,-1)#30).

Showing:

SB Shows the records from the beginning of the file.

SE Shows the records from the end of the file.

SB,POS Shows the records starting with record position POS.

SA,POS Shows the records around record position POS.

ST,POS Shows the records up to record position POS.

S,POS Shows the records beginning with record position POS / the records around record position POS / the records up to record position POS, depending on which of the instructions **SB**,POS / **SA**,POS / **ST**,POS has last been given.If none of these instructions has been given, **S**,POS has the same effect as **SA**,POS.

After giving a show instruction, the following entries are possible:

empty entry: scrolls
+ Scrolls forward; thereafter, each empty entry results in a further scroll towards the end of the file.
- Scrolls backward; thereafter, each empty entry results in a further scroll towards the beginning of the file.
different instruction: Showing is interrupted and the instruction given here will be executed.

Correcting:

Every record having its own record number as displayed by the Editor (e.g. with a show instruction) can be corrected directly on screen. The corrected record can be then sent to the computer by pressing the ENTER key or a function key. It is also possible to send more than one altered record at a time. However, in many cases a specific correcting instruction may be more appropriate:

RAN Shows a record or a range for correcting.

This instruction has the following advantages over a show instruction:

- If, for example, only one record (whose record number is known) is to be corrected, it takes less time to list this single record by itself than it does to list several records covering the whole screen by using a show instruction.

- If a large range is to be corrected, a show instruction would have to be given after correcting each screen in order to display the following records. With this command, however, the following records are listed automatically as soon as the the corrected records have been sent to the computer.

In the file, only records which have been sent to the computer after being corrected will be altered. Records which have not been sent to the computer remain unaltered! Of particular note here: deleting a record and its record number from the screen does not mean that this record is deleted from the file.

To erase one or more records on screen, the vertical line after the record number may be replaced by a minus sign; the text of the record must be erased on screen. (cp. control command DEL_REC on page 225). Such a record is not

erased in the file until it has been sent (along with any other altered records) to the computer.

To join two or more successive records on the screen to a single record, the vertical line located after the record number of the second record (or the following records) to be joined is to be replaced by a minus sign (cp. the control command JOIN on page 227). The records are not joined in the file until they (and any other altered recordes) have been sent to the computer.

To add new records to those already shown on screen, these new records and their record numbers can be inserted directly on screen: the vertical line between record number and text must be replaced by a plus sign (cp. control commands INS_LINE on page 222 and SPLIT on page 222). Such records are not inserted into the file until they (along with any other altered records) have been sent to the computer.

On display devices without a vertical line (this is usually the case with those using a German character set) an equals sign is used instead of the vertical line located between the record's number and its text.

The following contains several examples where the changes made on screen may not have produced the desired result. But first an important note:

The effects of any changes made on screen is determined only by the screen content at the time such changes are sent to the computer. The contents of the screen before any changes were made is no longer relevant; no comparison will be made of screen contents before and after alterations in order to determine any differences and write these to file.

Before changing:	After changing:	Result in the file:
` 1.1 | one ` 1.2 | two ` | three	` 1.2 | two ` | three	1.1 one 1.2 two three

Record 1.1 remains unaltered. Reason: this record (more precisely, this record number) is no longer displayed on screen and thus (like all other file records not shown on screen) remains unaltered. Record 1.2 also remains unaltered. Reason: no changes have been made in this record. It the user wishes to delete record 1.1 in its entirety, the record number must be marked with a "-" (e.g. using the control command DEL_REC) and the text of the record must be deleted from the screen. Another possibility would be to delete this record using the instruction "d!1.1".

Before changing:	After changing:	Result in the file:
` 1.1 | one ` 1.2 | two ` | three	` 1.1 | one ` | two ` | three	1.1 one two three 1.2 two three

To record 1.1 the text "two three" has been added. Reason: there is no record number in front of "two" and "three"; these lines are therefore regarded as continution lines of the previous line. Record 1.2 remains unaltered. Reason: this record (more precisely, this record number) is no longer displayed on screen and thus (like all other file records not shown on screen) remains unaltered. If the user wishes to append the contents of record 1.2 to that of record 1.1 and then delete record 1.2, the record number 1.2 must be marked with a "-".

Before changing:	After changing:	Result in the file:
` 1.1 | one	` 1.1 | one	1.1 one two three
` 1.2 | two	` 1.2 - two	
` | three	` | three	

Record 1.2 is deleted. Reason: the record number has been marked with a "-". To record 1.1 the text "two three" has been appended. Reason: record number 1.2 has been marked with a "-", which deletes this record. The text "two" following the deleted record number is now regarded as a continuation of the preceding line, as if the line containing "two" has no record number preceding it. Similarly, the text "three" is preceded by no record number, which makes this line a continuation line of the preceding line. It the user wishes to delete record 1.2 in its entirety, the record number must be marked with a "-" (e.g. using the control command DEL_REC) and the record's text ("two three") must be deleted from the screen.

Deleting records

D!,RAN Deletes a record or range

If the range to be deleted contains more than 10 records, the user will be asked beforehand if so many records should be deleted. The user may respond with either Y (yes) or N (no) or a new instruction may be given (in which case the records are also not deleted). The number of records that can be deleted before this message will appear can be given after RAN and separated from it by a number sign (e.g. (23,-1)#30).

Copying records

C,RAN,POS Copies a record or a range.

C,FILE,RAN,POS Copies a record or a ran from a different file.

C,FILE,SEGMENT,POS Copies a segment from a different file.

The record or range RAN will be copied into the file starting at record position POS (if a record with the record number POS does not yet exist) or after record position POS (if there is already a record with the record number POS). If the record or the range is to be copied to the end of the file, POS may be omitted. In text mode, the

first new record copied to the end of the file is given the number n.1, where n is one page number larger than that of the file's last record.

The way in which the inserted records are numbered depends on how many record numbers are available at this position. The numbering increment selected will be the largest possible of the increments 1/0, 0/1, 0/01, 0/001 (the last mentioned is only possible in text mode). If a different increment is to be used for numbering the records, it can be given after POS. The POS and increment must be separated by a semicolon (e.g. 23;0/2#30). The increment may be expressed as a line number and/or a distinction number. If there is not enough record numbers available for the selected increment, the specified increment will be ignored and a smaller increment will be used.

If the range to be copied contains more than 10 records, the user will be asked beforehand if so many records should be copied. The user may respond with either Y (yes) or N (no), or a new instruction may be given (in which case the records are also not copied). The number of records that can be copied before this message will appear can be given after RAN and separated from it by a numbers sign (e.g. (23,-1)#30).

Renumbering and moving records

R,RAN,POS Renumbers a record or range and, if necessary, moves the respective record or range.

The record number(s) of a record or a range RAN will be altered and, if necessary, the record or the range will be moved, so that its new position in the file starts at record position POS (if a record with the record number POS does not yet exist) or after record position POS (if there is already a record with the record number POS). If the record or the range is to be moved to the end of the file, POS may be omitted. In text mode, the first new record copied to the end of the file is given the number n.1, where n is one page number greater than that in the file's last record. However, POS must be specified if RAN is omitted in order to renumber all records in the file.

The way in which the records are numbered depends on how many record numbers are available at this position. The numbering increment selected will be the largest possible of the increments 1/0, 0/1, 0/01, 0/001 (the last mentioned is only possible in text mode). If a different increment is to be used for numbering the records, this can be specified after POS. The POS and increment must be separated by a semicolon (e.g. 23;0/2#30). The increment may be expressed as a line number and/or a distinction number. If not enough record numbers are available for the selected increment, the specified increment will be ignored and a smaller increment will be used.

If the range to be renumbered contains more than 100 records, or the range to be moved more than 10 records, the user will be asked beforehand if so many records should be either renumbered or moved. The user may respond with either Y (yes) or N (no) or a new insttruction may be given (in which case the records are also neither renumbered nor moved). The number of records that can be renumbered or moved before this message will appear can be specified after RAN and separated from it by a numbers sign (e.g. (23,-1)#30).

Note: If the records of a file numbered in text mode are to be renumbered for switching to program mode, this can be accomplished with the following insztruction: R,,0.0/1;/1 (for more than 9999 records: R,,0.0/1;/01 - for more than 99999 records: R,,0.0/1;/001).

Organizational instructions

The Editor is designed in a way which normally allows TUSTEP files to be edited directly. If a copy of the file is to be edited instead, the original file can be copied outside of the editor to a working file, which is then copied back to the original file after it has been edited. This process of copying, however, can be accomplished within the Editor by using the LOAD and UNLOAD instructions, which copy files to and from the Editor file. System files can also be edited in this manner.

Listing, defining and deleting file names and segment names

Names can be defined for FILE and SEGMENT which are to be used in subsequent Load and Unload instructions if the names are omitted in these instructions. Names can also be defined by being specified in a Load or Unload instruction, thus replacing any names previously defined by the respective name instruction. The defined names are also replaced whenever a new EDITOR file is used (cf. file instruction V).

N Lists the names defined for FILE and SEGMENT.

N,FILE,- Defines the name for FILE and deletes the name defined for SEGMENT.

N,FILE,SEGMENT Defines the names for FILE and SEGMENT.

If FILE is omitted and only SEGMENT is given, only the name specified for SEGMENT will be redefined.

N! deletes names defined for FILE and SEGMENT.

Loading (copying) data:

a) of a file

L,FILE,- Copies the TUSTEP or system file FILE to the EDITOR file. The data in the EDITOR file will be overwritten.

If FILE is omitted, the name defined for FILE (cf. Name instruction) will be used. If a name has been defined for FILE but no name has been defined for SEGMENT, FILE can be omitted.

If the EDITOR file contains data which have not yet been unloaded to a file since being last altered, this instruction will be rejected. If the data are to be overwritten anyway, the following instruction must be given.

L!,FILE,- Copies the TUSTEP or system file FILE into the EDITOR file. If the EDITOR file already contains data, they will be overwritten without any warning message.

If FILE is omitted, its defined name (cf. Name instruction N) will be used. If a name has been defined for FILE but none for SEGMENT, ,FILE,- can be omitted.

b) of a segment

L,FILE,SEGMENT Copies the segment SEGMENT from the SEGMEMT file FILE to the EDITOR file. The data in the EDITOR file will be overwritten.

If FILE or ,FILE,SEGMENT is omitted, the defined names (cf. name instruction N) will used instead.

If the EDITOR file contains data which have not yet been unloaded to a file since being last altered, this command will be rejected. If the data are to be overwritten anyway, the following instruction must be used.

L!,FILE,SEGMENT Copies the segment SEGMENT from the SEGMENT file FILE to the EDITOR file. If the EDITOR file already contains data, they will be overwritten without any warning message.

If FILE or ,FILE,SEGMENT is omitted, the corresponding defined names (cf. name instruction N) will used instead.

c) of a segment file's table of contents

L,FILE,? Copies the table of contents from the segment file FILE to the EDITOR file. The data in the EDITOR file will be overwritten.

If FILE is omitted, its defined name (cf. name instruction N) will be used.

If the EDITOR file contains data which have not yet been unloaded to a file since their last alteration, the command will be rejected. If the data are to be overwritten anyway, the following instruction must be used.

L!,FILE,? Copies the table of contents from the segment file FILE to the EDITOR file. The data in the EDITOR file will be overwritten without any warning message.

If FILE is omitted, its defined name (cf. name instruction N) will be used.

Unloading (saving) data:

a) to a file

U,FILE,- Copies the EDITOR file to the TUSTEP or system file FILE.

If FILE is omitted, its defined name (cf. name instruction N) will be used. If a name has been defined for FILE but none for the segment, ,FILE,- can be omitted.

If the file FILE already contains data, this command will be rejected. If these data are to be overwritten anyway, the following instruction must be used.

U!,FILE,- Copies the EDITOR file to the TUSTEP or system file FILE.

If FILE is omitted, its defined name (cf. name instruction N) will be used. If a name has been defined for FILE but none for the segment, ,FILE,- can be omitted.

If the file FILE already contains data, they will be overwritten.

a) to a segment file

U,FILE,SEGMENT Copies the EDITOR file to the segment file FILE under the segment name SEGMENT.

If FILE or ,FILE,SEGMENT is omitted, the name defined for each of them (cf. name instruction N) will be used instead.

If the segment file already contains a segment having the name specified here, this instruction will be rejected. If an already existing segment is to be replaced, the following instruction must be used.

U!,FILE,SEGMENT Copies the EDITOR file to the segment file FILE under the segment name SEGMENT.

If FILE or ,FILE,SEGMENT is omitted, the names defined for each of them (cf. name instruction N) will used instead.

If the segment file already contains a segment having the name specified here, this segment will be replaced.

If the EDITOR file is empty, the user will be asked if the entire segment in the segment file should be deleted. The following responses may be given:

Y The segment will be deleted and the name of the segment will be struck from the table of contents.
N The segment will be deleted (!), but the name of the segment will remain in the table of contents.
new instruction: The segment will remain in the segment file unchanged.

Deleting a segment

If a segment in a segment file is to be deleted with the Editor, the Editor file must be empty. The empty Editor file can then be "saved" to this segment using the Unload instruction U!,FILE,SEGMENT described above. In this case, the Editor will first ask whether the entire segment in the segment file should be deleted, which is answered as described above: N to delete only the contents of the segment, J to strike the name of the segment from the table of contents for the segment file.

Inquiring or switching the name of the EDITOR file/

v Shows the name of the file presently being processed with the Editor.

v,FILE Changes the file to be processed with the Editor.

After changing the EDITOR file, the names defined for FILE and SEGMENT (cf. name instruction N) will be automatically replaced by the names last defined when the new EDITOR file was previously edited.

v,-STD- The standard EDITOR file is to be edited.

Listing/setting the mode:

a) Numbering and character display

A complete mode specification consists of three subvalues

The first subvalue can be either "+" or "-". It specifies whether the records displayed on screen (e.g. after a Show instruction) are to be shown with (+) or without (-) their record numbers. One exception here are the "search instructions for structured data", which are always displayed without record numbers regardless of the setting specified in the first subvalue. Please note that records without record numbers cannot be corrected.

The second subvalue can be either "T" or "P". It specifies whether the records are to be numbered in text mode (with page number) or in program mode (without page number).

The third subvalue can be either "+" or "-". It specifies whether accent letters and characters encoded with the control character "#" are to be displayed on screen as such (+) or in their encoded (input) form (-). For example, whether an "e" with an accent grave should be displayed as è (+) or as %\e (-). However, this setting is only effective when a code table supporting these characters has been previously set with the command #DEFINE (see page 83).

M Lists the presently set mode

M,... Mode ... is set. A three-digit mode specification must be entered for "...", as described above for a complete mode specification. It is also permissible to enter just the first subvalue, or the first two subvalues of this mode specification.

M,T Set mode T: record numbering in text mode.

M,P Set mode P: record numbering in program mode.

Note: If the switch from mode T to mode P is not possible because the record numbers are too large, the records can also be renumbered in program mode with the instruction R,,0.0/1;/1 (for more than 9999 records use: R,,0.0/1;/01 - for more than 99999 records use: R,,0.0/1;/001).

b) Text windows

The Editor displays data in a single text window (full screen) unless one of the next two instructions are given for splitting the screen into an upper and lower window. The two screens can be used to edit the same file or two different files (by changing the file to be edited with the file instruction V).

M,1 Split screen, upper text window active.

M,2 Split screen, lower text window active.

M,0 Use only one text window (full screen display)

Due to the limitation imposed by individual operating systems, only one text window is possible under BS2000, MVS and VM/CMS.

Listing/setting the tabulator:

After TUSTEP has been initialized, the default tab stops are set at positions 11, 21, 31, 41, 51, 61, 71.

T Lists the tabulator character and the tab stops.

T,X N1 N2 N3 ... Defines character X as the tabulator character. X may be any character (except digits) chosen by the user. Positions N1 N2 N3 etc. are defined as tab stops.

A tabulator character defined in this way is only effective when data are entered after the instructions IE and I,POS#N. Blanks will be inserted from the position of the tabulator character to the next tabulator stop (this corresponds to the tab key on a typewriter). If no tab stop follows, the tabulator character will be replaced by a *single* blank space.

The defined tab stops can also be jumped to with the control command TAB (cf. page 218). If this is the only use to be made of tabulators, the character X (but not the comma) may be omitted in this instruction.

T! Deletes tabulator character and tab stops

Tabulator characters and tab stops can also be defined when the Editor is called up (see command #EDIT page 91). If the tabulator character and/or tab stops are to be automatically defined upon TUSTEP initialization, the appropriate Editor call can be written in the start file (see page 50).

Function keys: listing, defining, deleting and activating function keys:

In the Editor, a function is an abbreviated instruction having the form FN (N = any number from 1 to 40). It designates an instruction which can be defined by the user. Such an abbreviation is called a function because the instruction for which it stands can be executed by pressing the corresponding function key.

The following functions have been predefined by TUSTEP:

F1=SB	F7=ST,*
F2=SE	F8=SB,*
F3=B	F9=G-
F4=SA,*	F10=G+
F5=S	F11=M,1
F6=E #EXECUTE,<EDITOR>	F12=M,2

However, these may be redefined by the user.

FN Activates function N.

If the keyboard being used is equipped with function keys, a function can also be activated by pressing the respective function key. If an instruction is already present on the screen, it will be executed before the function.

FN=EDITOR INSTRUCTION Defines function N.

Up to 40 different functions may be defined. They are numbered from 1 to 40, with the respective number being given for N. At present, only one instruction may be given for INSTRUCTION. This instruction is executed each time its assigned function is activated. Each function retains its definition until it is either deleted or redefined.

FN= Deletes function N.

F Lists all functions and their definitions.

If all functions cannot be listed on a single screen, pressing the return key will display the next portion of the list.

Functions can also be defined when the Editor is called up (cf. #EDIT command, page 91). If functions are to be automatically defined upon TUSTEP initialization, the appropriate Editor call can be written in the start file (see page 50).

Macros: executing/defining/deleting/listing

Due to limitations imposed by some operating systems, Editor macros can only be executed under DOS, UNIX and VMS.

Execute macro NAME.

Executing an Editor macro is initiated by pressing a certain key twice. The appropriate key depends on the operating system and keyboard being used (cf. keyboard tables starting on page 230). Then the name of the macro is entered and sent to the computer. A quick key combination can be used for macros whose names consist of a single letter. Here a macro is initiated by pressing the appropriate first key of a quick key combination, followed by the desired letter of the macro - with no subsequent pressing of the ENTER key to send this to the computer. In some cases (e.g. under DOS) a quick key macro can also be executed by entering the letter of the macro while the "ALT" key" is held down.

When calling up the Editor (command #EDIT), the name of a macro can be given for the specification MACRO. This macro is then immediately executed after the Editor has been started.

Y,NAME=MACRO INSTRUCTION Defines the macro NAME.

A macro instruction is either a control command or a character string enclosed by a delimiter of the user's choice (special character except for a comma). A description of available control codes and their action starts on page 218. A character string will be displayed on screen starting at the current cursor position as if it were entered there manually. Individual macro instructions must be separated from each other by a comma. Each macro instruction can be preceded by number (which is separated from the instruction by a "*"), which specifies how many times the macro is to be executed.

Y,NAME= Deletes macro NAME.

Y! Deletes all macros.

Y Lists all macros and their definitions.

If not all macros can fit into a single screen, the next screen portion can be shown by pressing the RETURN key.

Macros can also be defined when the Editor is called up (see command #EDIT page 91). If macros are to be automatically defined upon TUSTEP initialization, the appropriate Editor call can be written in the start file (see page 50).

Character groups and string groups: defining, erasing and listing

A character group or a string group is a group of individual characters or strings which can be used in subsequent instructions within character string comparison tables CSCT (cf. page 210), character string search tables CSST (cf, page 208) and character string pair tables CSPT (cf. page 210).

For defining character groups and string groups and for refering to groups so defined, the group identifications ›N and ‹N are available, where each N is to be replaced by a digit. Thus, a maximum of 20 groups each can be defined simultaneously. Each group definition is valid for all subsequent instructions until it is redefined or deleted.

If the definition of a character group or a string group contains a group identification, this identification will be interpreted as an identification for the corresponding character group (even if a string group has been defined with the same identification). When used in all other instructions, a group identification will be interpreted as the identification for the corresponding string group, if a string group has already been defined with this identification. Otherwise it will be interpreted as the identification for the corresponding character group.

XN**C**=CHARACTER Defines the character group XN. Either "›" or "‹" must be given for X; the corresponding digit must be given for N (e.g. ›2Z=AEIOU for defining a character group ›2 which is to contain the vowels a, e, i, o, u).

For a complete description of how to define character groups, consult the section entitled "Parameter type V" (starting on page 247).

XN**C**= Deletes the character group XN. "›" or "‹" must be given for X; the corresponding digit must be given for N.

XN**S**=STRINGS Defines the string group XN. "›" or "‹" must be given for X; the corresponding digit must be given for N (e.g. ›2S='%/'%\'%<'%:'%;' for defining the string group ›2, which is to contain accent marks occurring in French texts).

For a complete description of how to define strings groups, consult the section entitled "Parameter type V" (starting on page 247).

XN**S**= Deletes the string group XN. "›" oder "‹" must be given for X, the corresponding digit must be given for N.

W Lists all defined character groups and string groups ("wild card")

If all character groups and string groups cannot be displayed on the screen, pressing the return key will display the the next portion.

Character groups and string groups can also be defined when the editor is started (cf. #EDIT command page 91).

Executing TUSTEP commands:

The Editor can only execute editor instructions and Editor macros. However, TUSTEP commands can be executed from the Editor with the following Editor instruction. It exits the Editor automatically and restarts the Editor after the command has been executed.

E,COMM Executes the command COMM.

The command COMM must be delimited at its beginning and end by a delimiter character of the user's choice. If more than one command is to be executed, the commands must be separated from each other by this delimiter. If a command consists of more than one line (e.g. for commands with parameters), each line must also be separated from one another with this delimiter character. When choosing a delimiter character, make sure that it does not occur in the command itself.

Note: commands given here must always be given in their complete form. When using commands followed by data (e.g. parameters), this data must also be given, including the final *EOF.

If the character string <EDITOR> occurs is specified in the command COMM, this character string (including the pointed brackets) will be replaced by the name of the EDITOR file. Similarly, the character string <FILE> will be replaced by the defined file name and the character string will be replaced by the defined segment name. The names for file and segment can be defined by using the name instruction N (cf. page 186).

Displaying and repeating previous instructions:

The instructions described below can be used to display previously used instructions on the screen. These instructions (except for those displayed with the command **GG**) can be modified or left unchanged before they are executed again.

The buffer will contain the 20 most recently used instructions which
- have the correct syntax,
- consist of more characters than the instruction itself, and
- differ from the previously given instruction.

These stored instructions can be displayed on the screen with the following instructions:

GG Displays the 20 most recently stored instructions

G Displays the most recently stored instruction

G- Displays the instruction stored prior to the last instruction which has been listed with **G**, **G-** or **G+**. If none of the instructions **G**,**G-** or **G+** have been given since the last instruction (except of course the instructions **G**, **G-** und **G+**), this will list the most recently stored instruction.

G+ Displays the instruction stored after the last instruction which has been listed with **G**, **G-** or **G+**. If none of the instructions **G**,**G-** or **G+** have been given since the last instruction (except of course the instructions **G**, **G-** and **G+**), this will list the most recently stored instruction.

The instructions **G+** and **G-** can thus be used to scroll forwards and backwards through the list of stored instructions. It is recommended to assign each of these instructions to a function key.

An instruction listed with **G-** or **G+** is ready to be executed once again. However, if it is not to be executed and the user merely wishes to scroll through the list of instructions using **G-** oder **G+**, the instruction just displayed must be either overwritten or deleted before the next instruction can be shown. This can be avoided by defining the instructions **G-** and **G+** as functions (see page 191) and assigning them to two function keys. Please note that only a single function can be defined for each of the two instructions. In the standard TUSTEP setup, the functions F9 and F10 have been defined as **G-** and **G+**, respectively. If a different function is to be defined with **G-** or **G+**, the functions F9 or F10 must first be redefined. If the instruction **G-** or **G+** have been assigned to a function key, pressing this function key will not execute the instruction listed on screen (as opposed to all other function keys), but the instruction that has been assigned to the function key (i.e. the instruction G- or G+). This spares the user the trouble of erasing the instruction last displayed with **G**, **G-** or **G+**.

In addition to the 20 most recent instructions, the most recent organizational instruction (except L) and the most recent extended instruction which start with the same letter are also stored. These stored instructions can be displayed again on the screen with the following instructions, where any modifications can be made before they are executed once again.

GE Displays the most recent organizational or extended instruction which starts with the letter E. The corresponding letter is to be given for E.

If an instruction is to be executed which has been defined as a function in a similar form, it can be displayed on screen with the following instruction, where it can be modified and then executed:

GN Displays the instruction which has been defined as function **F**N. The number of the corresponding function must be given for N.

If the definition of a function, character group, string group or parameter is to be changed, and has already been defined in a similar form, it can be displayed on screen with the following commands, where it can then be modified.

GFN Displays the definition of function **F**N. The number of the corresponding function must be given for N (e.g. GF2).

GX**N****C** Displays the definition of the character group XN. Either "›" or "‹" must be given for X, and the corresponding digit must be given for N. (z.B. G›2C).

GX**N****S** Displays the definition of the string group XN. Either "›" or "‹" must be given for X, and the corresponding digit must be given for N. (e.g. G›2S).

GPN,XXX Displays the definition of the parameter **P**N,XXX. The corresponding digit must be given for N, and the corresponding identification must be given for XXX. (e.g. GP2,ZV3).

Extended instructions:

Searching and showing

SB,RAN,LIMI,CSST Searches for the records of a range which fulfill the conditions LIMI,CSST and shows the data beginning with each record found.

SA,RAN,LIMI,CSST Searches for the records of a range which fulfill the conditions LIMI,CSST and shows the data around each record found.

ST,RAN,LIMI,CSST Searches for the records of a range which fulfill the conditions LIMI,CSST and shows the data up to each record found.

SO,RAN,LIMI,CSST Searches for the records of a range which fulfill the conditions LIMI,CSST and shows only the records found.

S,RAN,LIMI,CSST Searches for the records of a range which fulfill the conditions LIMI,CSST and shows each record according to the most recent of the four instruction described above which has been given. If none of these instructions has been given yet, **S**,RAN,LIMI,CSST has the same effect as **SA**,RAN,LIMI,CSST.

After giving these Show instructions, the following responses are possible:

return key: continues searching and showing in the same direction.

W Continue search. Do not display found records but only show the number of found character strings as well as the number of records involved when the search is completed.

SR Continues reverse searching and showing (i.e. towards the beginning of the file).

SF Continues forward searching and showing (i.e. towards the end of the file).

+ Interrupts search and scrolls forward; continued pressing of the return key results in a scroll towards the end of the file (not possible after **SO**).

- Interrupts search and scrolls backward; continued pressing of the return key results in a scroll towards the beginning of the file (not possible after **SO**).

new instruction: Interrupts the search. In this case, as is true for the entries **+** and **-**, the instruction **S** (or one of the instructions **SR** and **SF**) may be used to continue searching and showing from the position of the interruption, provided that the editor has not been ended or that a new file has not been edited in the meantime using the instruction V.

If in these Show instructions the range RAN is given in the form (POS,POS), the first position may be higher than the second; in this case, the given range is searched

backwards. If the range RAN is given in the form POS, it is searched forward beginning with this record position.

Inserting:

II,TEXT,RAN:COL,LIMI,CSST Inserts the character string TEXT in every record of the range RAN which fulfills the conditions LIMI,CSST.

The character string TEXT must be delimited at its beginning and its end by a delimiter of the user's choice. This delimiter character must not occur in the character string itself. Coding of the character string TEXT follows the same rules which apply to data (cf. "Character codes", page 176).

:COL may be omitted. In this case, the character string TEXT will be inserted at the end of each record. If the value N is given for COL, the character string will be inserted starting at the position N of each record (i.e. the characters from position N up to the end of the record are moved to the right). If +N is given, the character string is inserted after the character position N (counting the characters from the beginning of the record). If -N is given, the characters are counted from the end of the record and the character string is inserted in front of the respective character position N.

If the value N1-N2 is given for COL, the character string in the columns N1 to N2 is replaced by the character string TEXT. The characters from column N2 up to the end of the record are moved to the left if the character string TEXT is shorter than the character string it has replaced, or to the right if it is longer. If only the character string in column N is to be replaced, the value N-N must be given. (Giving N only would insert the character string TEXT at position N instead of replacing the character N.)

IB,RAN1,RAN2,LIMI,CSST Inserts the records of the range RAN1 before each record of the ran RAN2 which fulfills the conditions LIMI,CSST.

IA,RAN1,RAN2,LIMI,CSST Inserts the records of the range RAN1 after each record of the ran RAN2 which fulfills the conditions LIMI,CSST.

When these instructions are executed, each record which fulfills the conditions will be displayed. At the same time, the user is asked whether insertion should be carried out. One of the following responses may be given:

return key: The insertion is executed and the next record which fulfills the conditions is searched.
Y The record is inserted and insertion is interrupted.
N The record is not inserted, but the next record which fulfills the conditions is searched.

C The record is inserted and all following records which fulfill the conditions are inserted with no further user consultation.
new instruction: Insertion is interrupted, with no insertion ocurring in/before/after the current record. In this case, as is true for the response **Y**, the instruction **I** may be used later to continue insertion starting at the position of interruption, provided that the editor has not been ended or that a new file has not been edited in the meantime.

Deleting:

D!,RAN,LIMI,CSST Deletes specific records of a range.

Each record that fulfills the conditions LIMI,CSST will be displayed. At the same time, the user will be asked whether the record should be deleted. One of the following responses may be given:

return key: The record is deleted and the next one is searched.
Y The record is deleted and deletion is interrupted.
N The record is not deleted, but the next one is searched.
C The record is deleted and all following records which fulfill the conditions are deleted with no further user consultation.
new instruction: Deletion is interrupted; the current record is not deleted. In this case, as is true for the response **Y**, the instruction **D** may be used later to continue deletion starting at the position of interruption, provided that the editor has not been ended or that a new file has not been edited in the meantime using the instruction V.

Copying:

C,RAN,POS,LIMI,CSST Copies specific records of a range.

C,FILE,RAN,POS,LIMI,CSST Copying specific records from a range of another file.

C,DATEI,SEGMENT,POS,LIMI,CSST Copies specific records from a segment of another file.

Every record fulfilling the conditions LIMI,CSST will be displayed. At the same time, the user will be asked whether the record should be copied. One of the following responses may be given:

return key: The record is copied and the next record is searched.
Y The record is copied and copying is interrupted.
N The record is not copied, but the next record is searched.

C The record is copied and all following records which fulfill the conditions are copied with no further user consultation.

new instruction: Copying is interrupted, without copying the current record. In this case, as is true for the response **Y**, the instruction **C** may be used later to continue copying starting at the position of interruption, provided that the editor has not been ended or that a new file has not been edited in the meantime using the instruction V.

The records will be copied into the editor file starting with record number POS (if a record with the record number POS does not yet exist), or after the record number POS (if there is already a record with record number POS). If the records are to be copied to the end of the file, POS may be omitted. In text mode, the first new record copied at the end of the file is given the number n.1, where n is one page number greater than that of the file's last record.

Renumbering (moving) records

R,RAN,POS,LIMI,CSST Renumbers (moves) specific records of a range).

Every record which fulfills the conditions LIMI,CSST will be displayed. At the same time, the user will be asked whether the record should be moved. One of the following responses can be given:

return key: The record is moved and the next record is searched.

Y The record is moved and moving is interrupted.

N The record is not moved, but the next record is searched.

C The record is moved and all following records which fulfill the conditions are moved with no further user consultation.

new instruction: Moving is interrupted and the current record is not moved. In this case, as is true for the response **Y**, the instruction **R** may be used later to continue moving starting at the position of interruption, provided that the editor has not been ended or that a new file has not been edited in the meantime using the instruction V.

The records will be moved and renumbered so that they will start with record number POS (if a record with the record number POS does not yet exist), or after the record number POS (if there is already a record with record number POS). If the records are to be moved to the end of the file, POS may be omitted. In text mode, the first new record moved to the end of the file is given the number n.1, where n is one page number greater than that of the file's last record.

Exchanging (replacing) character strings:

X,RAN,LIMI,CSPT Exchanges the character strings CSPT in a range.

Before a character string is replaced, the record will be displayed in both its original version and its version after replacement. At the same time, the user is asked whether replacement should be carried out. One of the following responses can be given:

return key: The character string is replaced and the next record is searched.

Y The character string is replaced and replacement is then interrupted.

N The character is not replaced, but the next record to be replaced is searched.

C The character string is replaced and all following records are replaced with no further user consultation.

YF The character string is replaced. Any other character strings in the rest of the actual record will not be replaced, and the search for further character strings to be replaced will continue with the following record.

NF The character string is not replaced, but the search for further character strings to be replaced will continue in the following record.

CF The character string is replaced and all other character strings in the current record will be replaced with no further user consultation. User consultation is resumed starting with the next record containing character strings to be replaced.

new instruction: Replacement is interrupted and the character string in the current record is not replaced. In this case, as is true for the response **Y**, the instruction **X** may be used later to continue replacing starting at the position of interruption, provided that the editor has not been ended or that a new file has not been edited in the meantime with the instruction V.

Search instructions for structured data

With the help of the extended instructions Search and Show (cf. page 197), records which contain certain character strings can be searched. The records thus found are displayed either in their respective context (i.e. surrounding records, instruction SA), or only the found records are displayed (instruction SO). Since the records are displayed with their record numbers and unaltered text, any necessary corrections can be made.

When employing the search instructions described in this chapter (which correspond to database queries), the search conditions and the display format must first be defined with the appropriate instructions. Search conditions and display format can be defined by a combination of parameters.

For search condition the following can be defined:
- which records are to form a logical entity (e.g. a bibliographical entry), which will be referred to in the following as a text unit,
- which text parts of a certain text unit must contain a given character string in order for the search condition to be fulfilled, and
- which character strings (e.g. accent marks) are to be eliminated in these text parts or replaced by other character strings prior to checking whether a given character string is present.

For the show format, the following can be defined:
- which text parts of the text unit are to be shown,
- which character strings (e.g. heading markers) in these text parts are to be eliminated or replaced by others,
- at what points (e.g. for every heading) is a new line to be started for screen output.

If a text unit fulfills the search conditions, it will be displayed on screen with no record number (and no surrounding text) in accordance with the given show format. The data shown cannot be corrected in the show format. If corrections are to be made in the data, they must be called up with the respective editor instruction (e.g. the instruction SB,*).

All paramaters which belong to a search condition and a show format comprise a parameter group. Up to 9 parameter groups may be defined at the same time. They are named P1 to P9.

Defining and deleting parameters: general instructions

An instruction for defining a parameter has the syntax:

PN,XXX=CSCT|CSST|CSPT

With this instruction, the parameter with the identification xxx is defined for the parameter group PN. The corresponding digit is to be given for N and the corresponding parameter identification is to be given for xxx. Depending on the parameter being defined, either a character string comparison

table CSCT (see page 210), a character string search table CSST (see page 208) or a character string pair table CSPT (see page 210) must be given in place of CSCT|CSCT|CSPT.

The definition of a parameter remains in effect until it is replaced by a new definition or deleted. An instruction for deleting parameters has the syntax:

PN,XXX=

With this instruction, the parameter with the identification XXX will be deleted from the parameter group PN. This instruction differs from the definition instruction only in that CSCT|CSST|CSPT| is omitted after the equals sign.

Defining parameters for the search condition

Organizing records into a text unit

In case each input record already contains a complete text unit, the following two parameters should not be defined. Otherwise, the parameters AA and/or AE can be used to organize more records into a text unit.

PN,**AA**=CSCT Character strings placed at the beginning of a record which mark the start of a text unit.

PN,**AE**=CSCT Character strigs placed at the end of a record which mark the end of a text unit.

Selecting text units using search conditions:

A search condition consisting of up to 9 partial conditions may be defined for each parameter group. For each text part, a text part, which is marked by beginning and end markers, can first be selected from the text unit. After making any necessary character string replacements in this text part, it can then be checked to see if it contains one of the given character strings. If such a character string is found, the partial condition is fulfilled; otherwise, it is not fulfilled.

In order for a search condition to be fulfilled (only in this case will the text unit be shown), all partial conditions must be fulfilled. This corresponds to a logical AND. A present, a connection with a logical OR can only be made by giving more than one character string for a partial condition. Additional logical connections are not yet possible.

PN,**AVM**=CSST Character strings marking the beginning of the text part for the partial condition M (M = 1 to 9). If the text unit does not contain any of the given character strings, no text part will be selected; the partial condition is not fulfilled in this case. If this parameter is not defined, the text part starts at the beginning of the text unit.

PN,EVM=CSST Character strings marking the end of the text part for the partial condition M (M = 1 to 9). If the text unit (or that part of the text unit starting with the beginning marker for each text part) does not contain any of the given character strings, or if this parameter is not defined, the text part ends at the end of the text unit.

PN,XVM=CSPT Pairs of character strings (and exception strings). The first character string of a pair will be replaced by the pair's second character string in the text part for the partial condition M (M = 1 to 9).

PN,ZVM=CSST Character strings, of which at least one must occur in the (modified) text part for the partial condition M (M = 1 to 9).

Defining parameters for the show format

Selecting text parts for screen display:

If the complete text unit is to be shown, neither of the first two parameters below need to be defined. Otherwise, the parameters AZM and EZM (M = 1 to 9) can be used to select up to 9 text parts from the text unit which are to be displayed on the screen. The sequence of these text parts will correspond to their number (M). Gaps may be left in the numbering. The parameter XZM can be used to replace character strings in each of the text parts before they are displayed.

PN,AZM=CSST Character strings marking the beginning of the text part M (M = 1 to 9). If the text unit does not contain any of the given character strings, nothing will be selected for the text part. If this parameter is not defined, the text part starts at the beginning of the text unit, provided that the corresponding parameter for the end marker has been defined.

PN,EZM=CSST Character strings marking the end of the text part M (M = 1 to 9). If the text unit (or that part of the text unit starting with the beginning marker for the same text part) does not contain any of the given character strings, or if this parameter is not defined, the text part ends at the end of the text unit.

PN,XZM=CSPT Pairs of character strings (and exception strings). The first character string of a pair will be replaced by the pair's second character string in the text part M (M = 1 to 9).

Line division for screen display:

The following three parameters can be used to indicate where a new line is to be started for the screen display.

PN,ZA=CSST Character strings, before which a new line is to be started (beginning of line).

PN,ZE=CSST Character strings, after which a new line is to be started (end of line).

PN,ZW=CSST Character strings (not displayed) used to start a new line.

Searching and showing

After the parameters have been defined, the actual search can be carried out. Each of the following search instructions uses the current parameter group. The current parameter group is the one for which a parameter has been last defined or deleted, or the one whose parameters have been last shown with one of the instructions **P** or **P**N (N = 1 bis 9).

Q,RAN Searches the specified range for text units which fulfill the search condition of the current parameter group.

If the range RAN is given in the form (POS,POS), the first position may be greater than the second; in this case a reverse search is conducted in the given range.

Q,POS Search starts at the specified record position for text units which fulfill the search condition of the current parameter group.

QB Searches from the beginning of the file for text units which fulfill the search condition of the current parameter group.

QE Searches from the end of the file for text units which fulfill the search condition of the current parameter group.

After giving these search instructions, the following entries are possible:

return key: continues searching in the same direction.

W Continue search without displaying found text units; the number of found text units are to be displayed after the search is completed.

QR continue search backwards (i.e. towards the beginning of the file).

QF continue search forwards (i.e. toward the end of the file).

new instruction: interrupts the search. In this case, the instruction **Q** (or one of the instructions **QR** und **QF**) can be given later to continue the search from the point of interruption, provided that the Editor has not been ended or that a new file has not been edited in the meantime using the instruction V.

Listing, deleting and copying parameters

P displays on screen the parameters for each parameter group, starting with the current parameter group. This instruction can thus be used to determine which parameter group is presently current.

If the parameters for a single parameter group cannot fit into a single screen, the next portion of parameters can be displayed by pressing the return key. Otherwise, pressing the return key will display the parameters for the next parameter group.

PN displays on screen the parameters of the parameter group N. The corresponding group number is to be given for N.

If there is not enough room on the screen for all parameters, the next portion can be displayed by pressing the return key.

PN= deletes all parameters of the parameter group N.

For a description of how to delete individual parameters of a parameter group, see page 203 above).

PN1=**P**N2 deletes all parameters of parameter group N1 and then copies all parameters of parameter group N2 to parameter group N1.

Note:

Parameters can also be defined when the Editor is started. (see command #EDIT, page 91). If parameters are to be defined automatically upon TUSTEP initialization, the appropriate Editor call can be written in the start file (see page 50).

Syntax used for defining a file range RAN or a record position POS

a) Program mode:

RAN: POS | (POS,POS) | (POS#N) | ([C]/)

POS: REF | REF+N | REF-N | +N | -N

REF: L | [L]/D | *

b) Text mode:

RAN: POS | (POS,POS) | (POS#N) | ([P].) | ([[P.]L]/)

POS: REF | REF+N | REF-N | +N | -N

REF: [P.]L | [[P.]L]/D | *

The abbreviations and characters (in both program mode and text mode) have the following meanings: * record position of the most recently edited record

REF+N Nth record after record position REF

REF-N Nth record before record position REF

+N Nth record from the beginning of the file

-N Nth record from the end of the file

(POS1,POS2) Range of a file starting with record position POS1 and ending with record position POS2

(POS#N) Range of a file starting with record position POS and containing a total of N records

L Record having the line number L and the distinction number zero

L/D Record having the line number L and the distinction number D

(L/) Range of a file containing all records having the line number L, regardless of the distinction number

P.L Record having the page number P, line number L and distinction number zero

P.L/D Record having the page number P, line number L and distinction number D

(P.) Range of a file containing all records having the page number P, regardless of the line number L and the distinction number D

(P.L/) Range of a file containing all records having the page number P and the line number L, regardless of the distinction number D

The specification RAN can be omitted if the range to be edited contains all the records of the file.

Record definitions in square brackets [ ] are optional. In this case, any omitted value is substitued by the corresponding value in the record position * (i.e the current record). Exception: the second position value in the specification RAN when given in the form (POS,POS); here the first record position is used as a substitute for any omitted values.

The distinction number must always be written with leading zeros, whereas trailing zeros may be omitted (2/3 is equivalent to 2/30).

Limiting action to certain columns LIMI:

LIMI may be omitted. If so, the search for the character string given for CSST and the replacement of character strings given for CSPT will be carried out for the entire record. If the search or replacement action is to be carried out only for certain columns of each record, the following specifications can be given for LIMI:

n1-n2	from column n1 to column n2
n1+n2	n2 columns starting at column n1
+n	in the first n columns
-n	in the last n columns
n	in column n only

Character string search table CSST:

Character strings to be searched for are given here, as well as any character strings that are to be skipped during a a search. If character strings of different lengths are specified, the longer character string is given priority. For character strings of equal length, priority is assigned to the order in which they were entered.

Character strings must be separated from each other by a delimiter of the user's choice. The delimiter is the first character of the character string CSST. A double limiter separates character strings which are to be searched from those which are to be ignored during a search. This double delimiter can be employed as often as desired to switch between search strings and exception strings. The last character of the character string search table CSST must again be a delimiter.

If a character string contains letters, both uppercase and lowercase forms will be searched for. The distinction between uppercase and lowercase letters can be made by inserting "<" or

">" before the each letter. Thus, "a" and "A" will be interpreted as either an uppercase or lowercase a, whereas "<a" and "<A" will be interpreted as an uppercase a; ">a" and ">A" will be interpreted as a lowercase a. To represent the characters "<" and ">" themselves, they must be given as "<<" and ">>".

In addition to single characters, a character string may also contain one of the following identifications for character groups. These are used to represent groups of characters.

>*	all lowercase letters of the 7-bit and 8-bit TUSTEP character set
<*	all uppercase letters of the 7-bit and 8-bit TUSTEP character set
>/	all digits of the 7-bit and 8-bit TUSTEP character set
</	all letters of the 7-bit- and 8-bit TUSTEP character set
>%	all characters of the 7-bit TUSTEP character set
<%	all characters of the 7-bit and 8-bit TUSTEP character set

In addition to these predefined group identifications for character groups, the group identification (›N or ‹N, N = digit) of a user-defined character group or string group (cf. page 127) may also be given instead of a character. If both a character group and a string group have been defined for a group identification, the string group has priority.

A frequency condition may be given in a character string which is effective for the character which directly follows it:

><n	the character must occur at least n times
<>n	the character may occur up to n times
><0	te character may be missing
<>0	the character may occur any number of times

In this case, an integer from 1 to 9 may be given for "n". For "><0" and "<>0", the zero may be omitted if no digit follows. Thus, "<>0>/" has the same effect as "<>>/" and stands for "any number of digits". The frequency conditions "><n" and "><0" may be combined with "<>n" and "<>0". For example, "x><<> y" means that between x and y the blank may be missing, or that any number of blanks may be located between x and z.

In addition to frequency conditions, references and surroundings conditions can also be given. For a complete description of "CSST" specifications for group identifications, consult the chapter entitled "Parameters" (starting on page 241 ff.); "CSST" corresponds to the character string search table described for parameter type IX.

Examples:
1. The character string "xyz" is to be searched and each record in which it occurs is to be shown with its surrounding records: SA,,,-XYZ-
2. Every record with a # in column 1 is to be shown: SO,,1,/#/

3. The character string "und" is to be searched, but the character strings "round" and "undefined" are to be ignored in the search: SO,,,/und//round/undefined//

Character string comparison table CSCT:

Data given for a character string comparison table are specified with the same syntax used for a character string search table. However, frequency conditions and surroundings conditions are not allowed.

For a complete description of the specifications possible for a character string comparison table, consult the chapter entitled "Parameters" (starting on page 241); "CSCT" corresponds to the character string comparison table described for parameter type VIII.

Character string pairs CSPT:

Here the user specifies character strings in pairs; the first character string is the one to be searched for and then replaced by the pair's second character string, the character substitution string. In addition, any character strings that are to be skipped during replacement may be given as (exception character strings).

Character search strings are described in the section "Character string search table CSST". The characters of the substitution character string are inserted in the text as upper and lower case letters as entered.

Search and replacement character strings may be of different length. They are separated from each other by a delimiter of the user's choice. The delimiter is the first character of the character string pair table CSPT. Two successive delimiters are used to separate character string pairs from exception strings. The last character of a character string pair table CSPT must also be a delimiter.

For a complete description of the permissible specifications for "CSPT" please consult the chapter "Parameters" (starting on page 241); "CSPT" corresponds to the character string pair table described for parameter type X.

Examples:
1. The character string "xyz" is to be replaced by "yy": R,,,;xyz;yy;
2. "x" is to be deleted, and "y" is to be replaced by "x" unless there is a blank after y: R,,,/x//y/x//y //

Adjusting the Editor to screen display (options)

The Editor's default color/gray level settings have been configured in such a way to be compatible with as many screen types as possible. Yet due to the diversity of screen types and their settings, the display on some screens may be unsatisfactory or even unusable. For this reason, TUSTEP provides the user with the option of choosing his own settings for the monitor's colors / gray levels.

The Editor assumes a screen size of 24 lines à 80 characters per line, a typical standard for many screens in use today. To fully exploit the possibilities of larger screens, TUSTEP also lets the user define his own screen settings.

The cursor size can also be altered by the user; different cursor sizes can be selected for the insert mode and replacement mode. In addition, the cursor can be eliminated in its blinking form (e.g. if blinking is undesired). Here the color/gray level setting should be adjusted to make sure that the cursor position remains visible.

These various settings (hereafter referred to as options) are described below. Each setting remains in effect until the TUSTEP session is ended with the command #RESET. However, the option instruction (see below) can be used to save these current settings in coded form to file. They can then be reactivated with the DEFINITION specification used in the command for calling up the editor. They then remain in effect until altered or the TUSTEP session is ended.

Automatic setting of options

If user options are to be automatically set whenever TUSTEP is initialized, the following specifications should be written in the start file (called *TUSTEP.INI):

```
#EDIT,DEFINITIONS=*
O=00501904 17 7E 1E 67 37 3E 17 7E 1E 67 6E 01 02 00 ...
*EOF
```

The above record with the options (beginning with "O=") is incomplete and only serves to demonstrate how options can be defined. This record is not entered by itself but instead written to the file with the following instruction:

o,POS Enters currently set options at the record position POS. If a record having the number POS already exists, the instruction will be rejected.

If further alterations have been made to these settings and they are to be used for subsequent TUSTEP sessions, the options must be updated in the file with the following instruction. This instruction must also be used instead of the one listed above if the option is to be written to a record position that already exists.

o!,POS Enters the currently set options at record position POS. Any existing record having the record number POS will be overwritten.

In addition to the settings already mentioned, options can also be used to specify whether
- the insert mode or the replace mode is to be set (cf. control commands TGL_INS, SET_INS and SET_REP on page 221).
- the delete mode or backspace mode is to be set (cf. control commands TGL_DEL and SET_DEL on page 223).

Other definitions besides those used for options may also be set when calling up the Editor (cf. command #EDIT Seite 91f).

If TUSTEP is to be used with a number of screens which require different display settings, it is usually not advisable to have options defined automatically with the start file. In this case, a separate Editor call and its defined options for each screen type can be saved in its own file, or even better, in its own segment of a segment file. After TUSTEP is started, the desired file or segment file can be executed with the command #EXECUTE (see page 97). This procedure can also be used to continue a TUSTEP session at another monitor which requires different settings. But if the new settings vary only to a "minimal" degree, it may be easier to set them manually after starting the Editor.

Setting attributes for colors / gray levels

The key combination CTRL+F (press the "f" key while holding down the CTRL key) activates the screen menu for setting screen colors / gray levels. The screen displayed will be similar to that shown below.

```
Attr data         17 normal  7E cursor  1E emphasized  67 marked
Attr message line 37 normal             3E emphasized
Attr command line 17 normal  7E cursor  1E emphasized
Attr status line  67 normal             6E emphasized

     Select field to change with cursor up/down/left/right
     Select color by entering a two-character code given below
     Exit with ENTER

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
```

The cursor is initally located in the first entry field in the top line of the screen. The lower portion of the screen features 256 character pairs, which in the most favorable constellation (e.g. on an IBM-compatible PC with color display) will each display a different attribute (color and contrasting background and foreground). The two-character code representing the desired attributes can be entered in the entry field in the upper part of the screen (these are the fields which already contain such codes).

The cursor keys are used to move between entry fields (either forwards or backwards). Once a new code is entered in a field, the accompanying text will display the new setting. After making all desired alterations, the new settings must be confirmed by pressing the ENTER key. Then the previous screen display will appear. However, the new settings are not activated until a new screen is displayed (e.g. after a show instruction).

The attributes for the various displays used in the Editor can be set in the following entry fields:

Attr data = appearance of text on screen

- normal: for normal text (for display and input)
- cursor: for the character at the cursor position
- emphasized: for representing emphasized text (e.g. found character strings)
- marked: for marked text

Attr message line = message display

- normal: for questions and prompts
- emphasized: for error messages

Attr command line = display of entered instructions

- normal: for entry prompt
- cursor: for the character at the cursor position
- emphasized: for entered characters

Attr status line = status line display

- normal: for regular displays
- emphasized: for warnings

Caution:

The settings should be selected so that the words "normal", "cursor", "emphasized" and "marked" in the first four lines remain visible at all points of the screen in which they occur in the above illustration. Otherwise, working with the Editor will be practically impossible, since certain data or error messages cannot be seen on screen.

For this reason the default values should be first used when working with the Editor at a certain monitor for the first time. Afterwards, settings may be changed if so desired.

In some cases, pressing CTRL+F results in distorted (or blank) commentary and entry fields. Here a legible character pair in the lower half of the screen should be entered without previously moving the cursor with any cursor keys. Confirm the entry by pressing the ENTER key and press CTRL+F once again.

Setting line length and number of lines per screen

The key combination CTRL+L (press the "l" key while holding down the CTRL key) activates the menu for line settings. The screen display will be similar to that shown below::

Width of standard screen (only 0 or 80 or 132): 0

Number of characters (80-160) per line: 80

Number of lines (24-60) of the screen: 25

Line width (40-160) for data input in mode P: 80

Line width (40-160) for data input in mode T: 80

Width of region for wrapping (0-160) in mode P: 30

Width of region for wrapping (0-160) in mode T: 30

Select value to change with cursor up/down
Define value by entering the number of characters/lines
Exit with ENTER

Lines in which a new setting is desired can be accessed by using the "cursor up" or "cursor down" keys. To change the setting, enter the desired number. If a line cannot be accessed because it is jumped over, this means that this value cannot be altered for the current monitor type. After making any desired changes, confirm choices by pressing the ENTER key. The previous screen display will then appear (assuming the new settings allow for this).

The following settings can be made:

Width of standard screen:

This setting applies only to monitors (e.g. VT100) that can be switched between an 80 and 132 characters/line display. The setting "0" means that such a switch should not be made while working with the Editor.

Number of characters / lines:

This setting determines the number of characters per line and the total number of lines to be shown on screen with the Editor. Caution: Specify no more characters or lines than can actually be displayed on the monitor. The Editor has no way of knowing if the values entered are too large. The results of specifying such values are unpredictable. Data may be lost when edited with an improper screen setting.

Line width for data input:

This setting limits (for both program mode and text mode) the number of characters that can be inserted into a single line. The upper limit for both settings is the line length specified for "Number of characters per line".

Width of region for wrapping:

This setting is used to specify the number of characters (for both program mode and text mode) within which a blank space is to be searched for whenever a record cannot fit into a single line onscreen. The search is conducted from the end of the record, with a continuation line being started at the found blank space. If no blank space is found within the specified wrap region, the line will be written to its fullest extent and a continuation line will be started. To show that the line break does not include a blank space in the data, an accent grave "`" will be added to the end of broken line.

Setting the size and speed of the cursor

The key combination CTRL+G (press the "g" key while holding down the CTRL key) activates the cursor control menu. The screen display will be similar to that shown below.

Cursor type for REPLACE mode: 1

Cursor type for INSERT mode: 2

Cursor speed (in both modes): 0

Select REPLACE/INSERT/SPEED with cursor up/down
Select cursor type/speed with cursor left/right
Exit with ENTER

The line for making the setting can be reached with the keys "cursor up" and "cursor down". Settings are changed by pressing the keys "cursor right" and "cursor left", which lists the possible settings in succession. After the desired settings have been made, they are confirmed by pressing the ENTER key. The previous screen display will then appear.

The following settings are possible:

Cursor type:

Determines the shape of the cursor for replace mode (overwrite) and for insert mode The resulting cursor size will be shown for each setting.

Cursor speed:

Determines cursor speed when a cursor key is pressed for a lengthier period of time. A zero means that no particular speed will be set. Otherwise, a larger number results in a faster cursor speed.

Limitation:

Although the possible settings are identical, only IBM compatibles will achieve the corresponding results. For terminals (VT100) and terminal emulations (e.g. Kermit, XTERM) the above settings are partially or completely ineffective.

Editor control commands

The Editor features various control commands for altering data on screen. Each of these control commands has been assigned an abbreviated form. The result of each control command is described below, followed by a list of tables for the most common keyboard types featuring the respective key or key combination assigned to each control command.

The effects of all control commands are limited to the screen display. Changes thus made on screen are not recorded to file until sent to the computer by pressing ENTER, CR or a function key.

Control commands can also be activated in Editor macros by using their abbreviations. Some control commands are reserved exclusively for Editor macros and are thus not listed in the following tables.

Description

The character generated by the space bar is called a "blank". Blanks at the end of a line are not sent to the computer.

"Text" refers to the text in the line where the cursor is located. The left-hand part of the line reserved for the record number is not considered part of the "text".

A "word" is a character string delimited by blanks, the beginning of the "text" or the end of the "text". Blanks (or a number of blanks) are considered to be part of the preceding "word".

Notes

If onscreen changes have been made for a file opened for read only, the appropriate error message will appear. Any subsequent instructions not be accepted until these onscreen changes have been either erased with the control command CLEAR or declared invalid with the control command IGNORE.

If the execution of a control command requires data to be removed from the screen which have been altered but not yet sent to the computer with ENTER, CR or a function key, the control

command will be rejected. This, however, does not apply to control commands meant to erase data.

If a line is to be inserted onscreen with, for example, the INS_LINE control command, this means that all following lines have to be shifted one line downwards and that the bottom line (and any lines belonging to the record involved) must be erased from the screen display. If this bottom onscreen record (to which the last screen line belongs) has been altered but not yet sent to the computer, this alteration will be lost. In this case, the control command INS_LINE will therefore be rejected.

The same applies to lines shifted downwards when entering data in the insert mode. If altered data have to be shifted beyond the screen display, data input will be blocked.

Cursor movement

CUR_UP Cursor up

Cursor moves one line up

CUR_DN Cursor down

Cursor moves one line down

CUR_RI Cursor right

Cursor moves one character to the right

CUR_LE Cursor left

Cursor moves one character to the left

HOME "Home / Skip to command line"

When located in the instruction line (bottom line of the screen), cursor jumpt to the beginning of the instruction; if already at the beginning of the instructin, or not in the instruction line at all, the cursor will jump to the start of the first "text" line. If it is already located there, it will jump to the beginning of the instruction line.

CMD_LINE "Jump to command line"

Cursor jumps to the beginning of the instruction (command) line.

LF Line feed: "Skip to next start of text"

Cursor jumps to the next line at the start of the "text".

CR Carriage return: "End of input" / "Line feed"

- when entering data (e.g. after instruction IE): CR CR has the same effect as LF, i.e. the cursor jumps to the next line. In this case, ENTER must be used to send data to the computer.

- otherwise: data and/or instruction will be sent to the computer (as if ENTER has been given).

TAB "Skip to next tabulator"

Cursor jumps to the next tab stop.

Note: tab stops can be defined with the instruction T (see page 190).

SKP_BEG "Skip to start of text"

Cursor jumps to the beginning of the "text". If it is already located there, it will jump to the beginning of the "text" in the preceding line.

SKP_END "Skip to end of text"

Cursor jumps to the end of the "text". If it is already located there, it will jump to the end of the "text" in the following line.

SKP_WORD "Skip to next word / end of text"

same as SKP_RI

SKP_RI "Skip to next word / end of text"

Cursor jumps to the beginning of the next "word". If the cursor is located at the last word of a line, it will jump to the end of the "text". If the cursor is already located there, it will jump to the beginning of the first "word" in the following line.

SKP_LE "Skip to preceding word"

Cursor jumps to the beginning of the preceding "word". If located at or before the first "word" of a line, the cursor will jump to the end of the last "word" in the preceding line.

JMP_DN "Jump to next emphasized field"

Cursor jumps to the beginning of the next emphasized text position (e.g. following the show instruction so)

JMP_UP "Jump to preceding emphasized field"

Cursor jumps to the end of the previous emphasized text position (e.g. following the show instruction so)

Scrolling

SHW_DN "Show next screen of text"

Scrolls forwards, starting with the record in which the cursor is located. If the cursor is located in the instruction line: starting with the record which follows the last record presently on screen. Empty screen: starting with the record which follows the record corresponding to the current *-position.

If any changes have been made to screen, these must be first sent to the computer (e.g. with CR or ENTER) or declared invalid (with IGNORE).

SHW_UP "Show preceding screen of text"

Scrolls backwards, starting with the record in which the cursor is located. If the cursor is located in the instruction line: starting with the record which precedes the first record shown onscreen. Empty screen: starting with the record which precedes the record corresponding to the current *-position.

If any changes have been made to screen, these must be first sent to the computer (e.g. with CR or ENTER), declared invalid (with IGNORE), or deleted (with CLEAR).

Inserting

TGL_INS "Toggle insert mode / replace mode"

Toggles between replace mode and insert mode. In replace mode existing characters will be overwritten; in insert mode, newly-entered characters will be inserted at the current cursor position, will any previous characters between the cursor position and the end of the line being shifted to the right. If there is not enought room in the line for the existing words, the characters will be shifted word-by-word to the following (continuation) line.

SET_INS "Set insert mode"

Activates insert mode (cf. TGL_INS).

SET_REP "Set replace mode"

Activates replace mode (cf. TGL_INS).

INS_LINE "Insert line"

The current line and all following lines are shifted one line down, thus creating an empty line in which text can be subsequently entered. If possible, the newly created line will assigned its own record number automatically.

SPLIT "Split line"

Splits the current line at the cursor position. All lines following the current line are moved down one line. The "text" to the left of the cursor will remain in the current line; the rest of the "text" is shifted to the newly created line that follows. If the cursor is located at a blank, the blank will not be shifted to the following line. If the cursor is located after the last character of the line, SPLIT has the same effect as INS_LINE, except that the empty line will be inserted after the current line. If possible, the newly created line will be assigned its own record number automatically.

DUP_LINE "Duplicate line"

Makes a duplicate (copy) of the current line. All lines following the current line are moved down one line. The "text" of the current line is copied to the newly created line. If possible, the newly created line will be assigned its own record number.

DATE_1 "Write date as xx.xx.xx"

Inserts the date in the form xx.xx.xx (e.g. 01.12.90) at the current cursor position onscreen.

DATE_2 "Write date as xx. xxx. xxxx"

Inserts the date in der form xx. xxx. xxxx (e.g. 12. Jan. 1990) at the current cursor position onscreen.

DATE_3 "Write date as xx. xxxxxxxx xxxx"

Inserts the date in the form xx. xxxxxxxx xxxx (e.g. 12. January 1990) at the current cursor position onscreen.

PAGE_NR "Write current page number"

Inserts the current page number (i.e. the page number appearing after "*=" in the instruction line) at the cursor position onscreen.

PAGE_NR_INC "Write current page number incremented by 1"

Inserts a page number one greater than the current page number (i.e. that appearing after "*=" in the instruction line) at the cursor position onscreen.

PAGE_NR_DEC "Write current page number decremented by 1"

Inserts a page number one less than the current page number (i.e. that appearing after "*=" in the instruction line) at the cursor position onscreen.

Deleting

DEL Delete: "Delete character in line"

Deletes the character at the current cursor position and moves all following characters in the line one position to the left.

BSP Backspace: "Delete character in line"

Deletes the character to the left of the current cursor position and moves all following characters in the line one position to the left. The cursor also moves one position to the left.

TGL_DEL "Toggle delete mode / backspace mode"

Toggles between delete mode and backspace mode. In delete mode DEL has the same effect as described for DEL above; in backspace mode DEL is the same as BSP.

SET_DEL "Set delete mode"

Activates delete mode (cf. TGL_DEL).

DEL_BEG "Delete to start of text"

Deletes the "text" to the left of the current cursor position and moves the remaining "text" to the beginning of the line. The cursor is then positioned at the first "text" character in the line.

UND_BEG "Undo (= Insert text deleted with) DEL_BEG"

Inserts the characters last deleted with DEL_BEG in front of the current cursor position. All following characters in the line will be moved to the right. The cursor is also shifted and appears after the inserted characters.

DEL_END "Delete to end of text"

Deletes all characters from the current cursor position to the end of the "text".

UND_END "Undo (= Insert text deleted with) DEL_END"

Inserts the characters last deleted with DEL_END plus a blank space in front of the current cursor position. All following characters in the line are shifted to the right. The cursor is not shifted and will be located at the begin of the inserted characters.

DEL_WORD "Delete to next word / end of text"

Deletes the rest of the "word" starting at the current cursor position (or the entire "word" if the cursor is located at the beginning of the "word"). The following characters will be moved the corresponding number of characters to the left.

UND_WORD "Undo (= Insert text deleted with) DEL_WORD"

Inserts the characters last deleted with DEL_WORD in front of the current cursor position. All following characters in the line will be moved to the right. The cursor remains at its original position at the beginning of the inserted characters.

DEL_LINE "Delete line"

Deletes the current line (only on screen if this deletes a complete record and its record number) and moves all following lines one line up.

Note: If a complete record and its record number is deleted from the screen, this does not delete (or change) this record in the file (cf. example on page 182). If the entire record is to be deleted from the file, DEL_REC should be used instead.

UND_LINE "Undo (= Insert line deleted with) DEL_LINE"

The current line and all following lines are moved one line down. The line last deleted with DEL_LINE is inserted into the newly created line. If

possible, the newly created line will be assigned its own record number automatically.

DEL_REC "Delete record"

Deletes the record to which the current line belongs.

UND_REC "Undo (= Insert record deleted with) DEL_REC"

The current line and all following lines are moved down as required. The "text" of the record last deleted with DEL_REC is inserted into the newly created line(s). If possible, the new record will be assigned its own record number automatically.

CLEAR "Clear screen"

Clears the screen. This command is useful for canceling changes made onscreen, or for entering an instruction which is longer than a single instruction line.

Note: If onscreen changes have been made for a file opened only for reading, the appropriate error message will appear. Any subsequent instructions will not be accepted until these onscreen changes have been either erased with the control command CLEAR or declared invalid with the control command IGNORE.

Marking, copying, deleting, inserting and searching text

MRK_INI "Initialize marking of text"

Defines the beginning of marked text.

Note: once MRK_INI has been invoked, the cursor must then be positioned at the end of the text to be marked. Marking is then completed with one of the following control commands. The screen may not be changed in the meantime. If any action is taken that alters the screen content, MRK_IGN will be ignored automatically.

MRK_IGN "Ignore marking of text"

Cancels marking begun with MRK_INI.

MRK_REP "Copy marked text to the buffer"

Copies marked text to the buffer. Any text already present in the buffer will be erased.

MRK_BEG "Insert marked text at beginning of buffer"

Copies marked text to the buffer. If the buffer already contains text, the newly marked text will be inserted at the beginning of the old text.

MRK_END "Append marked text at end of buffer"

Copies marked text to the buffer. If the buffer already contains text, the newly marked text will be inserted at the end of the old text.

MRK_INS "Insert buffer contents into the text"

Inserts buffer text at the current cursor position. All following characters in the line will be shifted to the right. The cursor will also be shifted to the right, appearing after the inserted characters.

MRK_DEL_REP "Delete marked text, copy it to the buffer"

As in MRK_REP, except that that the text marked onscreen is deleted.

MRK_DEL_BEG "Delete marked text, insert it at beginning of buffer"

As in MRK_BEG, except that the text marked onscreen is deleted.

MRK_DEL_END "Delete marked text, append it at end of buffer"

As in MRK_END, except that the text marked onscreen is deleted.

MRK_DEL_INS "Replace marked text with buffer contents"

Deletes marked text without saving it to the buffer; marked text is replaced by the buffer text at this position.

MRK_DEL_DEL "Delete marked text without saving it"

Deletes marked text without saving it to the buffer.

MRK_FND "Find next occurance of marked text"

Begins a line-by-line search starting at the cursor position for onscreen text which matches the marked text. The cursor will be positioned at the beginning of the found text. No distinction is made between uppercase and lowercase letters.

If the search is to be repeated for the same text, this control command can be invoked without having to remark the text.

MRK_DEL_FND "Delete marked text and find next occurance"

As in MRK_FND, except that the previously marked text is deleted from the screen.

Running number

RD_NUM "Read and set current number"

Read a numerical sequence starting at the cursor position and saves it as a running number.

DEC_NUM "Decrement current number by 1"

Reduces the running number by 1.

INC_NUM "Increment current number by 1"

Increases the running number by 1.

WR_NUM "Write current number"

Writes the current number (i.e. any number read with RD_NUM and/or increased or decreased with INC_NUM or DEC_NUM, respectively) at the current cursor position onscreen.

Further control commands

ENTER "End of input"

Signals end of input and sends data / instruction to the computer.

JOIN "Join lines"

Appends the current line to the following line and any continuation lines involved.

RESHOW "Reshow screen"

Displays the last screen display. This can be useful for undoing any undesired screen displays.

Caution: the reshown screen display is identical to the last screen displayed. Any alterations made onscreen in the meantime will not be accounted for.

REFRESH "Refresh screen"

Restores screen contents disrupted or destroyed by system messages, disturbances in the data transfer line, etc.

CANCEL "Cancel input" / "terminate editor"

- when entering data (e.g. after the instruction IE): terminates data input, any data onscreen will be ignored.

- otherwise: terminates Editor. Any changes made onscreen, such as a new instruction, that have not been sent to the computer (e.g. with ENTER) will be ignored.

WAIT "Wait until striking any key"

Step processing of the Editor macro is interrupted and user input is expected. Pressing the space bar continues the execution of the macro. Pressing any other key will skip over the remaining control commands in the Editor macro.

IGNORE "Ignore modifications"

Any alterations made onscreen will be ignored.

HELP "call online help"

Activates the online help program (see command #HELP page 109).

Function key End of input. Data and/or instruction will be sent to the computer. The instruction defined for a function will be executed.

The use of the function keys is described in more detail on page 191.

Key combinations for control commands

Control commands are executed by using the keys in the number pad and the keys in the cursor pad. Key combinations with the number pad involve the digits 0-9, . , - + * / × and ÷ . In the DOS version, the ESC key can be used instead of / and ÷.

A key combination starting with Ctrl (e.g. Ctrl+A) requires that
- the Ctrl key is first held down and
- the key following the plus sign is then pressed (e.g. the letter a or A for the combination Ctrl+A).
- the Ctrl key is then released,
- and any other keys involved in the combination (e.g. after Ctrl+K) are pressed and released separately.

For all other key combinations, keys must be pressed in the given order and then released.

UNIX and VMS: VT100 keyboard

Name	Keys	Function	Alternate keys
CUR_UP	↑	Cursor up	
CUR_DN	↓	Cursor down	
CUR_LE	←	Cursor left	
CUR_RI	→	Cursor right	
BSP	BS	Backspace	CTRL+H
DEL	Del	Delete	
TAB	Tab	Skip to next tabulator	CTRL+I
LF	LF	Skip to next start of text	CTRL+J
ENTER	ENTER	End of input	
CR	Return	End of input	CTRL+M
SKP_BEG	PF2	Skip to start of text	
SKP_WORD	PF3	Skip to next word / end of text	
SKP_END	PF4	Skip to end of text	
SHW_DN	,	Show next screen of text	
SHW_UP	-	Show preceding screen of text	
HOME	.	Home / Skip to command line	
TGL_INS	PF1 ENTER	Toggle insert / replace mode	CTRL+A
SPLIT	PF1 Return	Split line	
DEL_BEG	PF1 PF2	Delete to start of text	
DEL_WORD	PF1 PF3	Delete to next word / end of text	
DEL_END	PF1 PF4	Delete to end of text	
DEL_LINE	PF1 -	Delete line	
INS_LINE	PF1 ,	Insert line	
CLEAR	PF1 .	Clear screen	
DEL_REC	PF1 Del	Delete record	
REFRESH	PF1 PF1 ENTER	Refresh screen	CTRL+W
JOIN	PF1 PF1 Return	Join lines	
UND_BEG	PF1 PF1 PF2	Insert text deleted with DEL_BEG	
UND_WORD	PF1 PF1 PF3	Insert text deleted with DEL_WORD	
UND_END	PF1 PF1 PF4	Insert text deleted with DEL_END	
UND_LINE	PF1 PF1 -	Insert line deleted with DEL_LINE	
DUP_LINE	PF1 PF1 ,	Duplicate line	
RESHOW	PF1 PF1 .	Reshow screen	CTRL+R
UND_REC	PF1 PF1 DEL	Insert record deleted with DEL_REC	

UNIX and VMS: VT100-Tastatur (continued)

Name	Keys	Function
	CTRL+P	Display cursor position
TGL_DEL	CTRL+B	Toggle delete / backspace mode
CANCEL	CTRL+Z	Cancel input, terminate editor
MRK_INI	CTRL+K Return	Initialize marking of text
MRK_BEG	CTRL+K PF2	Copy marked text to buffer begin
MRK_REP	CTRL+K PF3	Copy marked text to buffer
MRK_END	CTRL+K PF4	Copy marked text to buffer end
MRK_INS	CTRL+K Enter	Insert buffer contents into text
MRK_IGN	CTRL+K Del Return	Ignore marking of text
MRK_DEL_BEG	CTRL+K Del PF2	Move marked text to buffer begin
MRK_DEL_REP	CTRL+K Del PF3	Move marked text to buffer
MRK_DEL_END	CTRL+K Del PF4	Move marked text to buffer end
MRK_DEL_INS	CTRL+K Del Enter	Replace marked text with buffer
MRK_DEL_DEL	CTRL+K Del Del	Delete marked text
HELP	CTRL+V	Call online help
INTRPT	CTRL+C	Interrupt program
	CTRL+G	Set cursor type
	CTRL+F	Set color attributes
	CTRL+L	Set length of line
	PF1+a - PF1+z	Invoke macro: a to z
	PF1 PF1 name Return	Invoke macro: name
	0 - 9	Function keys: F10, F1 to F9
	PF1 0 - PF1 9	Function keys: F20, F11 to F19
	PF1 PF1 0 - PF1 PF1 9	Function keys: F30, F21 to F29

DOS: AT keyboard - English layout

Name	Keys	Function	Alternate keys
CUR_UP	↑	Cursor up	
CUR_DN	↓	Cursor down	
CUR_LE	←	Cursor left	
CUR_RI	→	Cursor right	
BSP	Bsp	Backspace	Ctrl+H
DEL	Del	Delete	
TAB	Tab	Skip to next tabulator	Ctrl+I
LF	Ctrl+Return	Skip to next start of text	Ctrl+J
CR	Return	End of input	Ctrl+M
ENTER	+	End of input	
SKP_BEG	Home	Skip to start of text	
SKP_WORD	-	Skip to next word / end of text	Ctrl+→
SKP_END	End	Skip to end of text	
SKP_RI	-	Skip to next word / end of text	Ctrl+→
SKP_LE		Skip to preceding word	Ctrl+←
SHW_DN	PgDn	Show next screen of text	
SHW_UP	PgUp	Show preceding screen of text	
HOME	*	Home / Skip to command line	
TGL_INS	Ins	Toggle insert / replace mode	Ctrl+A
SPLIT	ESC Return	Split line	
DEL_BEG	ESC Home	Delete to start of text	
DEL_WORD	ESC -	Delete to next word / end of text	
DEL_END	ESC End	Delete to end of text	
DEL_LINE	ESC PgUp	Delete line	
INS_LINE	ESC PgDn	Insert line	
CLEAR	ESC *	Clear screen	Ctrl+C
DEL_REC	ESC Del	Delete record	
REFRESH	ESC ESC +	Refresh screen	Ctrl+W
JOIN	ESC ESC Return	Join lines	
UND_BEG	ESC ESC Home	Insert text deleted with DEL_BEG	
UND_WORD	ESC ESC -	Insert text deleted with DEL_WORD	
UND_END	ESC ESC End	Insert text deleted with DEL_END	
UND_LINE	ESC ESC PgUp	Insert line deleted with DEL_LINE	
DUP_LINE	ESC ESC PgDn	Duplicate line	
RESHOW	ESC ESC *	Reshow screen	Ctrl+R
UND_REC	ESC ESC DEL	Insert record deleted with DEL_REC	

DOS: AT keyboard - English layout (continued)

Name	Keys	Function
JMP_UP	Ctrl+↑	Jump to next emphasized field
JMP_DN	Ctrl+↓	Jump to prec. emphasized field
TGL_DEL	Ctrl+B	Toggle delete / backspace mode
CANCEL	Ctrl+Z	Cancel input, terminate editor
MRK_INI	Ctrl+K Return	Initialize marking of text
MRK_BEG	Ctrl+K Home	Copy marked text to buffer begin
MRK_REP	Ctrl+K -	Copy marked text to buffer
MRK_END	Ctrl+K End	Copy marked text to buffer end
MRK_INS	Ctrl+K Enter	Insert buffer contents into text
MRK_IGN	Ctrl+K Del Return	Ignore marking of text
MRK_DEL_BEG	Ctrl+K Del Home	Move marked text to buffer begin
MRK_DEL_REP	Ctrl+K Del -	Move marked text to buffer
MRK_DEL_END	Ctrl+K Del End	Move marked text to buffer end
MRK_DEL_INS	Ctrl+K Del Enter	Replace marked text with buffer
MRK_DEL_DEL	Ctrl+K Del Del	Delete marked text
HELP	Ctrl+V	Call online help
INTRPT	Ctrl+C	Interrupt program
	Ctrl+G	Set cursor type
	Ctrl+F	Set color attributes
	Ctrl+L	Set length of line
	Alt a - Alt z	Invoke macro: a to z
	ESC ESC name Return	Invoke macro: name
	F1 - F10	Function keys: F1 to F10
	Shift+F1 - Shift+F10	Function keys: F11 to F20
	Ctrl+F1 - Ctrl+F10	Function keys: F21 to F30
	Alt+F1 - Alt+F10	Function keys: F31 to F40

DOS: MF keyboard - English layout

Name	Keys	Function	Alternate keys
CUR_UP	↑	Cursor up	
CUR_DN	↓	Cursor down	
CUR_LE	←	Cursor left	
CUR_RI	→	Cursor right	
BSP	Bsp	Backspace	Ctrl+H
DEL	Del	Delete	
TAB	Tab	Skip to next tabulator	Ctrl+I
LF	Ctrl+Return	Skip to next start of text	Ctrl+J
CR	Return	End of input	Ctrl+M
ENTER	Enter	End of input	
SKP_BEG	Home	Skip to start of text	
SKP_WORD	-	Skip to next word / end of text	Ctrl+→
SKP_END	End	Skip to end of text	
SKP_RI	-	Skip to next word / end of text	Ctrl+→
SKP_LE		Skip to preceding word	Ctrl+←
SHW_DN	PgDn	Show next screen of text	
SHW_UP	PgUp	Show preceding screen of text	
HOME	*	Home / Skip to command line	
TGL_INS	Ins	Toggle insert / replace mode	Ctrl+A
SPLIT	/ Return	Split line	
DEL_BEG	/ Home	Delete to start of text	
DEL_WORD	/ -	Delete to next word / end of text	
DEL_END	/ End	Delete to end of text	
DEL_LINE	/ PgUp	Delete line	
INS_LINE	/ PgDn	Insert line	
CLEAR	/ *	Clear screen	Ctrl+C
DEL_REC	/ Del	Delete record	
REFRESH	/ / +	Refresh screen	Ctrl+W
JOIN	/ / Return	Join lines	
UND_BEG	/ / Home	Insert text deleted with DEL_BEG	
UND_WORD	/ / -	Insert text deleted with DEL_WORD	
UND_END	/ / End	Insert text deleted with DEL_END	
UND_LINE	/ / PgUp	Insert line deleted with DEL_LINE	
DUP_LINE	/ / PgDn	Duplicate line	
RESHOW	/ / *	Reshow screen	Ctrl+R
UND_REC	/ / DEL	Insert record deleted with DEL_REC	

DOS: MF keyboard - English layout (continued)

Name	Keys	Function
JMP_UP	Ctrl+↑	Jump to next emphasized field
JMP_DN	Ctrl+↓	Jump to prec. emphasized field
TGL_DEL	Ctrl+B	Toggle delete / backspace mode
CANCEL	Ctrl+Z	Cancel input, terminate editor
MRK_INI	Ctrl+K Return	Initialize marking of text
MRK_BEG	Ctrl+K Home	Copy marked text to buffer begin
MRK_REP	Ctrl+K -	Copy marked text to buffer
MRK_END	Ctrl+K End	Copy marked text to buffer end
MRK_INS	Ctrl+K Enter	Insert buffer contents into text
MRK_IGN	Ctrl+K Del Return	Ignore marking of text
MRK_DEL_BEG	Ctrl+K Del Home	Move marked text to buffer begin
MRK_DEL_REP	Ctrl+K Del -	Move marked text to buffer
MRK_DEL_END	Ctrl+K Del End	Move marked text to buffer end
MRK_DEL_INS	Ctrl+K Del Enter	Replace marked text with buffer
MRK_DEL_DEL	Ctrl+K Del Del	Delete marked text
HELP	Ctrl+V	Call online help
INTRPT	Ctrl+C	Interrupt program
	Ctrl+G	Set cursor type
	Ctrl+F	Set color attributes
	Ctrl+L	Set length of line
	Alt+a - Alt+z	Invoke macro: a to z
	/ / name Return	Invoke macro: name
	F1 - F10	Function keys: F1 to F10
	Shift+F1 - Shift+F10	Function keys: F11 to F20
	Ctrl+F1 - Ctrl+F10	Function keys: F21 to F30
	Alt+F1 - Alt+F10	Function keys: F31 to F40

DOS: AT keyboard - German layout

Name			Keys	Function	Alternate keys
CUR_UP			↑	Cursor up	
CUR_DN			↓	Cursor down	
CUR_LE			←	Cursor left	
CUR_RI			→	Cursor right	
BSP			Bsp	Backspace	Strg+H
DEL			Lösch	Delete	
TAB			Tab	Skip to next tabulator	Strg+I
LF			Strg+Return	Skip to next start of text	Strg+J
CR			Return	End of input	Strg+M
ENTER			+	End of input	
SKP_BEG			Pos1	Skip to start of text	
SKP_WORD			-	Skip to next word / end of text	Strg+→
SKP_END			End	Skip to end of text	
SKP_RI			-	Skip to next word / end of text	Strg+→
SKP_LE				Skip to preceding word	Strg+←
SHW_DN			Bild↓	Show next screen of text	
SHW_UP			Bild↑	Show preceding screen of text	
HOME			*	Home / Skip to command line	
TGL_INS			Einfg	Toggle insert / replace mode	Strg+A
SPLIT		E/L	Return	Split line	
DEL_BEG		E/L	Pos1	Delete to start of text	
DEL_WORD		E/L	-	Delete to next word / end of text	
DEL_END		E/L	End	Delete to end of text	
DEL_LINE		E/L	Bild↑	Delete line	
INS_LINE		E/L	Bild↓	Insert line	
CLEAR		E/L	*	Clear screen	Strg+C
DEL_REC		E/L	Lösch	Delete record	
REFRESH	E/L	E/L	Enter	Refresh screen	Strg+W
JOIN	E/L	E/L	Return	Join lines	
UND_BEG	E/L	E/L	Pos1	Insert text deleted with DEL_BEG	
UND_WORD	E/L	E/L	-	Insert text deleted with DEL_WORD	
UND_END	E/L	E/L	End	Insert text deleted with DEL_END	
UND_LINE	E/L	E/L	Bild↑	Insert line deleted with DEL_LINE	
DUP_LINE	E/L	E/L	Bild↓	Duplicate line	
RESHOW	E/L	E/L	*	Reshow screen	Strg+R
UND_REC	E/L	E/L	DEL	Insert record deleted with DEL_REC	

DOS: AT keyboard - German layout (continued)

Name			Keys	Function
JMP_UP			Strg+↑	Jump to next emphasized field
JMP_DN			Strg+↓	Jump to prec. emphasized field
TGL_DEL			Strg+B	Toggle delete / backspace mode
CANCEL			Strg+Z	Cancel input, terminate editor
MRK_INI		Strg+K	Return	Initialize marking of text
MRK_BEG		Strg+K	Pos1	Copy marked text to buffer begin
MRK_REP		Strg+K	-	Copy marked text to buffer
MRK_END		Strg+K	End	Copy marked text to buffer end
MRK_INS		Strg+K	+	Insert buffer contents into text
MRK_IGN	Strg+K	Lösch	Return	Ignore marking of text
MRK_DEL_BEG	Strg+K	Lösch	Pos1	Move marked text to buffer begin
MRK_DEL_REP	Strg+K	Lösch	-	Move marked text to buffer
MRK_DEL_END	Strg+K	Lösch	End	Move marked text to buffer end
MRK_DEL_INS	Strg+K	Lösch	+	Replace marked text with buffer
MRK_DEL_DEL	Strg+K	Lösch	Lösch	Delete marked text
HELP			Strg+V	Call online help
INTRPT			Strg+C	Interrupt program
			Strg+G	Set cursor type
			Strg+F	Set color attributes
			Strg+L	Set length of line
			Alt+a - Alt+z	Invoke macro: a to z
		E/L	E/L name Return	Invoke macro: name
			F1 - F10	Function keys: F1 to F10
			Shift+F1 - Shift+F10	Function keys: F11 to F20
			Strg+F1 - Strg+F10	Function keys: F21 to F30
			Alt+F1 - Alt+F10	Function keys: F31 to F40

DOS: MF keyboard - German layout

Name	Keys	Function	Alternate keys
CUR_UP	↑	Cursor up	
CUR_DN	↓	Cursor down	
CUR_LE	←	Cursor left	
CUR_RI	→	Cursor right	
BSP	Bsp	Backspace	Strg+H
DEL	Entf	Delete	
TAB	Tab	Skip to next tabulator	Strg+I
LF	Strg+Return	Skip to next start of text	Strg+J
CR	Return	End of input	Strg+M
ENTER	Enter	End of input	
SKP_BEG	Pos1	Skip to start of text	
SKP_WORD	-	Skip to next word / end of text	Strg+→
SKP_END	Ende	Skip to end of text	
SKP_RI	-	Skip to next word / end of text	Strg+→
SKP_LE		Skip to preceding word	Strg+←
SHW_DN	Bild↓	Show next screen of text	
SHW_UP	Bild↑	Show preceding screen of text	
HOME	×	Home / Skip to command line	
TGL_INS	Einfg	Toggle insert / replace mode	Strg+A
SPLIT	÷ Return	Split line	
DEL_BEG	÷ Pos1	Delete to start of text	
DEL_WORD	÷ -	Delete to next word / end of text	
DEL_END	÷ Ende	Delete to end of text	
DEL_LINE	÷ Bild↑	Delete line	
INS_LINE	÷ Bild↓	Insert line	
CLEAR	÷ ×	Clear screen	Strg+C
DEL_REC	÷ Entf	Delete record	
REFRESH	÷ ÷ Enter	Refresh screen	Strg+W
JOIN	÷ ÷ Return	Join lines	
UND_BEG	÷ ÷ Pos1	Insert text deleted with DEL_BEG	
UND_WORD	÷ ÷ -	Insert text deleted with DEL_WORD	
UND_END	÷ ÷ Ende	Insert text deleted with DEL_END	
UND_LINE	÷ ÷ Bild↑	Insert line deleted with DEL_LINE	
DUP_LINE	÷ ÷ Bild↓	Duplicate line	
RESHOW	÷ ÷ ×	Reshow screen	Strg+R
UND_REC	÷ ÷ DEL	Insert record deleted with DEL_REC	

DOS: MF keyboard - German layout (continued)

Name	Keys	Function
JMP_UP	Strg+↑	Jump to next emphasized field
JMP_DN	Strg+↓	Jump to prec. emphasized field
TGL_DEL	Strg+B	Toggle delete / backspace mode
CANCEL	Strg+Z	Cancel input, terminate editor
MRK_INI	Strg+K Return	Initialize marking of text
MRK_BEG	Strg+K Pos1	Copy marked text to buffer begin
MRK_REP	Strg+K -	Copy marked text to buffer
MRK_END	Strg+K Ende	Copy marked text to buffer end
MRK_INS	Strg+K Enter	Insert buffer contents into text
MRK_IGN	Strg+K Entf Return	Ignore marking of text
MRK_DEL_BEG	Strg+K Entf Pos1	Move marked text to buffer begin
MRK_DEL_REP	Strg+K Entf -	Move marked text to buffer
MRK_DEL_END	Strg+K Entf Ende	Move marked text to buffer end
MRK_DEL_INS	Strg+K Entf Enter	Replace marked text with buffer
MRK_DEL_DEL	Strg+K Entf Entf	Delete marked text
HELP	Strg+V	Call online help
INTRPT	Strg+C	Interrupt program
	Strg+G	Set cursor type
	Strg+F	Set color attributes
	Strg+L	Set length of line
	Alt+a - Alt+z	Invoke macro: a to z
	÷ ÷ name Return	Invoke macro: name
	F1 - F10	Function keys: F1 to F10
	Shift+F1 - Shift+F10	Function keys: F11 to F20
	Strg+F1 - Strg+F10	Function keys: F21 to F30
	Alt+F1 - Alt+F10	Function keys: F31 to F40

P a r a m e t e r s

Survey:

Introduction

Parameters are used to describe in more detail the desired effects of a TUSTEP program. Parameter functions are outlined in the description of the individual programs.

Parameters can be interpreted according to either "old" or "new" conventions. The old convention of interpreting parameters was the one first used in TUSTEP and whose concept was based on the punch card, then the standard medium for the input of data and programs. The notation used here took into account the fact that punch cards ordinarily did not distinguish between uppercase and lowercase letters, and that umlauts were not capable of being interpreted. The "new" method of interpreting parameters has been designed with more modern input technology in mind; the limitations imposed by the "old" method therefore no longer apply. The command #PARAMETER (see page 154) can be used to specify whether parameters are to be interpreted according to the old or new convention.

The default value here is still the old parameter interpretation - future versions of TUSTEP will use the new parameter interpretation as the default value. It is therefore recommended to specify in each command sequence which convention is to be used for interpreting parameters with the command #PARAMETER. In the following description of various parameter types, passages which apply exclusively to either the old or new convention will be marked accordingly.

Parameter format

Column 1-3: parameter identification consisting of 1 to 3 characters, left-aligned. Parameters containing blanks in columns 1 to 3 are treated as comments. They may occur at any place.

4-5: blank: parameter is evaluated.

"+n": parameter is evaluated only if the selection switch n (n = 1 to 7) has been set.

" n": is the same as "+n"

"-n": parameter is evaluated only if the selection switch n (n = 1 to 7) has not been set.

6-7: blank, or a right-aligned number (depending on the respective program)

8: blank

9: blank normal case

":" if the information field in column 11 begins with a blank (e.g. for parameter type VII)

"=" if the information field refers to a preceding parameter of the same type whose information field is to be used for the current parameter. Column 11-13 must contain the parameter identification and column 16-17 must contain the number given in column 6-7 of the parameter being referred to.

10: blank

11-80: Information field. Trailing blanks in the parameter information field will be ignored.

If a line's information field is too small for the parameter's data, continuation lines (with the same parameter identification) may be used. The continuation line for parameters using delimiter characters must use the same delimiter character. The delimiter in column 11 of a continuation line and the one at the end of the preceding line are interpreted as a single delimiter character. For better readability, it is recommended to take advantage of this feature by dividing lines in such a way that every information field begins with a delimiter character (column 11) and also ends with a delimiter character (last column used). However, if a character string is continued into the next line, a delimiter character may not be written at the end of the line or in column 11 of the following line. In this case, it should be remembered that blanks at the end of a line will be ignored.

Rules for uppercase and lowercase letters

a) for parameter types II to IV:

NEW: Uppercase and lowercase will be interpreted as such.

OLD: The characters "<" and ">" are interpreted as shift characters for shifting to uppercase and lowercase letters, respectively. Each shift character is effective until the next shift character, or until the end of the paramter. The letters following a shift character may be typed as either uppercase or lower case. If no shift character is present at the start of a string, all letters up to the first occurance of a shift character will be converted to lowercase letters. After "<" letters are read as uppercase letters, after ">" as lowercase letters. "<<" and ">>" are used to represent the characters "<" and ">", respectively.

b) for parameter types V to X:

Letters specified in the information field refer to the respective uppercase or lowercase letter (i.e., it does not matter whether the letter is entered as uppercase or lowercase). If upper case letters are to be distinguished from lower case letters, the respective letter has to be marked by a preceding "<" or ">". Thus, "a" and "A" will be interpreted as either an upper or lower case a, ">a" and ">A" will be interpreted as a lower case a, "<a" and "<A" will be interpreted as an upper case A. "<<" and ">>" are used to represent the characters "<" and ">", respectively.

c) for parameter type X:

For character search strings, the same applies as for parameter type IX.

NEW: Uppercase and lowercase letters in replacement character strings will be interpreted as such, unless a lowercase or uppercase letter has been specifically requested with the shift character ">" or "<".

OLD: Uppercase and lowercase letters in replacement character strings will be interpreted as lowercase letters, unless marked otherwise with a preceding "<" (for uppercase) before each letter. Lowercase letters may be marked with a ">", but this has no real effect when letters are interpreted.

Rules for the circumflex character

NEW: A circumflex "^" (entered as "^^") will be interpreted as a normal character.

OLD: A circumflex "^" (entered as "^^") is regarded as a control character and will be interpreted along the the character immediately following it as a single character. (cf. the tables 7-bit and 8-bit TUSTEP Character Sets on page 298f).

When using "<" or ">" to mark a letter entered with "^", please observe the proper sequence: e.g. ">^a" for a lowercase "ä" ("^>a" would be interpreted as "]a").

Parameter type I: numerical values

Specified in the information field are numerical values which are written as arabic numbers. Digits written directly one after the other are interpreted as a single number. Individual numbers may be separated by any character (excluding digits). Numerical values must be entered in the sequence required by the respective description; in this case, no numerical value may be omitted. If fewer numbers are given than required by the respective program, its default numerical values will be inserted for the missing numerical values.

If one line is too small for the parameter's data, continuation lines (having the same parameter identification) may be used. A blank is always added at the end of each line.

For this parameter type, column 9 may only contain a blank, i.e. reference cannot be made to a preceding parameter.

Parameter type II: text parts

Specified in the information field contains are text parts which are separated from each other by a delimiter character of the user's choice. The delimiter is the first character (column 11) in the information field of the parameter. The last text part must also be followed by a delimiter.

If fewer text parts are given than required by the respective program, its default text parts will be inserted for the missing text parts.

If one line is too small for the parameter's data, continuation lines (with the same parameter identification and same delimiter character, cf. page 244) may be used. If a text part is to be continued in the next line, a "-" can be used as a hyphen, provided that it has not been chosed as the delimiter character. However, a "-" at the end of a line will only interpreted as a hyphen if the next-to-last character is a letter and the third-to-last character is not a control character ($, &, @, \, #, %).

Parameter type III: text part comparison table

Specified in the information field are text parts which are separated from each other by a delimiter character of the user's choice. The delimiter is the first character (column 11) in the information field of the parameter. The last text part must also be followed by a delimiter.

The significance of uppercase and lowercase letters in the program can be obtained from the description for the respective parameter. Character groups and string groups may not be specified (see below).

If one line is too small for the parameter's data, continuation lines (with the same parameter identification and same delimiter character, cf. page 244) may be used.

Parameter type IV: text part replacement table

Specified in the information field are pairs of text parts whose first text part is used for comparison with the text part to be processed. Any matching text parts will be replaced by the pair's second text part. The comparison text part and the substitution text part may be of different length.

The text parts specified in the information field are separated from each other by a delimiter of the user's choice. The delimiter is the first character (column 11) in the information field of the parameter. The last text part must also be followed by a delimiter.

The significance of uppercase and lowercase letters in the comparison text parts can be obtained from the description for the relevant parameter. Character groups and string groups may not be specified (see below).

If one line is too small for the parameter's data, continuation lines (with the same parameter identification and same delimiter character, cf. page 244) may be used.

Parameter card type V: character groups and string groups

A character group is a combination of single characters. Once defined, a character group can be referred to by specifying its group identification in subsequent parameters of the type VI to X (for type X, only in character search strings) of the same program. The characters of the specified character group will be equally treated as substitutes for the group identification.

A string group is a combination of character strings. Once defined, a string group can be referred to by giving its group identification in subsequent parameters of the type IX and X (for type X, only in character search strings) of the same program. The character strings of the specified string group

will be equally treated as a substitute for the group identification.

The group identifications ">n" and "<n" are used to define character groups and string groups, and to refer to groups so defined, Here, "n" is represents a single digit, so that a maximum of 20 groups may defined at one time. The definition of a group remains in effect for all subsequent parameters until the group is redefined (by the definition of either a character group or of a string group).

Priority assignments in character groups and string groups

If the definition of a character group or a string group contains a group identification, this will always be interpreted as an identification for the corresponding character group (even if a string group definition has used the same identification). This also applies to parameter types VI to VIII. For parameter types IX and X, a group identification will be interpreted as an identification for the respective string group if this has already been defined. Otherwise it will be interpreted as an identification for the corresponding character group.

Predefined character groups

In addition to user-defined character groups, there are six internally-defined character groups with the following character group identifications:

>*	all lowercase letters of the 7-bit and 8-bit TUSTEP character set
<*	all uppercase letters of the 7-bit and 8-bit TUSTEP character set
>/	all digits of the 7-bit and 8-bit TUSTEP character set
</	all letters of the 7-bit and 8-bit TUSTEP character set
>%	all characters of the 7-bit TUSTEP character set
<%	all characters of the 7-bit and 8-bit TUSTEP character set

Defining character groups

The parameter identification for defining a character group consists of the group identification in column 1 and 2, followed by a "Z" written in column 3.

To be specified in the information field (starting with column 11, with no intervening spaces) are characters which are assigned to the group to be defined. The identification of a character group previously defined by the user or the identification of an internally-defined character group may be given in place of a character. The information field is processed from left to right, with the characters being written to the (originally empty) group. The character string "><" causes the characters following it to be deleted from the group. The character string "<>" may be used to cancel "><", i.e. the following characters will be again be added to the group.

Defining (character) string groups

The parameter identification for defining a string group consists of the group identification in columns 1 and 2, followed by an "S" in column 3.

To be specified in the information field are strings assigned to the string group to be defined. A previously-defined character group identification may be given instead of a single character of a string.

The character strings given in the information field are separated from each other by a delimiter character of the user's choice. The delimiter is the first character (column 11) in the information field of the parameter. The character string must also followed by a delimiter.

Two kinds of character strings may be specified: "search strings" and "exception strings" (= character search strings which are to be ignored during a search). They are separated from each other by two consecutive delimiters. Such double delimiters can be used to switch between search strings and exception strings as often as desired.

The texts are always scanned from left to right. If a string, starting in the text at the current scan position, corresponds to more than one search string, then the longer string is given precedence in each case. For strings of equal length, priority corresponds to the order in which they were given in the parameter. If an exception string is found, this has the same effect as if none of the characters in the string group had been found.

Caution: Because of this search procedure, exception strings included in a search string should start with a text identical to the start of the text in the other string groups to be searched (e.g. "/xy//xy*/", but not "/xy//*xy/"). Furthermore, attention should be given to the order in which search strings and exception strings of equal length are entered. Specifying, for example, "/>*>*//xy/" will not achieve the desired result because both character strings, ">*>*" and "xy", are two characters long (">*" is regarded as one character since it stands for a lowercase letter) steht), and that the check in the text for any occurances of the character string "xy" will be overridden by the search string ">*>*", since it has been specified before the exception string "xy". To make sure that the exception string will have an effect, the two strings should be specified in the order: "//xy//>*>*/".

If one line is too small for the parameter's data, continuation lines (with the same parameter identification and same delimiter character, cf. page 244) may be used.

In order to avoid errors, the following is recommended:

If an exception string is to be given at the beginning of an information field, it is possible to switch from search strings to exception strings by using two consecutive delimiters at the beginning of the information field.

If an exception string is given at the end of a line, this line should be ended with two consecutive delimiters placed at the end of the information field, thereby switching from exception strings to search strings.

Parameter type VI: character table

To be specified in the information field (starting with column 11, with no intervening spaces) are characters which are to be entered into the table. Unless otherwise specified in the respective parameter's description, characters may be given in any order.

The identification of a previously defined character group may be given in place of a character. The characters of the specified character group will be treated as if they were all located at the position occupied by the character group identification.

Parameter type VII: sort alphabet table

The information field is interpreted in the same way as for a character table. However, the given sequence of individual characters is always significant. Their given sequence determines the priority in which characters are sorted or compared, i.e., the first character (or all the characters of the character group given at the first character position) is assigned the lowest priorty, the second character (or the characters of the corresponding group) has the next higher priority etc. Characters (and character groups) to be assigned highest priority may be written after the character combination "><". During sorting they would sorted at the end of the sort. If less characters are specified than available in the entire character set, any missing characters will be inserted according to their priority in the standard sort order (see page 355).

Parameter type VIII: character string comparison table

To be specified in the information field are strings assigned to the string group to be defined. A previously-defined character group identification may be given instead of a single character of a string.

The character strings given in the information field are separated from each other by a delimiter character of the user's choice. The delimiter is the first character (column 11) in the information field of the parameter. The character string must also followed by a delimiter.

Two kinds of character strings may be specified: "search strings" and "exception strings" (= character search strings; they are separated from each other by a double delimiter character. A double delimiter can therefore be used to switch between search strings and exception strings as often as desired.

Special features of parameter type VIIIa

Texts will only be checked as to whether they begin with one of the given strings. If the text beginning is identical to more than one search or exception strings, the longer string will be given priority in each case. For strings of equal length, their priority will respond to the order in which they were specified. If (in accordance with the priority sequence described here) the text matches an exception string, this has the same effect as if none of the specified strings agreed with the beginnng of the text.

Note: Attention should to be given to the order in which search strings and exception strings of equal length are entered. Specifying, for example, "/>*>*//xy/" will not achieve the desired result because both character strings, ">*>*" and "xy", are two characters long (">*" is regarded as one character since it stands for a lowercase letter) and that the check in the text for any occurances of the character string "xy" will be overridden by the search string ">*>*", since it has been specified before the exception string "xy". To make sure that the exception string will have an effect, the two strings should be specified in the order: "//xy//>*>*/".

Special features of parameter type VIIIb

Texts will only be checked as to whether they end with one of the given strings. If the end of the text is identical to more than one search or exception strings, the longer string will be given priority in each case. For strings of equal length, their priority will respond to the order in which they were specified. If (in accordance with the priority sequence described here) the text matches an exception string, this has the same effect as if none of the specified strings agreed with the end of the text.
Note: Attention should be given to the order in which search strings and exception strings of equal length are entered. Specifying, for example, "/>*>*//xy/" will not achieve the desired result because both character strings, ">*>*" and "xy", are two characters long (">*" is regarded as one character since it stands for a lowercase letter) steht), and that the check in the text for any occurances of the character string "xy" will be overridden by the search string ">*>*", since it has been specified before the exception string "xy". To make sure that the exception string will have an effect, the two strings should be specified in the order: "//xy//>*>*/".

Parameter type IX: character string search table

Entered in the information field are character strings which are to be searched. In place of a single character, a predefined identification for a character or string group may be specified. In addition, a reference (pointer) can also be given, which refers to other elements (see below) of the same character string. Furthermore, frequency conditions may be added to characters, character groups, string groups and references, and ambient conditions for the character strings may also be specified.

The character strings given in the information field are separated from each other by a delimiter character of the user's choice. The delimiter is the first character (column 11) in the information field of the parameter. The character string must also followed by a delimiter.

Two kinds of character strings may be specified: "search strings" and "exception strings" (= character search strings which are to be ignored during a search). They are separated from each other by two consecutive delimiters. Such double delimiters can be used to switch between search strings and exception strings as often as desired.

The texts are always scanned from left to right. If a string, starting in the text at the current scan position, corresponds to more than one search string, then the longer string is given precedence in each case. For strings of equal length, priority corresponds to the order in which they were given in the parameter. If an exception string is found during the search (in accordance with the priority rules stated here), the corresponding characters in the text will be skipped over. The search in the text will then continue at the position following the exception string. However, this does not apply to exception strings specified in string groups; here the exception string is only used to specify that other character strings given in the same string group may be passed over and thus none of the search strings specified in the string group will be found beyond the current position in the text.

Caution: Based on these outlined procedures for searching the text according to given character strings, a search of the text "...01234..." for the string "/1234/01/" would find the string "01" but not the string "1234". Reason: Although the string "1234" is longer than the string "01", what is decisive here is the fact that string "01" is further to the left in the text and (since the search is carried out from left to right) is thus the first string found. Similarly, a search of the text "...01234..." for the string "/1234//01/" will first find the exception string "01". The text search will then be continued at the position that follows the string "01", thereby passing over (i.e. no longer finding) the string. "1234". In addition, attention should be given to the order in which search and exception strings of equal length are given. Specifying, for example, "/>*>*//xy/" will not achieve the desired result because both character strings, ">*>*" and "xy", are two

characters long (">*" is regarded as one character since it stands for a lowercase letter) and that the check in the text for any occurances of the character string "xy" will be overridden by the search string ">*>*", since it has been specified before the exception string "xy". To make sure that the exception string will have an effect, the two strings should be specified in the order: "//xy//>*>*/".

If one line is too small for the parameter's data, continuation lines (with the same parameter identification and same delimiter character, cf. page 244) may be used.

Characters, character group identifications, string group identifications and references will be refered to in the following as "elements" of the character search string. A frequency condition preceding a character, a character group identification, a string group identification or a reference is considered to be part of the element.

Frequency conditions

><n If an element contains the frequency condition "><n", the characters in the text to be searched which correspond to the character (or to the given group identification or to the given reference) specified in the element must occur at least n times in direct succession. All n occurrences of these characters will be assigned to this element.

"n" is a one-digit number and indicates the desired minimum frequency. To indicate that the characters corresponding to the element may be missing from the text entirely, n = 0 must be given. The digit "0" may be omitted if this does not lead to any confusion (i.e. unless a digit directly follows this frequency condition).

If the minimum frequency condition is not specified, "1" is assumed as the value for the minimum frequency.

<>n Maximum frequency condition: If an element contains the frequency condition "<>n", and if in the text to be searched the characters which correspond to the characters (or to the given group identification or to the given reference) specified in the element occur more than once in direct succession, up to n occurrences of these characters will be assigned to this element.

"n" is a one-digit number and indicates the desired maximum frequency. To indicate that the characters corresponding to the element may occur in the text in direct succession any number of times, n = 0 must be given. This digit "0" may be omitted if this does not lead to any confusion (i.e. unless a digit directly follows this frequency condition).

If the maximum frequency condition is not given, the maximum frequency will be same as the minumum frequency indicated with "><n", or, if the minimum frequency "><0" or no minimum frequency has been given, 1 will be assumed as the value for the maximum frequency.

Only one frequency condition may be given at a time for the minimum frequency and for the maximum frequency of a character, a group identification or a reference. However, the frequency conditions ><n and <>n may be combined; e.g. "><<> " means: no blank or any number of blanks. If the same value, other than 0, is to be specified for the minimum and the maximum frequency, only the minimum frequency needs to be specified. If different values are specified for n, the order in which they are given has the following meaning:

If the minimum frequency condition precedes the maximum frequency condition (or if only the maximum frequency condition has been given), a scan of the text attempts as soon as possible to procede from the element containing the frequency condition to the next element of the search string in order to find a string in the text which matches the search string and which also matches the elements of the search string which immediately follow (up to an element for which the specified minimum frequency differs from the specified maximum frequency).

If the maximum frequency condition precedes the minimum frequency condition, a scan of the text will attempt to validate characters of the text as matching the element containing the frequency conditions until the specified maximum frequency is reached.

As described above, the longest string corresponding to a search string is given precedence when the text is scanned. If the minimum frequency "><n" is specified for an element, for the calculation of the length of the string, this element is assumed to consist of n characters.

Examples:

"/A<>*B/" is equivalent to "/A<>0*B/" and defines a search string consisting of "a", any number of "*" (but at least one) and one "b", in this order.

"/A><<>*B/" is equivalent to "/A><0<>0*B/" and defines a search string consisting of "a", any number of "*" (which also may be missing) and one "b", in this order.

Assume a text contains the string "1230000". For the search string "><1<>5>/0" (and for the equivalent search string "<>5>/0") the corresponding text string would be "1230", whereas for the the search string "<>5><1>/0", the corresponding string would be "123000".

Assume a text contains the string "abxdxyzxyz.". For the search strings "><2<>4</XYZ", "><2<>5</XYZ" etc. up to "><2<>9</XYZ", the corresponding text string would be "abxdxyz", whereas nothing in this text would correspond to the search strings "<>9><2</XYZ" and "<>8><2</XYZ". For the search string "<>7><2</XYZ" the corresponding text string would be "abxdxyzxyz", for the search string "<>6><2</XYZ" the corresponding text string would be "bxdxyzxyz", for the search string "<>5><2</XYZ" the corresponding text string would be "xdxyzxyz", and for the search string "<>4><2</XYZ" the coresponding text string would be "abxdxyz".

References (in search strings)

>=nn Refers to the nn-th element of the search string: the characters in the text to be scanned which correspond to the nn-th element of the search string must also occur at this location in the text. A distinction is made between lower case and upper case letters. It is only possible to refer to an element located to the left of the reference.

For "nn", a two-digit number (numbers less than 10 with a leading zero) is to be entered.

>:nn is the same as >=nn, yet with no distinction being made between uppercase and lowercase letters.

<=nn Refers to the nn-last element of the search string: the characters in the text to be scanned which correspond to the nn-last element of the search string (as counted from the end of the string) must also occur at this location in the text. A distinction will be made between upper case and lower case letters. It is only possible to refer to an element located to the left of the reference.

For "nn", a two-digit number (numbers less than 10 with a leading zero) is to be entered.

<:nn is the same as <=nn, yet with no distinction being made between uppercase and lowercase letters.

Examples:

"|</>/>=01|" is equivalent to "|</>/<=03|" and indicates a search string consisting of three characters and containing two identical letters to the right and left of a digit (e.g a2a, but not A2a or a2A); it must consist of any letter ("</"), a digit (">/") and an additional character which is identical to the first (">=01") or the third-to-the-last ("<=03") character of the string in the text, i.e. the character must be the same as the one preceding the digit.

"|<>>/:>=01|" indicates a search string containing two identical strings of digits of any length to the right and left of the colon (e.g. 123:123, but also 23:23 in the string 123:234). The string consists of any number ("<>") of digits (">/"), a colon (":") and the same string of digits as the one preceding the colon (">=01"), i.e. the string consists of the same characters which correspond to the first element of the search string in the text to be scanned.

Ambient conditions

Ambient conditions make it possible to specify the surroundings in which a string must occur in the text to be scanned. A search string for which ambient conditions have been specified consists of a "core string" (which corresponds to a search string with no ambient conditions) and a left and/or right "margin string". In the text to be scanned, the string corresponding to the left margin string must directly precede the text string corresponding to the core string. The string corresponding to

the right margin string must directly follow the text string corresponding to the core string.

The left margin string is specified in front of the core string and is separated from the core string by the "delimiter for the left margin", which is described below; the right margin string is specified after the string and is separated from the core string by the "delimiter for the right margin", which is described below.

Margin strings are specified according to the same rules which apply to search strings without margin conditions; for references, it should be taken into account that when elements are counted, the elements of the margin strings will be counted as well; however, it is only possible to refer to elements of the core string.

><x delimiter for the left margin

<>x delimiter for the right margin

For "x", the delimiter chosen for the entire table (= the character entered in column 11 of the parameter line) must be used.

Examples:

" >/><|123|123<>|>/ 123|" defines a string search table with which the number 123 is to be searched, and where, for example, the numerical sequence 123 in the number 1234 is to be ignored in the search. The chosen delimiter used here is "|". Thus, the delimiter for the left margin is the string "><|" and for the right margin the string "<>|". For the specification "any number" the internally-defined string having the string group identification ">/" is used. The two delimiter characters entered in succession at the beginning of the condition " " are used to switch to exception strings. The first exception string is ">/><|123", stipulating that 123 is to be excepted from the search if another digit is located immediately to its left. The next delimiter is followed by the second exception string "123<>|", stipulating that 123 is to be excepted from the search if another digit is located immediately to its right. There then follows two successive delimiters, which are used to switch from exceptions strings to search strings. These are followed by the actual search string "123" and the concluding delimiter character. If this exception string were written after the search string (i.e. "|123 >/><|123|123<>|>/|"), this would result in all numerical sequences of "123" being found, with the exception string condition remaining ineffective. This is because it has the same length (the characters given for the margin strings are not counted) as the search string and thus receives no priority, since the priority of strings of equal length is assigned to the order in which they were entered. Note: A shorter character search table having the same effect would be "|123 ><4<>>/|".

"|>< ><|AND<>|>< |" defines a search string with which all the strings "and" are searched that occur after a blank or at the left margin and, at the same time, in front of a blank or at the

right margin of the text to be scanned. The specification is to be read as follows: "|" = delimiter freely chosen for the string search table, ">< " = one or no blank, "><|" = separator for the left margin of the core string (together with the preceding specification, this means: a blank must directly precede the core string, or if it is missing, no other characters are allowed at this left margin; in that case, the left margin must correspond to the beginning of the text to be scanned), "AND" = string to be searched, "<>|" = delimiter for the right margin of the core string, ">< " = one or no blank (together with the preceding specification, this means: a blank must directly follow the string which is searched, or if it is missing, no other characters are allowed at this right margin; in that case, the right margin must correspond to the end of the text to be scanned).

Parameter type X: Character string pair table

Specified in the information field are string pairs whose first string is the search string, which is to be replaced by the pair's second string, the substitution string. Search and substitution strings may be of different length. They are separated from each other by a delimiter of the user's choice. The delimiter is the first character (column 11) in the parameter's information field.

If one line is too small for the parameter's data, continuation lines (with the same parameter identification and same delimiter character, cf. page 244) may be used.

Besides string pairs, exception strings (= search strings to be skipped on replacing, for which therefore no substitution strings exist) may also be included. To switch from string pairs to exception strings, an additional delimiter is to be entered at the location where a search string would normally be entered. Then, one or more exception strings may be specified. To switch back from exception strings to string pairs, an additional delimiter must again be entered at the location where an exception string would be expected, and string pairs may again be entered. By using this procedure, it is possible to switch between string pairs and exception strings as often as desired. The last string must also be followed by a delimiter.

For the specification of the search strings and the exception strings, the same rules are valid as for the string search table (parameter type IX).

NEW: In substitution strings, uppercase and lowercase letters will be interpreted as such.

OLD: In substitution strings, letters are interpreted as lowercase letters, unless they are individually marked by a preceding "<". It is possible to specifiy lowercase letters by ">", but this does not have any effect.

References (in substitution strings)

Instead of specifying single characters in substitution strings, it is also possible to include references to elements of the corresponding search string (including any specified margin strings), if the characters found in the text which correspond to an element of the search string are to be substituted at this position during a search and replace operation.

>=nn Reference to the nn-th element of the search string: characters corresponding to the nn-th element of the search string will be inserted in the text unchanged at this position during a search and replace operation. The reference may refer to an element of the core string or to an element of the margin string. "nn" represents a two-digit number (numbers less than 10 are written with a leading zero).

<=nn Reference to the nn-th last element of the search string: characters corresponding to to the nn-last element of the search string (counting from the end of the string) will be inserted in the text unchanged at this position during a search and replace operation. The reference may refer to an element of the core string or to an element of the margin string. "nn" represents a two-digit number (numbers less than 10 are written with a leading zero).

>+nn as in >=nn; but during replacement all lowercase letters which correspond to the specified element of the search string will be changed to uppercase letters.

<+nn as in <=nn; but during replacement all lowercase letters which correspond to the specified element of the search string will be changed to uppercase letters.

>-nn as in >=nn; but during replacement all uppercase letters which correspond to the specified element of the search string will be changed to lowercase letters.

<-nn as in <=nn; but during replacement all uppercase letters which correspond to the specified element of the search string will be changed to lowercase letters.

Parameter type XI: Miscellaneous

The information field contains specifications in a format described in the respective program.

If one line is not sufficient for the parameter card, continuation lines (having the same parameter identification) may be used. A blank is always added to the end of a line.

Selecting and elminating text parts

One of the most common operations used in TUSTEP is the selection or elimination of specific parts of a text unit (= running text up to 32000 characters long). This is done with the help of unique markers contained in the text. These markers can take the form of a ("beginning marker") and an end "end marker" to mark the beginning and end of the text part to be selected or eliminated. Or they can be interpreted as a pair of brackets, the the text enclosed in each pair of brackets being either selected or eliminated.

These two methods of marking text for selection or elimination may also be combined. In this case the text parts marked by beginning and end markers are first selected. Then each selected text part can be further processed based on any brackets it may contain.

In some programs, this process of selecting or eliminating text can be applied to the same text unit repeatedly, with each selected text part being joined to a new text unit. Individual text parts can therefore be moved and arranged in the desired order.

Selecting or eliminating text using beginning and end markers

The beginning and end markers for text parts to be either selected or eliminated are specified in separate parameters. In the following, the parameter for the beginning marker will be called A, that for the end parameter will be called. E. More than one character string may be specified as a marker for each of these two parameters, with each character string being equal in value.

An index can also be set up with an additional parameter. This index can be used to determine whether the marked text part is to be selected or eliminated, and whether the marker itself is to be included in the selection or elimination procedure. In the following, the parameter for this index will be called AEI. The index is a numerical specification having the following definitions:

1 = Selects the first text part marked by parameters A and E (starting with a beginning marker and ending with either the next end marker or the end of the text unit). The rest will be eliminated.

    If only parameter A has been specified, the selected text part ends at the end of the text unit. If only parameter E has been specified, the selected text part starts at the beginning of the text unit.

0 = Eliminates that part of the text unit that would be selected with the value 1 (preceding definition).

3 = As with 1, but selects all text parts (instead of just the first) marked by character strings of parameters A and E. (The second text part starts with the beginning marker that follows the end of the first marked text part.) The rest will be eliminated.

If only parameter A has been specified, the selected text parts starts with the last beginning marker occuring in the text unit and ends at the end of the text unit. If only parameter E has been specified, the selected text part starts at the beginning of the text unit and ends before the last end marker occuring in the text unit.

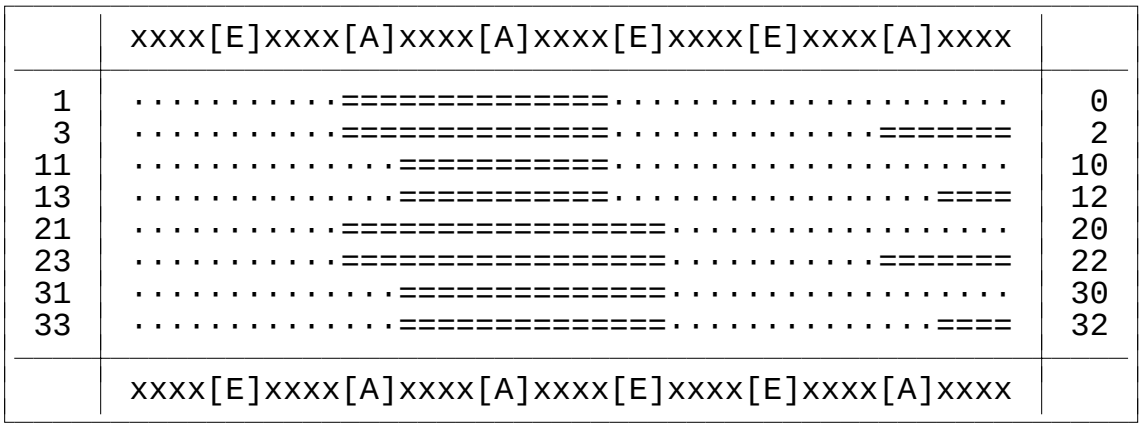
2 = Eliminates that part of the text unit tha would be selected by the value 3 (preceding definition).

For the values 0 to 3, each beginning marker will be regarded as part of the following text, while the end marker is no longer considered part of the text preceding it. This rule for beginning and end markers can be reversed by adding 10 and/or 20. When 10 is added to the above values (i.e. specifying a value from 10 to 13), a beginning marker will no longer be considered part of the text following it. When 20 is added, each end marker will be considered part of the text preceding it. When 30 is added, beginning markers will not be considered part of the text following them but end markers will be considered part of the text preceding them.

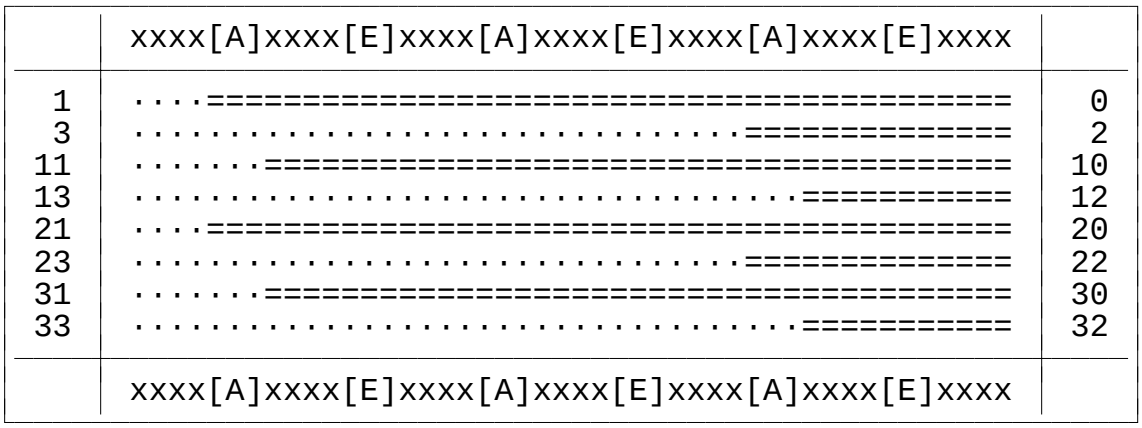
For the values 2 and 3, the search for the next beginning marker commences at the first position following the most recently found end marker, as it is not considered to be part of the text preceding it. Beginning and end markers may therefore overlap in the text. If 20 or 30 has been added to these values, the search for the next beginning marker will commence at the position following the last position of the most recently found end marker, as this is no longer considered to be part of the preceding text part.

The following tables provide a schematic illustration of how different index definitions affect the selection and elimination of text parts. The upper and bottom line of each diagram represents the text from which the text parts are to be either selected or eliminated. An "[A]" stands for a beginning marker specified with parameter A, and "[E]" stands for an end marker specified with parameter E. Each of the lines between the top and bottom lines are flanked by two index values which can be specified with parameter AEI. One value is located in the left column and the other in the right. If parameter AEI has been used to specify the value in the left column, those parts of the middle column line represented by a "=" will be selected, while those represented by a "." will be eliminated. If the value in the right column is has been specified, the "=" text parts will be eliminated and the "." parts will be selected.

1. If both parameters A and E have been specified:



2. If only parameter A has been specified:



```
    | 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. If only parameter E has been specified:

```
    | 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 |
```

Selecting and eliminating using parentheses

The markers serving as opened and closed parentheses are specified in separate parameters. In the following, the parameter for opened parentheses will be known as KLA; the parameter for closed parehtheses will be known as KLZ. More than one string can be specified as "parentheses" in each of these two parameters, with each string of a parameter being regarded as equal in value.

An index can be specified with a further parameter. The index is used to determine whether the text part in "parentheses" to to be selected or eliminated, as well as whether the marker itself is to be included in the selection or elimination process. In the following, the parameter fir the index will be called KLI. The index required numerical specifications defined as follows.

0 = Eliminates the text parts in parentheses (including the parentheses themselves). Any missing parentheses will be supplied in their logical position at either the beginning or end of the text unit.

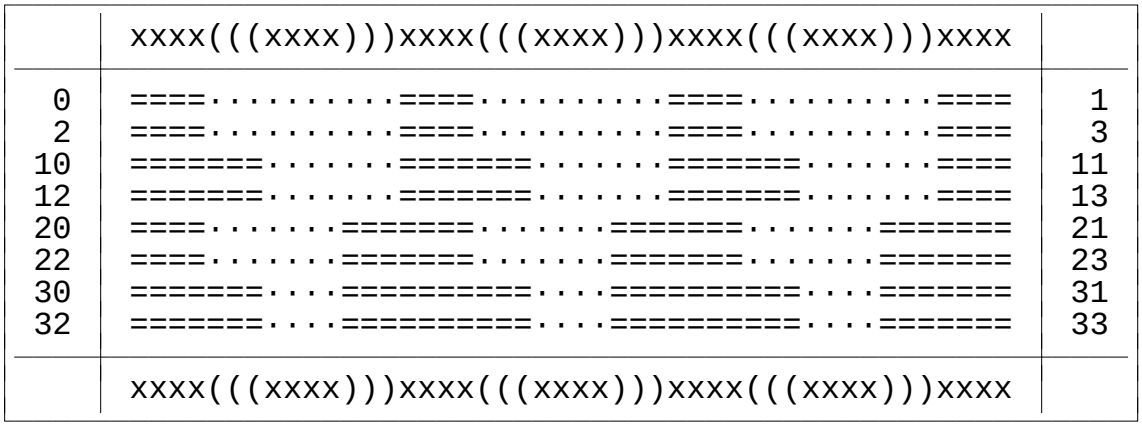
1 = Selects parts that would be eliminated by the value 0 (preceding definition)

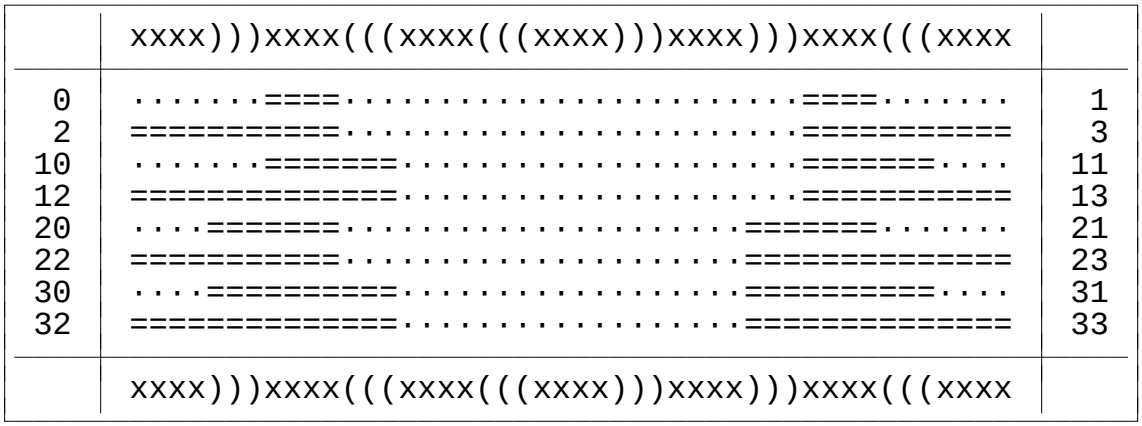
2 = Same as 0, but any missing parentheses will not be supplied at their logical position; unpaired parentheses will be ignored.

3 = Selects parts that would be elimined with 2.

For the values 0 to 3, each set of parentheses are considered part of the text they enclose; they will be either eliminated or selected along with this text. This rule can be reversed for opened and/or closed parentheses by adding 10 and/or 20 to the specified value. By adding 10 (i.e. specifying the values 10 to 13), the opened parentheses will not be considered part of the enclosed text. If 20 is added, the closed parentheses will not be considered part of the enclosed text. If 30 is added, neither of the parthensis pair will be considered part of the enclosed text.

The following tables provide a schematic illustration of how the index affects the selection and elimination of text parts. The upper and bottom line of each diagram represents the text from which the text parts are to be either selected or eliminated. "(((" stands for a opened parenthesis specified with parameter KLA, and ")))" stands for a closed parenthesis specified with parameter KLE. Each of the lines between the top and bottom lines are flanked by two index values which can be specified with parameter KLI. One value is located in the left column and the other in the right. If parameter AEI has been used to specify the value in the left column, those parts of the middle column line represented by a "=" will be selected, while those represented by a "." will be eliminated. If the value in the right column is has been specified, the "=" text parts will be eliminated and the "." parts will be selected.


xxxx(((xxxx)))xxxx(((xxxx)))xxxx(((xxxx)))xxxx
0 2 10 12 20 22 30 32
1 3 11 13 21 23 31 33
xxxx(((xxxx)))xxxx(((xxxx)))xxxx(((xxxx)))xxxx


xxxx)))xxxx(((xxxx(((xxxx)))xxxx)))xxxx(((xxxx
0 2 10 12 20 22 30 32
1 3 11 13 21 23 31 33
xxxx)))xxxx(((xxxx(((xxxx)))xxxx)))xxxx(((xxxx

M a c r o s

Survey:

General notes

By using macros, the TUSTEP user may define commands of his own. A macro is a sequence of macro instructions and TUSTEP commands (if necessary, including parameters) which is recorded in a "macro file" (cf. page 20). On calling a macro, a command sequence is composed which (depending on the macro instructions) is then normally executed.

In the macro, lines containing two dollar signs in the first two columns are interpreted as macro instructions. Lines without this instruction marker are inserted into the command sequence to be composed, unless they are skipped according to a preceding macro instruction or unless they are part of the definition of an asterisk variable (see below).

In order to make macros more readable, it is possible to insert blanks after column 3 in macro instruction. Blanks are significant only in strings enclosed by quotation marks (") and in messages. All other blanks in marco instructions will be ignored.

Calling macros

The macro-call has the same structure as a TUSTEP command (see "General information" concerning commands, page 55); a dollar sign and the macro name (#$Macroname, ...) is given instead of the command name. The dollar sign distinguishes macros from commands. In contrast to command names, macro names may not be abbreviated. The macro also defines whether specifications are possible, and if so, which ones can be given.

On calling a macro, a check is first made as to whether the macro has been recorded (generally, by the appropriate Load instruction) in the file ***most recently*** proocessed with the Editor. In this case, the macro is read from this file; otherwise, it is read from the macro file. This makes it possible to test a macro without having to copy the macro (with the Unload instruction) back into the macro file each time.

Before macros can be called, the command #DEFINE (see page 83) must first be used to specify the file which contains the macros.

Macro variables

Macro variables are used to represent a character string. This string is known as the variable value or variable contents. Wherever the name of a macro variable in pointed brackets occurs in a macro, this name (including the pointed brackets) will be replaced by the current value of the macro variable. The name of a macro variable may consist of 1 to 12 characters (letters, digits and " ", but must begin with a letter.

A macro variable is defined by assigning it a value (i.e. a character string) using a macro instruction. After being so defined, it can be used within macros, remaining valid only until the macro has been run. The next time a macro is called, the contents of the variable can no longer be accessed.

However, variables can also be defined independent of macros. These are called TUSTEP variables and can be defined with the command #DEFINE (see page 85). Each variable so defined remains valid until it is either redefined or the TUSTEP session is terminated. A TUSTEP variable can also be defined using a macro instruction and the contents of such a TUSTEP variable can be accessed within a macro.

The "asterisk variables" are a special kind of macro variable. They represent zero, one, or more than one string. When modifying a line of the macro using asterisk variables, the following must be observed: in a line containing a macro instruction (i.e., a line which begins with $$), the names of the asterisk variables (including the brackets) will be replaced by a "*". Lines not beginning with $$ are inserted into the command sequence n-times, where n is the number of strings represented by the asterisk variable; the name of the asterisk variable (including the brackets) is replaced each time by one string of the asterisk variable. Such a line of the macro may not contain more than one asterisk variable.

In addition, integer variables can be used in arithmetic operations. Having a name from I(0) to I(99), these variables may only contain numerical values. Similar to the value (contents) of a macro variable, the results of arithmetic calculations stored in these variables can be inserted at any position by placing these integer variables in pointed brackets.

Example:

Let's say the following variables have been defined:

1. The macro variable "name" contains the string "PAR".
2. The integer variable I(1) contains the value 2.
3. The asterisk variable "words" contains the three strings "but", "or", "and".

The line

```
     Example: <name>   <I(1)>   /<word>/
```

would produce the following line after the variables have been replaced:

```
     Example: PAR   2   /but/
     Example: PAR   2   /or/
     Example: PAR   2   /and/
```

Macro instructions

Specifications

Macro variables can be assigned current values when the macro is called. To do so, the respective macro variables must be defined as specifications with the following macro instruction.

```
$$! Specificationname1, Specificationname2, ...
```

The names of the macro variables so defined are identical to the specification names. If specification values are given when calling the macro, these values will be assigned to the respective macro variables. If no value is given for a specification when calling the macro, an empty string is assigned to the respective macro variable. If in this case a different string is to be assigned instead of an empty string, this may be indicated by using the following form of the above macro instruction:

```
$$! Specificationname1=Specificationvalue1, ...
```

In this way, default values can be defined for the specifications.

The definition of the specifications can take up more than one line. These lines must follow in direct succession. However, the definition for each individual specification must be written in a single line, i.e. it cannot occupy different lines.

```
$$! Specificationname1, Specificationname2
$$! Specificationname3
```

The definition of specifications must be made at the beginning of the macro. Only comment lines marked by "$$-" (see below) may precede them.

Assignments

A macro variable can also be defined and assigned a value with the following macro instruction:

```
$$: Macrovariablename = "String"
```

The string, written between the quotation marks and which may not contain any quotation marks itself, is assigned to the macro variable with the given name. This instruction may also be used in order to assign a new value to a macro variable defined previously.

This macro instruction has two alternatives:

```
$$: Macrovariablename = #"String"
```

assigns a number to the macro variable specified. It indicates the number of substrings which make up the given string. Substrings are parts of a string which are separated from each other by an apostrophe. A substring may be empty.

```
$$: Macrovariablename = ##"String"
```

also assigns a number to the given macro variable. It indicates the number of characters which make up the given string.

An asterisk variable can be defined and assigned a value with the following macro:

```
$$: Macrovariablename = *
```

The strings entered between this macro instruction and the following macro instruction (= next line beginning with $$) are assigned to the macro variable with the given name. Each line represents a string. They may contain macro variables excluding asterisk variables. This instruction may also be used to assign a new value to a macro variable defined previously.

With the command #DEFINE (see page 83) TUSTEP variables can be defined and given a preassigned value. If the value of such a variable is to be used in a macro, it must be made accessible with the following macro instruction:

```
$$; Macrovariablename = "String"
```

If a TUSTEP variable having the same name has already been defined with the command #DEFINE (or with the macro instruction DEFINE outlined below), its value will be assigned to the macro variable. Otherwise, the string given in the instruction is assigned to the macro variable.

The following macro instruction can also be used to define a TUSTEP variable within a macro:

```
$$ DEFINE Macrovariablenname
```

Here the contents of the specified macro variable will be copied to a TUSTEP variable having the same name. If the contents of the macro variable are subsequently altered, this will have no effect on the TUSTEP variable of the same name.

Inquiries

With the following macro instruction, a macro variable can be defined and assigned a value. In interactive mode, a message will be displayed on screen with a response prompt:

```
$$? "Message", Macrovariablename = "String"
```

The string entered as the response (one line maximum) is assigned to the given macro variable. If an empty response is entered or if the macro is called during a batch job, the string given in the instruction itself is assigned to the macro variable. This instruction may also be used to assign a new value to a macro variable defined previously.

Entering data

The following macro instruction is used to define an asterisk variable. Data entered after the macro is called will be read and assigned to the asterisk variable whose name is given in the macro instruction:

```
$$? "Message", Macrovariablename = *
```

The data must be ended with "*EOF". Whenever this type of macro instruction is processed, the each data set that follows will be read up to the next "*EOF".

If the data are entered at the display device, they will be requested with the message given in the macro instruction.

File information

With the following macro instructions, a macro variable can be defined and assigned a numerical value. This can be used to establish the following sizes of an opened TUSTEP file (but not a system file):

- number of records

```
$$: Macrovariablename = RECORDS "filename"
```

- length of the shortest record

```
$$: Macrovariablename = MINLEN "filename"
```

  Limitation: if the shortest record of a file is deleted or lengthened in the Editor, the length of the resulting shortest record will not be reestablished. In this case, the old (incorrect) value will be retained by the macro variable.

- average record length

  $$: Macrovariablename = RECLEN "filename"

- length of the longest record

  $$: Macrovariablename = MAXLEN "filename"

  Limitation: if the longest record of a file is deleted or shortened in the Editor, the length of the resulting longest record will not be reestablished. In this case, the old (incorrect) value will be retained by the macro variable.

- page number of the first record

  $$: Macrovariablenname = FIRSTPAGE "filename"

- page number of the last record

  $$: Macrovariablenname = LASTPAGE "filename"

- number of pages (more precisely: page number of the last record - page number of the first record + 1)

  $$: Macrovariablename = PAGES "filename"

- file size in bytes

  $$: Macrovariablename = BYTES "filename"

These instructions can also be used to assign new values to macro variables already defined.

Miscellaneous information

The following macro instructions are used to define a macro variable and assign a string to it.

- Date written as xx.xx.xx (e.g. 12.01.90)

  $$: Macrovariablenname = DATE_1

- Date written as xx. xxx. xxxx (e.g. 12. Jan. 1990)

  $$: Macrovariablenname = DATE_2

- Date written as xx. xxxxxxx xxxx (e.g. 12. January 1990)

  $$: Makrovariablenname = DATE_3

- Time written as xx:xx (e.g. 12:00)

  $$: Macrovariablenname = TIME

- Host computer name

```
    $$: Macrovariablenname = HOST

- User identification

    $$: Macrovariablenname = USER

- Current project name

    $$: Macrovariablenname = PROJECT
```

These instructions can also be used to assign a new value to a macro variable already defined.

Files: opening, closing, creating, deleting, renaming

With the following macro instruction, files can be opened, closed, created, deleted or renamed while a macro is being processed. This is carried out by calling the corresponding TUSTEP command:

```
    $$ EXECUTE #command
```

One of the following commands is used for #command,

```
    #CLOSE, ...   (described on page 64)
    #RENAME, ...   (described on page 163)
    #OPEN, ...   (described on page 151)
    #CREATE, ...     (described on page 79)
    #ERASE, ...   (described on page 93)
```

Command names may not be abbreviated. Otherwise, there are no limitations concerning these commands.

Commands generated by macros are not executed until the entire macro has been processed. However, with this macro instruction the commands will be executed immediately.

To obtain, for example, information about a file (see below), the file must first be opened. If it is not opened (this can ascertained with a macro instruction, see below), it must first be opened with the EXECUTE instruction. An OPEN command used without this instruction will not produce the desired result, since the file will be opened only after the macro has been processed.

Loops

With the following macro instructions it is possible to evaluate one or more successive lines more than once. There are three kinds of loops:

```
    $$ LOOP Macrovariablename = "String"
    ...
    $$ ENDLOOP
```

The lines entered between LOOP and ENDLOOP are processed for each substring which contains the given string. At the beginning of the loop the current substring will be assigned to the macro variable specified. Substrings are parts of a string which are separated from each other by an apostrophe. A substring may be empty.

```
$$ LOOP Macrovariablename=Initialvalue,Endvalue,Stepsize
...
$$ ENDLOOP
```

The lines entered between LOOP and ENDLOOP are processed for each number of the loop counter. At each start of the loop, the contents of the loop counter are assigned to the specified macro variable. The loop counter runs through the values from the specified initial value up to the specified end value in the given step size. The step size (including the preceding comma) may be omitted. In this case, 1 is the default step size.

```
$$ LOOP
...
$$ ENDLOOP
```

The lines entered between LOOP and ENDLOOP are repeatedly processed until the loop is exited with the macro instruction

```
$$ EXIT
```

This macro instruction may be used for all three kinds of loops in order to terminate the processing of the loop prematurely. The processing of the macro continues after ENDLOOP.

Arithmetic operations

Arithmetic operations can be executed in the macro instruction

```
$$# Computestatements
```

If one line is not sufficient for the compute statements, they can be distributed over several lines (each line must begin with $$#). The compute statements are executed after the names of the macro variables are replaced by their assigned values. As a consequence, the value of macro variables cannot be changed by compute statements.

Computed numerical values which are to be used outside compute statements must be stored in the integer variables I(0) to I(99). The numerical values of these integer variables can be inserted at any place in the same manner as the values of macro variables. In this case, the name of the respective integer variable is entered in pointed brackets instead of the name of the macro variable; it is to be written as I(n), where n is any number from 0 to 99.

Compute statements are described on page 283.

Selection

With the following macro instructions, it is possible to have one or more successive lines interpreted only under certain conditions:

```
$$ IF (condition 1) THEN
...
$$ ELSEIF (condition 2) THEN
...
$$ ELSEIF (condition 3) THEN
...
...
...
$$ ELSEIF (condition n) THEN
...
$$ ELSE
...
$$ ENDIF
```

The conditions are checked in succession. If a condition is true, the following lines will be interpreted up to the next ELSEIF or ELSE. All other lines between IF and ENDIF are skipped. If no condition is true, the lines between ELSE and ENDIF will be evaluated.

The number of ELSEIF instructions depends on the number of conditions which are to be checked. If only a single condition is to be checked, the ELSEIF instructions are not necessary. Similarly, the ELSE instruction is not necessary if there are no further lines (between IF and ENDIF) which are to be evaluated in case none of the conditions are true.

The conditions are described on page 280.

Messages

With the macro instruction

```
$$+ Message
```

messages may be listed into the journal (i.e. in interactive mode, the screen) during the processing of the macro.

For messages to be listed only when the macro is called in interactive mode, the following macro instruction may be used:

```
$$* Message
```

With this macro instruction, information concerning user responses and their effect can be listed.

Comments

With the macro instruction

```
$$- Comment
```

comments may be inserted in a macro. They are ignored when the macro is processed.

A comment given directly at the beginning of a macro is interpreted as a description of the macro and can be listed with the command #INFORM (cf. page 111).

Surpressing command execution

Normally, the command sequence which has been compiled by the macro-call is then executed. If execution is to be surpressed because the macro e.g. has been called with invalid specification values, this can be achieved by setting the error flag with the following macro instruction:

```
$$ ERROR
```

By using this macro instruction, the processing of the macro is not aborted. However, loops will be run only once after the error flag has been set.

Terminating macro processing

With the macro instruction

```
$$ STOP
```

the processing of the macro can be terminated prematurely. It does not set the error flag.

Defining control characters

There are three control characters which may be changed: the dollar sign (with which the macro instructions are marked in the first two columns of a line) and the pointed brackets (which enclose the name of a macro variable at places where the contents of the macro variables are to be inserted). After the macro instruction

```
$$= * [ ]
```

the asterisk and the square brackets are valid as control characters. Instead of the asterisk (as here in the example) any special character is allowed; instead of square brackets,

parentheses, pointed brackets or braces may be used. After having changed the control characters with the instruction described above, the default control characters may be set again with the following instruction:

```
**= $ < >
```

If only the control character which marks the macro instruction is to be changed, pointed brackets may be omitted in the macro instruction.

Conditions

The following describes possible conditions for the Selection instruction. Number-1 and Number-2 represent a number (= string of digits) with or without a sign; Cstr, Cstr-1, Cstr-2, ... represent any character string. Normally, the name of the macro variable enclosed in pointed brackets is entered instead of the number or the string to be checked. The validy of a condition is checked after the names of the macro variables have been replaced by their assigned values. No distinction is made between upper and lower case letters. They are treated equally.

Comparing strings

"Cstr-1" .EQ. "Cstr-2"	Cstr-1 and Cstr-2 are equal
"Cstr-1" .EQ. "Cstr-2","Cstr-3",...	Cstr-1 corresponds to at least one of the strings Cstr-2, Cstr-3, ...
"Cstr-1" .NE. "Cstr-2"	Cstr-1 and Cstr-2 are not equal
"Cstr-1" .NE. "Cstr-2","Cstr-3",...	Cstr-1 does not correspond to any of the strings Cstr-2, Cstr-3, ...
"Cstr-1" .LT. "Cstr-2"	Cstr-1 precedes Cstr-2 in alphabetical order.
"Cstr-1" .GT. "Cstr-2"	Cstr-1 follows Cstr-2 in alphabetical order.
"Cstr-1" .LE. "Cstr-2"	Cstr-1 and Cstr-2 are equal, or Cstr-1 precedes Cstr-2 in alphabetical order.
"Cstr-1" .GE. "Cstr-2"	Cstr-1 and Cstr-2 are equal, or Cstr-1 follows Cstr-2 in alphabetical order.
"Cstr-1" .AB. "Cstr-2"	Cstr-1 and Cstr-2 are equal, or Cstr-1 is an abbreviation of Cstr-2. An abbreviation period (e.g. "Cstr.") which may occur in Cstr-1 is ignored.
"Cstr-1" .AB. "Cstr-2","Cstr-3",...	Cstr-1 corresponds to at least one of the strings Cstr-2, Cstr-3, ... , or Cstr-1 is an abbreviation of at least one of the strings Cstr-2, Cstr-3, ... An abbreviation period that may occur in Cstr-1 is ignored.

Checking strings

In the following conditions, .NE. may be given instead of .EQ. A condition containing .NE. is true if it is false when it contains .EQ. and vice versa.

"Cstr" .EQ. NUMBER	Cstr is a number with or without a sign.
"Cstr" .EQ. NAME	Cstr consists of 1 to 12 letters and digits and begins with a letter.
"Zflg" .EQ. PROJECTNAME	Cstr is a valid project name (regardless of whether a project with this name already exists).
"Zflg" .EQ. PROJECT	Cstr is the name of an existing project (regardless of whether it has existing files).
"Cstr" .EQ. FILENAME	Cstr is a valid file name (regardless of whether a file with this name exists).
"Cstr" .EQ. FILE	Cstr is the name of an existing file file (regardless of whether the file with this name is opened).
"Cstr" .EQ. READ	Cstr is the name of a file which has been opened for reading or for writing (and thus also capable of being read)
"Cstr" .EQ. WRITE	Cstr is the name of a file which has been opened for writing.
"Cstr" .EQ. SCRATCH	Cstr is the name of a scratch file (temporary file).
"Cstr" .EQ. EMPTY	Cstr is the name of an empty file which has been opened.
"Cstr" .EQ. SEQ	Cstr is the name of a file of the type SEQ which has been opened.
"Cstr" .EQ. RAN	Cstr is the name of a file of the type RAN which has been opened.
"Cstr" .EQ. SDF	Cstr is the name of a file of the type SDF which has been opened.

Comparing numbers

Number-1 .EQ. Number-2	Number-1 is equal to Number-2
Number-1 .NE. Number-2	Number-1 is not equal to Number-2
Number-1 .LT. Number-2	Number-1 is less than Number-2
Number-1 .GT. Number-2	Number-1 is greater than Number-2
Number-1 .LE. Number-2	Number-1 is less than or equal to Number-2
Number-1 .GE. Number-2	Number-1 is greater than or equal to Number-2

Inquiring selection switches etc.

+SWn	Sense switch n (n = 1 to 7) is set.
-SWn	Sense switch n (n = 1 to 7) is cleared.
BATCH	Macro was called during a batch job.
DIALOG	Macro was called in interactive mode.
ERROR	Macro instruction ERROR was already executed.
ERRORSTOP	Error stop is set.
PARAMETER	Parameter logging is activated.
PARAMETER_OLD	Parameter mode is set to OLD.
PARAMETER_NEW	Parameter mode is set to NEW.
JOURNAL	Journal file is active.

Connecting conditions

Instead of one condition, several conditions may also be specified which are connected by .AND. (logical AND) or .OR. (logical OR). When the condition is processed, the logical AND is given precedence over the logical OR. Additional brackets are not allowed.

Compute statements

In the following, the term "instruction" always refers to a compute statement which is part of a macro instruction beginning with $$#. It should not be confused with a macro instruction (which may look similar).

The individual compute statements must be separated from each other by a semicolon. Blanks are insignificant and may be inserted at any place to make the compute statements more legible. A compute statement may also be interrupted at any place and continued in the next line.

To start with, a few terms will be defined which are used in defining various compute statements.

Integer variables

An integer variable is a storage location which is identified by a name and contains an integer value. Each of these integer variables has a unique name by which the contents of the variable can be accessed. This access is made by placing the name of the integer variable at the location where its numerical value is defined (cf. "Value assignment") or is required (cf. "Arithmetic expression"). The names of the integer variables are predefined and cannot be freely chosen. At the start, all integer variables have the value zero. The largest numerical value that can be saved to a variable depends on the type of computer being used. In most cases this is a the value 2 147 483 647.

The integer variables (100 storage locations) may be addressed as indexed variables written as I(Index). The index may be any arithmetic expression (cf. below). However, the value of the arithmetic expression must be between 0 and 99 (inclusively).

Functions

In order to support calculations, certain functions are available to the user. A function is called with its name and one or more arguments. Arguments are separated from each other by a comma and are entered in brackets after the function name. They may take the form of any arithmetic expression (cf. below). If a function with these arguments is called, a numerical value - the function value - is calculated according to the function regulations.

Available functions

(For simplicity's sake, the following examples employ numbers in the individual arguments. In actual practice, integer variables or the names of macro variables (in brackets) are used in most cases instead of numbers. Integer variables as well as the names of the macro variables will be replaced by their assigned values before the compute statements are executed.)

- Calculating the absolute value: IABS (arg)

  The function value obtained is the absolute value of arg.

  Example: The function value of IABS(-4) is 4.

- Calculating the minimum: MIN (arg1, arg2)

  The function value obtained is the smaller of the two numerical values arg1 and arg2.

  Example: The function value of MIN(-5,+3) is -5.

- Calculating the maximum: MAX (arg1, arg2)

  The function value obtained is the larger of the two numerical values arg1 und arg2.

  Example: The function value of MAX(-5,+3) is +3.

- Calculating the division remainder: MOD (arg1, arg2)

  The function value obtained is the remainder obtained when arg1 is divided by arg2.

  Example: The function value of MOD(234,10) is 4.

- Interval function: IV (arg, arg1, arg2, arg3, ...)

  This function can be used to determine the interval to which the numerical value arg belongs. A function value of 0 is obtained if arg is less than arg1; the value 1 is obtained if arg is greater than or equal to arg1 but less than arg2; the value 2 if arg is greater than or equal to arg2 but less than arg3, etc. The number of arguments is not limited to this function. However, arg1 must be less than arg2, arg2 must be less than arg3, etc.

  Example: The function value of IV(16,1,10,100,1000) is 2.

- Date function: ID (day, month, year, number, mode)

With this function it is possible to calculate and convert calendar dates. This function is controlled via the argument mode.

To facilitate the calculation of calendar dates, days are numbered in succession. Thus, each day bears a unique number, the day number.

mode=0: Current date
Input: none
Output: Current date in the arguments day, month, year and current day number in the argument number

mode=1: Calculating the day number by specifying the date
Input: Date in the arguments day, month, year
Result: Day number in the argument number

mode=2: Calculating the date by specifying the day number
Input: Day number in the argument number
Result: Date in the arguments day, month, year

mode=3: Calculating the date of Easter
Input: Year number in the argument year
Result: Date of Easter in the arguments day, month, year and the corresponding day number in the argument number

mode=4: Calculating the time interval between two calendar dates
Input: Date of the first calendar date in the arguments day, month, year and the day number of the second (later) date in the argument number
Result: The time interval between the two dates in years, months, and days (in the arguments year, month and day) as well as in days (in the argument number)

The function value so obtained - independent of the value of the argument mode - is the day of the week (1=Monday, 2=Tuesday, 3=Wednesday, 4=Thursday, 5=Friday, 6=Saturday, 7=Sunday) whose is given in the arguments day, month and year. If there is an illegal input, the function value is zero (e.g. if mode has a value which is not defined, or if an illegal date is given, such as the 29th of February in a non-leap year).

Calculations are based on the Gregorian ||| calendar; however, if a date occurs before October 15, 1583, the will be used. If the Julian calendar is to be used for later dates as well, the respective negative value is to be given as argument mode.

Example: Calculating the date of Pentecost/Whitsuntide 1984

```
I(0) = ID (I(1), I(2), 1984, I(4), 3);
I(0) = ID (I(1), I(2), I(3), I(4)+49, 2);
```

First, the date of Easter 1984 is calculated. It does not matter which numerical values I(1), I(2) and I(4) contain before the date function is called. Because 3 has been given

as the mode, only the argument year is evaluated. The day of week, day and month of Easter are stored in I(0), I(1) and I(2), but are not used for further processing. The day number of Easter is stored in I(4). Because the interval between Easter and Pentecost is exactly 7 weeks (= 49 days), the day number of Pentecost is obtained by adding 49 to the day number of Easter. This day number is then converted into the respective calender date. It is regardless which numerical values I(1), I(2) and I(3) contain before the date function is called. Because 2 has been given for the mode, only the argument number is evaluated. The day of the week (7 for sunday) is stored in in I(0); day, month and year of the date of Pentecost are stored in I(1), I(2) and I(3).

Arithmetic expressions

An arithmetic expression is a rule for computing used to define a numerical value. It consists of operands, arithmetic operators and pairs of brackets.

An operand may be an (integer) number, an integer variable or a function call. In the simplest case, an arithmetic expression consists of only one of these three operands.

The arithmetic operators for the four basic mathematical operations are:

+ Addition	* Multiplication
- Subtraction	/ Division

When the arithmetic expression is evaluated, multiplication and division are executed prior to addition and subtraction. If there are several consecutive multiplication and division operations, they are carried out from left to right. The same is true for consecutive addition and subtraction operations. Exceptions to this rule can be made by placing brackets at the appropriate positions.

Please note that division is executed with integer results only; remainders will be get lost and there is no rounding off. Thus, e.g. 3/2 results in the value 1 (not 1.5) and 3/2*4 results in the value 4 (not 6).

Relation conditions

A relation condition is where two numerical values are compared. It consists of two arithmetic expressions connected by a relation operator:

arithm. expression relation operator arithm. expression

There are six relation operators:

.EQ. equal	.NE. not equal
.GT. greater than	.LT. less than
.GE. greater or equal	.LE. less or equal

A relation condition is either satisfied, resulting in the logical value TRUE, or it is not satisfied, resulting in the logical value FALSE.

Logical expressions

A logical expression consists of relation conditions which may be connected to each other by logical operators. In the simplest case, a logical expression may consist of only one relation condition.

The logical operators are:

.AND. logical AND
.OR. logical OR

A logical expression is processed analogously to an arithmetic expression. The result is either the logical value TRUE or the logical value FALSE. When the logical value is evaluated, the logical AND is executed prior to the logical OR. This order may be changed by the appropriate use of brackets.

Value assignments

A value assignment takes the form:

Integer variable = arithmetic expression

It causes the arithmetic expression to be evaluated. The result is stored in the integer variable placed to the left of the equals sign. In this instruction, the equals sign has the function of an assignment operator and thus differs from its normal mathematical function.

Examples: I(1) = 0; I(1) = I(1) + 1; I(I(1)) = MAX (I(2),I(3));
I(1) = I(1) * (I(1) + I(2)); I(0) = MOD (I(1), 1000)

Conditional statements

A conditional statement is used to specify that one or more successive instructions are to be executed only under certain conditions. These conditions are specified by a logical expression. There are three forms of conditional statements:

1.) If a single instruction (in this case, this may be only a value assignment) is to be executed only under the conditions

specified in the logical expression, the following form may be chosen:

```
IF (logical expression) instruction
```

The instruction specified after the brackets is executed only if the logical expression in brackets has the value TRUE. Otherwise, this instruction is skipped.

Example: `IF (I(1).EQ.0) I(2) = I(2) + 1;`

2.) If the following form of the conditional statement is chosen, one or more instructions of any type may be executed on the basis of the conditions specified in the logical expression:

```
IF (logical expression) THEN;
   Instructions;
END IF
```

The instructions entered between THEN and END IF are executed only if the logical expression in brackets has the value TRUE. Otherwise, these instructions are skipped.

Example: `IF (I(1).EQ.60) THEN; I(1)=0; I(2)=I(2)+1; ENDIF;`

3.) The third form of the conditional statement is an extension of the second form. With it, two sequences of instructions may be given, one of which is executed and the other one is skipped:

```
IF (logical expression) THEN;
   Instruction;
ELSE;
   Instruction;
END IF;
```

If the logical expression in brackets has the value TRUE, the instructions located between THEN and ELSE are executed and the instructions between ELSE and END IF are skipped. If the logical expression has the value FALSE, the instructions between THEN and ELSE are skipped and the instructions between ELSE and END IF are executed.

Note

The compute statements that can be made in macros correspond to those used in the TUSTEP program COPY. However, the GO TO statement may not be used in macros.

Testing aids

If a macro is called, the following may be specified directly after the macro name:
- whether all the commands generated by the macro are to be executed, or only a certain range
- whether the generated commands are to be additionally listed/logged
- whether the generated commands are to be executed

```
#$Macro name-RANGE;LISTING;EXECUTION, ...
```

RANGE	= pos		Only the command sequence generated in the macro and starting at record position pos is to be executed.
	= pos1-pos2		Only the command sequence generated in the macro from record position pos1 to record position pos2 (inclusively) is to be generated. In this case, "pos" represents a record number whose syntax corresponds to program mode (cf. page 24).
LISTING	= -	*	No additional listing of the generated command sequence.
	= +		The generated command sequence is to be listed into the journal (in interactive mode this is the screen).
	= file		Name of the file to which the generated command sequence is to be listed.
EXECUTION	= +	*	The generated command sequence is to be executed.
	= -		The generated command sequence is not to be executed.

Examples

a) for the structure of a macro:

```
$$- Function: The macro PRINT outputs a file
$$-           to a printer.
$$-
$$- Call:     #$PRINT, FILE=..., MODE=..., PRINTER=...
$$-
$$-           Mode=F   formatting and printing
$$-           Mode=T   printing in text mode
$$-           Mode=P   printing in program mode
$$-
$$! FILE, MODE, PRINTER=HPII
$$-
$$- If no printer has been specified, HPII is used
$$-
$$- If no file name has been specified, user will provide
it
$$- (if an empty response is given, no commands will
generated)
$$-
$$ IF ("<FILE>".EQ."")  THEN
$$? "Enter file name", FILE="-"
$$ IF ("<FILE>".EQ."-")  THEN
$$ STOP
$$ ENDIF
$$ ENDIF
$$-
$$- If no mode has been specified, it will be requested from the
user
$$- (if an empty response is given, F will be assumed for mode)
$$-
$$ IF ("<MODE>".EQ."")  THEN
$$+ For mode, the following entries are valid:
$$+          F  for formatting and printing
$$+          T  for printing in text mode
$$+          P  for printing in program mode
$$+          empty entry is equivalent to F
$$? "Enter mode", MODE="F"
$$ ENDIF
$$-
$$- Generating the commands
$$-
$$ IF ("<MODE>".EQ."F")  THEN
#format,source=<FILE>,erase=+,parameter=*
dev       <PRINTER>
*eof
$$ ELSE
#glisting,source=<FILE>,mode=<MODE>,erase=+,parameter=*
dev       <PRINTER>
*eof
$$ ENDIF
#print,,<PRINTER>
```

b) for using the command #MACRO

In this example, the functions of the macro PRINT in the preceding example are enhanced.

Errors may occur whenever texts are formatted. On the one hand, they are caused by incorrect coding of formating instructions and special characters; on the other hand, by the fact that not all of the defined special characters can be printed by every type of printer. If FORMAT reports too many errors, it may be desirable to correct the input data first. For this purpose, it must be possible to decide in the above example whether the result of FORMAT is to be printed or not. The following approach may be considered:

```
$$? "Print listing file? (yes/no)", RESPONSE=""
$$IF ("<RESPONSE>".AB."yes") THEN
#print,,<PRINTER>
$$ENDIF
```

However, this instruction sequence will not have the desired effect. Namely, it will be executed as soon as the macro PRINT is called, and not when the commands generated by the macro are executed. But at this time FORMAT has not yet been executed and it cannot be determined whether the text is to be printed or not. The macro instructions for this decision may be executed only after formating. This may be achieved with the command #MACRO:

```
#macro
$$? "Print listing file? (yes/no)", RESPONSE=""
$$IF ("<RESPONSE>".AB."yes") THEN
#print,,<PRINTER>
$$ENDIF
*eof
```

In this form, however, the macro instructions would also be executed as soon as the macro is called. To prevent this, the marker for macro instructions (in this case the dollar sign) must also be changed. Thus, the macro instructions for the command #MACRO are not identified as such when the macro PRINT is called, but only when the command #MACRO is executed:

```
$$=*
#macro
$$? "Print listing file? (yes/no)", RESPONSE=""
$$IF ("<RESPONSE>".AB."yes") THEN
#print,,<PRINTER>
$$ENDIF
*eof
**=$
```

One problem still remains: the value assigned for the macro variable RESPONSE which is enclosed in pointed brackets may not be inserted before the command #MACRO is executed, because the variable has not been defined before that point in time. Thus, not only the marker for the macro instructions has to be changed, but also the brackets which enclose macro variables to

be replaced. The desired effect is now achieved by the following instruction:

```
$$=* [ ]
#macro
$$? "Print listing file? (yes/no)", RESPONSE=""
$$IF ("<RESPONSE>".AB."yes") THEN
#print,,[PRINTER]
$$ENDIF
*eof
**=$ < >
```

Because this inquiry in the macro PRINT is to be listed only after FORMAT and not after GLISTING, the PRINT command in the last line may be replaced by the following lines:

```
$$IF ("<MODE>".EQ."F") THEN
$$=* [ ]
#macro
$$? "Print listing file? (yes/no)", RESPONSE=""
$$IF ("<RESPONSE>".AB."yes") THEN
#print,,[PRINTER]
$$ENDIF
*eof
**=$ < >
$$ELSE
#print,,<PRINTER>
$$ENDIF
```

When the macro PRINT that has been changed in this way is called with the command

```
#$PRINT,source file,f
```

the following command sequence will be generated:

```
#format,source=source file,erase=+,parameter=*
dev       HPII
*eof
#macro
$$? "Print listing file? (yes/no)", RESPONSE=""
$$IF ("<RESPONSE>".AB."yes") THEN
#print,,HPII
$$ENDIF
*eof
```

Character set

Survey:

General notes

In the following tables, the input code for the desired character is given in the far left column. Since some keyboards do not feature all ASCII characters (DIN 66003, International Reference Version = 7-bit code complying with ISO-Norm 664), the first two tables (7-bit and 8-bit TUSTEP character set) provide an alternate input code where applicable. If a keyboard featuring the German umlaut characters and the double-S character is being used, these keys can be used instead of the corresponding letters preceded by the control character "^".

For reasons of space and legibility, the following tables feature no alternate key code combinations. However, the individual characters can always be entered with one of the input codes outlined in the first two tables. For example, the accent grave e can be written as either "%\e" or "%^/e" (the accent grave character is encoded as a procent sign + backslash, but the first table also features "^/" as an alternative for the backslash character). The ligature ae can be written as either "#.^a" or "#.ä" ("#.^a" is specified for the ligature, but in the second table "ä" can be used as an alternative for "^a".

Umlauts and the double-S character can only be used for input when the appropriate code has been set with the command #DEFINE. This also means than various accented letters may also be typed in directly. For example, if one of the codes CP437 or CP850 has been set on a IBM-compatibe PC, an e grave can also be entered with "`e". In this case, "è" will appear on screen upon input. But regardless of how it is entered, e grave will be recorded in the file as "%\e".

Characters from the first two tables (7-bit and 8-bit TUSTEP character set) will be written to file as a single character and will occupy one byte. Characters from the following tables will be written as character combinations composed of characters taken from the first two tables and will therefore occupy more bytes. For example, the ligature ae, specified by the input code "#.^a", takes up three bytes with the combination "#.ä" ("#" and "." are characters from the first table, "^a" is a character from the second table).

Every letter can be assigned up to three accent marks: two above the letter and one below the letter. Accent codes must be entered before the respective letter (for more than one accent, in the order from top to bottom). Freestanding accent marks must be written before a hard space (e.g. %<_ for a free standing circumflex).

For representing non-Latin alphabets, an additional display code (see page 327) is necessary at the beginning and end of the appropriate text, since the input coding for special alphabets is not automatically provided.

A ("CTC)" after the name of a character means that this character is used as a print control character in TUSTEP

programs. If a character so marked is to be used as a printed character instead of a control character, it should be preceded by a "^" when entered. (cf. table for the "8-bit TUSTEP character set").

Special features of the typesetting program

The available characters for the typesetting program (command #TYPESET) deviates in certain points from the list of characters outlined here. Please refer to the description of the typesetting program starting in Chapter 12. The most important deviations from the Latin alphabet are:

- The characters { and } are used as control characters and must be encoded as #.{ and #.} to be printed.
- The character pairs "", ++, << and >> as well as the character ^ (input code: ^^) are used as control codes.
- Characters encoded with the control code #(name) are only partially available and must be encoded with a typeset macro.

7-bit TUSTEP character set

		Space, blank
!	!	Exclamation point
"	"	Quotation marks
#	#	Hash mark (CTC for special characters)
$	$	Dollar (CTC for listing file)
%	%	Procent (CTC for accent marks)
&	&	Ampersand (CTC for listing file)
'	'	Apostrophe
(	(	Left parenthesis
)	)	Right parenthesis
*	*	Asterisk
+	+	Plus
,	,	Comma
-	-	Minus
.	.	Period
/	/	Forward slash
:	:	Colon
;	;	Semicolon
<	<	Left pointed bracket
>	>	Right pointed bracket
=	=	Equals sign
?	?	Question mark
@	@	At (CTC for listing file)
[	[	Left bracket
]	]	Right bracket
\		Backslash (CTC for discretionary hyphen)
_		Underscore (CTC for hard space)
{	{	Left brace
}	}	Right brace
|	|	Vertical bar
a	a	Lowercase letters a to z
A	A	Uppercase letters A to Z
1	1	Numerals 0 to 9

8-bit TUSTEP character set

^!		¡	Inverted exclamation point (Spanish)
^"		▒	Filled block
^#		#	Hash mark
^$		$	Dollar
^%		%	Procent
^&		&	Ampersand
^'		'	Inverted apostrophe
^*		×	Multiplication sign
^+		†	Dagger
^-		—	Dash (long minus sign)
^.		·	Multiply dot
^=		●	Large filled dot
^?		¿	Spanish question mark
^@		@	At
^\	^,	\	Backslash
^_		_	Underscore
^|	^;	|	Long vertical bar
^1		¹	small superscript digits 0 to 9
^a	ä	ä	lowercase umlaut ä
^o	ö	ö	lowercase umlaut ö
^s	ß	ß	double s (German)
^u	ü	ü	lowercase umlaut ü
^A	Ä	Ä	uppercase umlaut Ä
^O	Ö	Ö	uppercase umlaut Ö
^U	Ü	Ü	uppercase umlaut Ü

Characters encoded with the escape character "#"

#'x	^{x}	Superscript character (7-bit TUSTEP character set)
#,x	$_{x}$	Subscript character (7-bit TUSTEP character set)
#;x	^{x}	Superior character (7-bit TUSTEP character set)
#!x	$_{x}$	Inferior character (7-bit TUSTEP character set)
#.!	§	Paragraph
#.'	″	Double quote
#.,	„	Base double quote
#.(	ʿ	Ayin
#.)	ʾ	Aleph
#.%	ℶ	Double Aleph
#.*	°	Degree
#.-	′	Minute
#.=	″	Second
#..	→	Reference arrow
#./	‖	Double vertical bar
#.:	›	Single guillemet pointing right
#.;	‹	Single guillemet pointing left
#.<	«	Double guillemet pointing left
#.>	»	Double guillement pointing right
#.[	⌜	Angle upper left
#.]	⌝	Angle upper right

Characters encoded with the escape character "#"

#'x	^{x}	Superscript letter
#,x	$_{x}$	Subscript letter
#;x	x̆	Superior letter
#!x	$\underset{x}{o}$	Inferior letter
#.d	đ	d with cross-stroke (Serbocroatian)
#.D	Đ	D with cross-stroke (Serbocroatian)
#.i	ı	Dotless i (Turk)
#.j	ij	Ligature ij
#.J	IJ	Ligature IJ
#.l	ł	l Slash (Polish)
#.L	Ł	L Slash (Polish)
#.o	ø	o Slash (Danish)
#.O	Ø	O Slash (Danish)
#.p	þ	Lowercase thorn (Icelandic)
#.P	Þ	Uppercase thorn (Icelandic)
#.s	ſ	Long s
#.z	ȝ	Long z / lowercase yogh (Old/Middle English)
#.Z	Ȝ	Uppercase yogh (Old/Middle English)
#.^a	æ	Ligature ae
#.^A	Æ	Ligature AE
#.^d	ð	Lowercase eth (Old/Middle English)
#.^D	Ð	Uppercase eth (Old/Middle English)
#.^o	œ	Ligature oe
#.^O	Œ	Ligature OE

Characters encoded with the control character "#(name)" special characters

#(AEH)	~	Similar
#(BSL)	\	Backslash
#(BOL)	┌	Box: upper left corner
#(BOM)	┬	Box: upper intersection
#(BOR)	┐	Box: upper right corner
#(BML)	├	Box: left intersection
#(BMM)	┼	Box: center intersection
#(BMR)	┤	Box: right intersection
#(BUL)	└	Box: lower left corner
#(BUM)	┴	Box: lower intersection
#(BUR)	┘	Box: lower right corner
#(C)	©	Copyright
#(CE)	¢	Cent
#(DIF)	∂	Differential
#(DIV)	÷	Division
#(DO)	$	Dollar
#(DEA)	⟦	Double left brackets
#(DEZ)	⟧	Double right brackets
#(DEAG)	⟦	Double left brackets (large)
#(DEZG)	⟧	Double right brackets (large)
#(DP)	″	Double stroke
#(DSS)	‖	Double vertical bar
#(DF)	‴	Triple stroke
#(DM)	∩	Average set
#(DMG)	∩	Average set (large)
#(EOM)	⊃	Real superset
#(ETM)	⊂	Real subset
#(EAG)	⎡	Left bracket (large)

#(EZG)	⎤	Right bracket (large)
#(EF)	′	single stroke
#(EL)	∈	Element
#(ENT)	≙	Corresponds to
#(EG)	⋁	There exists
#(EGE)	∃	There exists (reverse E)
#(FA)	⋀	For all
#(FAA)	∀	For all (inverse A)
#(FUE)	∝	Proportional
#(GES)	∩	Intersects
#(GAG)	⎧	Left brace (large)
#(GZG)	⎫	Right brace (large)
#(GR)	>	Greater than
#(GGL)	≥	Greater than or equal to (with -)
#(GGLD)	≧	Greater than or equal to (with =)
#(ID)	≡	Equivalence
#(IW)	¤	International currency symbol
#(INT)	∫	Integral
#(INTG)	⎮	Integral (large)
#(KP)	X	Cartesian product
#(KPG)	⨯	Cartesian product (large)
#(KL)	<	Less than
#(KGL)	≤	Less than or equal to (with -)
#(KGLD)	≦	Less than or equal to (with =)
#(KGR)	≃	Congruent (with -)
#(KGRD)	≅	Congruent (with =)
#(KRS)	○	Circle
#(LM)	∅	Empty set
#(LN)	¬	Logical Not
#(LO)	∨	Logical Or
#(LU)	∧	Logical And
#(MKR)	×	Multiply x
#(MPU)	·	Multiply dot
#(MC)	ℂ	Set of complex numbers
#(MN)	ℕ	Set of natural numbers

#(MQ)	ℚ	Set of rational numbers
#(MR)	ℝ	Set of real numbers
#(MZ)	ℤ	Set of complete numbers
#(MKS)	⏑	Meter: short syllable
#(MKLS)	⏒	Meter: short or long syllable
#(MLS)	‒	Meter: long syllable
#(MPL)	∓	Minos or Plus
#(NBL)	∇	Gradient
#(NEL)	∉	Not an element
#(NID)	≢	Not identical (with /)
#(NIDS)	≣	Not identical (with |)
#(NOM)	⊅	Not a superset
#(NTM)	⊄	Not a subset
#(P)	℗	Published
#(PA)	§	Paragraph
#(PFB)	↔	Arrow pointing left and right
#(PFBD)	⇔	Double arrow pointing left and right
#(PFL)	←	Arrow pointing left
#(PFLD)	⇐	Double arrow pointing left
#(PFOL)	↖	Arrow pointing upper left
#(PFLR)	⇆	Arrows pointing left and right
#(PFR)	→	Arrow pointing right
#(PFRD)	⇒	Double arrow pointing right
#(PFO)	↑	Arrow pointing up
#(PFOO)	⇈	Two arrows pointing up
#(PFOU)	⇅	Arrows pointing up and down
#(PFU)	↓	Arrow pointing down
#(PF)	£	Pound sterling
#(PLM)	±	Plus or Minus
#(PR)	∏	Product
#(PRG)	⎲⎳	Product (large)
#(PRM)	‰	Per thousand
#(R)	®	Registered
#(RAG)	⎛	Left parenthesis (large)
#(RZG)	⎞	Right parenthesis (large)
#(SR)	⊥	Perpendicular

#(SUM)	$\sum$	Summation
#(SUMG)	$\sum$	Summation (large)
#(UOM)	$\supseteq$	Reflex superset
#(UTM)	$\subseteq$	Reflex subset
#(UE)	∞	Infinite
#(UGF)	$\approx$	Approximately equal
#(UGR)	$\gtrsim$	Approximately greater than
#(UKL)	$\lesssim$	Approximately less than
#(UGL)	$\neq$	Not equal (with /)
#(UGLS)	$\neq$	Not equal (with |)
#(VER)	$\cup$	Union
#(VM)	$\cup$	Union
#(VMG)	$\bigcup$	Union (large)
#(WI)	$\sphericalangle$	Angle
#(WU)	$\surd$	Radical
#(WUG)	$\surd$	Radical (large)
#(ZWR)	␣	Intervening space

Example of a box (using FORMAT instructions):

```
$$$ &a10 &+12 &m &+12 &m
$$$ #(BOL) @w ^- &t2 #(BOR)
$$$ ^| Example of a box &t2 ^|
$$$ #(BML) @w ^- &t #(BOM) @w ^- &t #(BMR)
$$$ ^| left &t ^| right &t ^|
$$$ ^| column &t ^| column &t ^|
$$$ #(BUL) @w ^- &t #(BUM) @w ^- &t #(BUR)
```

Example of a box	
left column	right column

Accents and diacritical marks

a) above letters

%"	ő	Double acute
%(	o	Semicircle (open at bottom)
%)	ŏ	Semicircle (open at top)
%*	o̊	Ring
%,	ó	Comma, apostrophe
%-	ō	Macron (horizontal crossbar)
%.	ȯ	Dot
%:	ö	Diaeresis
%/	ó	Acute
%\	ò	Grave
%>	ǒ	Haček
%<	ô	Circumflex
%?	õ	Tilde

b) placed below letters

%""	ő	Double acute
%((	o̜	Semicircle (open at bottom)
%))	o̯	Semicircle (open at top), brevis
%**	o̥	Ring
%,,	o̧	Comma
%--	o̲	Macron (horizontal crossbar)
%..	ọ	Dot
%::	o̤	Diaeresis
%;	o̧	Cedilla
%;;	ǫ	Ogonek (Polish hook)
%//	o̩	Acute
%\\	o̗	Grave
%>>	o̯	Haček
%<<	o̭	Circumflex
%??	o̰	Tilde

Greek alphabet

Start	Greek:	End	Greek	#G+ : #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	ϑ	U	Θ	Theta
i	ι	I	Ι	Iota
c	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,ϲ	S,Q	Σ,Ϲ	Sigma
t	τ	T	Τ	Tau
y	υ	Y	Υ	Ypsilon
f	φ	F	Φ	Phi
x	χ	X	Χ	Chi
c	ψ	C	Ψ	Psi
v	ω	V	Ω	Omega
		W	Ϝ	Digamma

!	·	Greek semicolon
;	;	Greek question mark
?	;	Greek question mark

Greek accents

%(	ὁ	Spiritus asper
%)	ὀ	Spiritus lenis
%\	ὸ	Grave
%/	ό	Acute
%?	ὸ	Tilde
%(\	ὃ	Spiritus asper + Grave
%(/	ὅ	Spiritus asper + Acute
%(?	ὅ̃	Spiritus asper + Tilde
%)\	ὃ	Spiritus lenis + Grave
%)/	ὅ	Spiritus lenis + Acute
%)?	ὅ̃	Spiritus lenis + Tilde

^a	ᾳ	Alpha with iota subscriptum
^h	ῃ	Eta with iota subscriptum
^v	ῳ	Omega with iota subscriptum

%-	ο̄	Macron, horizontal crossbar (above the letter)
%.	ȯ	Dot (above the letter)
%:	ö	Diaeresis (above the letter)
%--	o̲	Macron, horizontal crossbar (below the letter)
%..	ọ	Dot (below the letter)
%::	o̤	Diaeresis (below the letter)

If an accent is to be placed in front of a Greek uppercase letter, a "_" must be inserted between the accent code and the letter (does not apply to data for the typeset program).

Hebrew ||| type

Start/end of Hebrew type: #H+/#H-

a	א			Alef
b	ב	B	בּ	Beth
g	ג	G	גּ	Gimel
d	ד	D	דּ	Daleth
h	ה	H	הּ	Heh
u	ו	U	וּ	Vav
z	ז	Z	זּ	Zayin
x	ח			Cheth
j	ט	J	טּ	Teth
i	י	I	יּ	Yod
k,^k	כ,ך	K	כּ	Chapg
l	ל	L	לּ	Lamed
m,^m	מ,ם	M	מּ	Mem
n,^n	נ,ן	N	נּ	Nun
s	ס	S	סּ	Samekh
y	ע			Ayin
p,^p	פ,ף	P	פּ	Peh
c,^c	צ,ץ	C	צּ	Tsadi
q	ק	Q	קּ	Koph
r	ר			Resh
w	ש	W	שּ	Sin,Shin
t	ת	T	תּ	Tav

#.u	וו	Ligature Vav Vav
#.^u	וי	Ligatur Vav Yod
#.i	״	Ligatur Yod Yod
'	׳	single abbreviation
"	״	double abbreviation
-	־	Maqqef
:	׃	Soph pasuq
^.	׀	

Dotting in Hebrew

#"0	הְ	Shewa
#"1	הִ	Hireg
#"2	הֵ	Sere
#"3	הֶ	Segol
#"4	הָ	Qamas
#"5	הַ	Pathah
#"6	הָ	Qubbus
#"7	הֹ	Holem
#"8	שׁ	Shin
#"9	שׂ	Sin
#"0#"3	הֶ	Hataf Segol
#"3#"0	הֶ	Hataf Segol
#"0#"4	הָ	Hataf Qamas
#"4#"0	הָ	Hataf Qamas
#"0#"5	הַ	Hataf Patah
#"5#"0	הַ	Hataf Patah
^k#"0	ךְ	final Chaph with Shewa
^k#"4	ךָ	final Chaph with Qamas
u#"7	וֹ	Vav with Holem
#"-	הֿ	single Rafe
#"=	ה֒	double Rafe
#"*	ה֯	Ring
#"+	ה֮	X
#">	ה֎	Hachek

Encoding example

#h+w#"8#"4l#"7^m#h- ⟶ שָׁלֹם

Russian (Cyrillic) alphabet

Start	End	Russian		#R+ / #R-
a	а	A	А	a
b	б	B	Б	be
v	в	V	В	ve
g	г	G	Г	ge
d	д	D	Д	de
e	е	E	Е	ye
^e	ё	^E	Ё	yo
h	ж	H	Ж	she
z	з	Z	З	ze
i	и	I	И	i
j	й	J	Й	i kratkoye
k	к	K	К	ka
l	л	L	Л	el
m	м	M	М	em
n	н	N	Н	en
o	о	O	О	o
p	п	P	П	pe
r	р	R	Р	er
s	с	S	С	es
t	т	T	Т	te
u	у	U	У	u
f	ф	F	Ф	ef
x	х	X	Х	kha
c	ц	C	Ц	tse
q	ч	Q	Ч	che
w	ш	W	Ш	sha
^w	щ	^W	Щ	shcha
^p	ъ	^P	Ъ	yor, twerdyj znak
y	ы	Y	Ы	ery
^b	ь	^B	Ь	yer, mjagkij znak
^a	э	^A	Э	e oborotnoye
^u	ю	^U	Ю	yu
^o	я	^O	Я	ya
^i	і	^I	І	i s totschkoj
^y	ѣ	^Y	Ѣ	yat

^f	ѳ	^F	Ѳ	fita
^v	ѵ	^V	Ѵ	izhitsa

Special Bulgarian characters

^w	щ	^W	Щ	št
^p	ъ	^P	Ъ	ă
^z	ѫ	^Z	Ѫ	ǫ

Special Macedonian characters

%/g	ѓ	%/G	Ѓ	ǵ
%/k	ќ	%/K	Ќ	ḱ
^s	ѕ	^S	Ѕ	dz
^j	ј	^J	Ј	j
x	х	X	Х	h
^l	љ	^L	Љ	lj
^n	њ	^N	Њ	nj
^c	џ	^C	Џ	dž

Special Serbian characters

^d	ђ	^D	Ђ	đ
^j	ј	^J	Ј	j
^l	љ	^L	Љ	lj
^n	њ	^N	Њ	nj
^h	ћ	^H	Ћ	ć
x	х	X	Х	h
^c	џ	^C	Џ	dž

Special Ukranian characters

g	г	G	Г	h
^g	є	^G	Є	je
i	и	I	И	y
%:^i	ї	%:^I	Ї	ї
^i	і	^I	І	i
'	'			-

Special characters in Belorussian

g	г	G	Г	h
%)u	ў	%)U	Ў	ŭ

Accents

%)	ŏ	Semicircle (open at top)
%:	ö	Diaeresis
%/	ó	Acute
%\	ò	Grave

Cyrillic (Church Slavonic) alphabet

Start	End	Cyrillic font		#C+ / #C-	
a	ѧ	A	Ѧ	a	Az
b	ѣ	B	Ѣ	b	Buki
v	ѵ	V	Ѵ	v	Vědi
g	г	G	Г	g	Glagol'
d	д	D	Д	d	Dobro
e	ѥ	E	Ѥ	e	Est'
^g	є	^G	Є	e	Est'
h	ж	H	Ж	ž	Živěte
^s	ѕ	^S	Ѕ	dz	Zělo
z	з	Z	З	z	Zemlja
i	и	I	И	i	Iže
j	і	J	І	i	I
k	к	K	К	k	Kako
l	л	L	Л	l	Ljudi
m	м	M	М	m	Myslite
n	н	N	Н	n	Naš
o	о	O	О	o	On
p	п	P	П	p	Pokoj
r	р	R	Р	r	Rci
s	с	S	С	s	Slovo
t	т	T	Т	t	Tverdo
^u	ѹ	^U	Ѹ	u	Uk
f	ф	F	Ф	f	Fert
x	х	X	Х	ch	Chěr
^o	ѡ	^O	Ѡ	ω	Ó
c	ц	C	Ц	c	Tsi
q	ч	Q	Ч	č	Tšerv'
w	ш	W	Ш	š	Ša
^w	щ	^W	Щ	št	Šta
^p	ъ	^P	Ъ		Jer
#.^p	ꙑ	#.^P	Ꙑ	y	Jery
^b	ь	^B	Ь		Jerek
#.^b	ы	#.^B	Ы	y	Jery
^y	ѣ	^Y	Ѣ	ě	Jet'
#.a	ѩ	#.A	Ѩ	ja	Ja
#.o	ю	#.O	Ю	ju	Ju

#.^g	ⱑ	#.^G	Ⱑ	je	Je
^e	ⱕ	^E	Ⱕ	ę	Ęs
^a	ⱘ	^A	Ⱘ	ą	Ąs
#.^e	ⱙ	#.^E	Ⱙ	ję	Jęs
#.^a	ⱚ	#.^A	Ⱚ	ją	Jąs
^x	ⱛ	^X	Ⱛ	ks	Ksi
^c	ⱜ	^C	Ⱜ	ps	Psi
^f	ⱝ	^F	Ⱝ	ѳ	Fita
u	ⱞ	U	Ⱞ	ÿ	Ižica

Syrian alphabet

1st column: final letters (joined to the right)
2nd column: central letters (joined to both sides)
3rd column: initial letters (joined to the left)
4th column: isolated letters

Start / End Syrian font: #Y+ / #Y-

^a			^A	ܐ			ܐ	Alef
^b	b	B	^B	ܒ	ܒ	ܒ	ܒ	Beth
^g	g	G	^G	ܓ	ܓ	ܓ	ܓ	Gomal
^d			^D	ܕ			ܕ	Dolath
^h			^H	ܗ			ܗ	He
^u			^U	ܘ			ܘ	Vaw
^z			^Z	ܙ			ܙ	Zain
^x	x	X	^X	ܚ	ܚ	ܚ	ܚ	Khet
^j	j	J	^J	ܛ	ܛ	ܛ	ܛ	Teth
^i	i	I	^I	ܝ	ܝ	ܝ	ܝ	Jud
^k	k	K	^K	ܟ	ܟ	ܟ	ܟ	Koph
^l	l	L	^L	ܠ	ܠ	ܠ	ܠ	Lomad
^m	m	M	^M	ܡ	ܡ	ܡ	ܡ	Mim
^n	n	N	^N	ܢ	ܢ	ܢ	ܢ	Nun
^s	s	S	^S	ܣ	ܣ	ܣ	ܣ	Semkath
^y	y	Y	^Y	ܝ	ܝ	ܝ	ܝ	Ee
^p	p	P	^P	ܦ	ܦ	ܦ	ܦ	Pe
^c			^C	ܨ			ܨ	Sode
^q	q	Q	^Q	ܩ	ܩ	ܩ	ܩ	Qoph
^r			^R	ܪ			ܪ	Ris
^w	w	W	^W	ܫ	ܫ	ܫ	ܫ	Sin
^t			^T	ܬ			ܬ	Tau

Arabian alphabet

1. Spalte: final leters (joined to the right)
2. Spalte: medial letters (joined on both sides)
3. Spalte: initial letters (joined to the left)
4. Spalte: isolated letters

Start / End Arabian font: #A+ / #A-

^a			^A	ـا			ا	ʾ/a	Alif
^b	b	B	^B	ب	بـ	ـبـ	ـب	b	Bāʾ
^t	t	T	^T	ت	تـ	ـتـ	ـت	t	Tāʾ
^o	o	O	^O	ث	ثـ	ـثـ	ـث	ṯ	Ṯāʾ
^j	j	J	^J	ج	جـ	ـجـ	ـج	ǧ	Ǧīm
^h	h	H	^H	ح	حـ	ـحـ	ـح	ḥ	Ḥāʾ
^x	x	X	^X	خ	خـ	ـخـ	ـخ	ḫ	Ḫā
^d			d	د			ـد	d	Dāl
^D			D	ذ			ـذ	ḏ	Ḏāl
^r			r	ر			ـر	r	Rāʾ
^R			R	ز			ـز	z	Zāy
^s	s	S	^S	س	سـ	ـسـ	ـس	s	Sīn
^w	w	W	^W	ش	شـ	ـشـ	ـش	š	Šīn
^c	c	C	^C	ص	صـ	ـصـ	ـص	ṣ	Ṣād
^g	g	G	^G	ض	ضـ	ـضـ	ـض	ḍ	Ḍād
^p	p	P	^P	ط	طـ	ـطـ	ـط	ṭ	Ṭāʾ
^z	z	Z	^Z	ظ	ظـ	ـظـ	ـظ	ẓ	Ẓāʾ
^y	y	Y	^Y	ع	عـ	ـعـ	ـع	ʿ	ʿAin
^v	v	V	^V	غ	غـ	ـغـ	ـغ	ġ	Ġain
^f	f	F	^F	ف	فـ	ـفـ	ـف	f	Fāʾ
^q	q	Q	^Q	ق	قـ	ـقـ	ـق	q	Qāf
^k	k	K	^K	ك	كـ	ـكـ	ـك	k	Kāf
^l	l	L	^L	ل	لـ	ـلـ	ـل	l	Lām
^m	m	M	^M	م	مـ	ـمـ	ـم	m	Mīm
^n	n	N	^N	ن	نـ	ـنـ	ـن	n	Nūn
^e	e	E	^E	ه	هـ	ـهـ	ـه	h	Hāʾ
^u			^U	و			ـو	w/u	Wāw
^i	i	I	^I	ى	يـ	ـيـ	ـى	y/i	Yāʾ

Ligatures and special characters in Arabic

	A			ا	Alif preceding Lām at start of word
#.^f	#.^F	ﻔ		ﻓ	(fi) Ligature Fā-Yāʾ
#.^l	#.^L	ﻠ		ﻟ	(li) Ligature Lām-Yāʾ
#.^x	#.^X	ﻼ		ﻻ	(la) Ligature Lām-Alif
#.^A				ء	Hamza (isolated)
#.i	#.I	ﺌ		ﺉ	Hamza above Yāʾ
#.j	#.J	ﺒ		ﺑ	Hamza below Yāʾ
#.^u	#.^U	ﺆ		ﺅ	Hamza above Wāw
#.^e	#.^E	ﺔ		ﺓ	Tāʾ marbūṭa

!	!	Exclamation point
"	‗	Quotation marks
'	'	Apostrophe
,	،	Comma
.	۔	Period
:	؛	Colon
;	؛	Semicolon
?	؟	Question mark

Addition characters in Persian

#.^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

Vowel points and other diacritical marks in Arabic

a) above letters

%)	◌ٕ	Hamza
%/	◌َ	Fatḥa
%"	◌ً	Fatḥa-Tanwīn
%,	◌ُ	Ḍamma
%:	◌ٌ	Ḍamma-Tanwīn
%*	◌ْ	Sukūn
%>	◌ّ	Tašdīd
%!	◌ٰ	Alif
%?	◌ٓ	Madda
%.	◌ٔ	Waṣla
%(	◌ۡ	Sukūn above ligature Lām-Alif
%\	◌ٖ	Fatḥa above ligature Lām-Alif

b) below letters

%))	◌ٕ	Hamza
%//	◌ِ	Kasra
%\\	◌ٖ	Kasra below ligature (to the right)
%;	◌ٕ	Hamza + Kasra
%""	◌ٍ	Kasra-Tanwīn
%;;	◌ٖ	i dots for final Yāʾ
%,,	◌ٕ	i dots + Kasra for final Yāʾ
%''	◌ٍ	i dots + Kasra-Tanwīn for final Yāʾ

Notes for data preparation

The TUSTEP program features a specail macro, called #*CASH, used to distinguish letters that are left-joined, joined on both sides, right-joined, or isolated. When this macro is used, all letters (except for Dal and Zay) need only to be entred as lowercase letters. erfaßt zu werden. This also applied to characters encoded with a "#.". For example, with this macro an "a" can be entered instead of an "^A", and when entering Persian letters, for example, "#.p" can be written instead of "#.^P". In addition, this macro converts "al" located at the beginning of a word to "Al" (in order to move the Alif closer to Lam). The character combination "fi" and "li" (if not joined to the left) as well as "la" will be automatically converted into the codes for the corresponding ligatures.

Vowel points and other diacritical marks are to written before the character above/under which they are to be set. If more than one vowel point/diacritical mark are given for a single letter, they are to be written in the order they would appear from top to bottom.

Example: %;a%*b%/ra%//ei%,m %.al%/%>n%//b%,%>i

(or %;A%*b%/r^A%//Ei%,^m %.AL%/%>n%//b%,%>^i)

for إِبْرَاهِيمُ ٱلنَّبِيُّ Ibrāhīmu 'n-nabīyyu

Phonetic symbols

The names used for the phonetic symbols are based on the "Phonetic Symbol Guide" by Geoffrey K. Pullum and William A. Ladusaw (Chicago 1986). If a letter itself is part of the name used for it, the name will be inverted so that the letter is located at the beginning of the name, with a comma marking the point of inversion.

The name of each symbol is supplemented by a systematic phonetic description of the symbol according to the conventions used by the IPA (International Phonetic Association). The abbreviations and terms used are:

first position:

sv	semivowel
c	consonant
v	vowel

Vowels:

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

Consonants:

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
		post-alv	post-alveolar
dent	dental	rfl	retroflex
flap	flap	tap	tap
fric	fricative	trill	trill
gl	glottal	uvu	uvular
impl	implosive	v+	voiced
intd	interdental	v-	voiceless
lab	labial	vel	velar
lb-den	labiodental		

Start / End phonetic symbols: #P+ / #P-

i	ɨ	i, lower case	v hi fr unr
I	ɪ	iota	v sem-hi fr unr
^i	ɨ̵	i, barred	v hi ce unr
^I	ᵻ	i umlaut	v hi ce unr
y	ʏ	y, lower case	v hi fr r
Y	ʏ	y, small capital	v sem-hi fr r
u	ʉ	u, lower case	v hi back r
U	ʊ	upsilon	v sem-hi back r
^u	ʉ̵	u barred	v hi ce r
%:u	ʉ̈	u umlaut	v hi ce r
^m	ɯ	m, turned	v hi back unr
^W	ɷ	omega, closed	v sem-hi back r
e	ɐ	e, lower case	v up-mid fr unr
E	ɛ	epsilon	v lo-mid fr unr
^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	ä	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	ɤ	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	ɔ̈	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	ɵ	o, barred	v up-mid ce r
^Q	ɞ	epsilon, closed reversed	v lo-mid ce r
a	ɐ	a, lower case	v low fr unr
A	ɑ	a, script	v low back unr
^a	ɶ	o-e ligature, small capital	v lo-mid fr r
^A	ɒ	a, turned script	v low back r
^3	ɐ	a, turned	v sem-lo ce unr
^G	ɡ	a, inverted	v sem-lo ce unr
^v	ɻ	r with right tail, turned	sv med-appr rfl
V	ʋ	v, script	sv med-appr lb-den

^y	ʯ	h, turned	sv med-appr lb-pal
^M	ɯ	m with long right leg, turned	sv med-appr vel
^R	ɹ	r, turned	sv med-appr post-alv
w	w	w, lower case	sv med-appr lb-vel
%:w	ẅ	w umlaut	sv med-appr lb-vel ce
p	p	p, lower case	c pl bil v-
b	b	b, lower case	c pl bil v+
^p	ɸ	phi	c fric bil v-
^b	β	beta	c fric bil v+
B	ɓ	b, hooktop	c impl bil
t	t	t, lower case	c pl alv v-
d	d	d, lower case	c pl alv v+
T	θ	theta	c fric intd v-
^t	ð	eth	c fric intd v+
^T	ʧ	t-esh ligature	c affr pal-alv v-
^D	ʤ	d-yogh ligature	c affr pal-alv v+
^d	ɖ	d, right-tail	c pl rfl v+
D	ɗ	d, hooktop	c impl alv
^K	ʈ	t with right tail	c pl rfl v-
c	c	c, lower case	c pl pal v-
C	ɟ	j, barred dotless	c pl pal v+
^c	ç	c cedilla	c fric pal v-
k	k	k, lower case	c pl vel v-
^k	q	q, lower case	c pl uvu v-
g	g	g, lower case	c pl vel v+
G	ɢ	g, small capital	c pl uvu v+
^g	ɠ	g, hooktop	c impl vel
?	ʔ	glottal stop	c pl gl v-
f	f	f, lower case	c fric lb-den v-
v	v	v, lower case	c fric lb-den v+
^P	ɾ	r with long leg	c alv fric trill
s	s	s, lower case	c fric alv v-
z	z	z, lower case	c fric alv v+
S	ʃ	esh	c fric pal-alv v-
Z	ʒ	yogh	c fric pal-alv v+
^s	ɕ	c, curly-tail	c fric alv-pal v-
^z	ʑ	z, curly-tail	c fric alv-pal v+
6	ʃ	esh, curly-tail	c db-fr pal-alv v-
^6	ʒ	yogh, curly-tail	c db-fr pal-alv v+

^S	ʂ	s with right tail	c fric rfl v-
^Z	ʐ	z with right tail	c fric rfl v+
j	j	j, lower case	c fric pal v+
^j	ʲ	j, superscript	c col j
x	x	x, lower case	c fric vel v-
^x	ɣ	gamma	c fric vel v+
X	χ	chi	c fric uvu v-
^X	ʁ	r, inverted small capital	c fric uvu v+
h	h	h, lower case	c fric gl v-
H	ɧ	heng, hooktop	c db-fr vel
^h	ʰ	h, superscript	c asp
^H	ɦ	h, hooktop	c fric gl v+
^F	ħ	h, crossed	c fric phar v-
^?	ʕ	glottal stop, reversed	c fric phar v+
m	m	m, lower case	c nas bil
M	ɱ	m with leftward tail at right	c nas lb-den
n	n	n, lower case	c nas alv
^n	ŋ	eng	c nas vel
N	ɴ	n, small capital	c nas uvu
^N	ɲ	n with leftward hook at left	c nas pal
^9	ɳ	n with right tail	c nas rfl
l	l	l, lower case	c lat alv appr v+
^Y	ʎ	y, turned	c lat pal appr
^l	ɬ	l with tilde	c lat vel alv appr
1	ɮ	l, belted	c lat alv fric v-
L	ɭ	l with right tail	c lat rfl appr
^L	ɮ	l-yogh ligature	c lat alv fric v+
r	r	r, lower case	c alv trill
^r	ɾ	r, fish-hook	c alv tap
R	ʀ	r, small capital	c uvu trill
P	ɽ	r with right tail	c rfl flap
^V	ɺ	r, turned long-legged	c alv lat flap
4	ʳ	r, superscript	c col r
^w	ʷ	w, subscript	c lab
W	ʍ	w, inverted	c db-fr lb-vel
0	ʘ	bull's eye	c click bil
5	ɓ	glottal stop, inverted	c click lat alv
^f	ʇ	t, turned	c click dent
^C	ƈ	C, stretched	c click post-alv

"	"	quotation marks
(	(	left parenthesis
)	)	right parenthesis
<	<	left angle bracket
[	[	left sqare bracket
]	]	right square bracket
>	>	right angle bracket
{	{	left curly bracket
}	}	right curly bracket
/	/	slash
=	=	equals
'	’	apostrophe
^'	‘	apostrophe, reversed
*	*	asterisk
,	,	comma
^@	‚	comma turned
^7	⌜	corner, left
7	⌝	corner, right
.	.	full stop
^.	ˑ	half-length mark
:	ː	length mark
^8	⁀	ligature, top
8	‿	ligature, bottom
^&	⁀‿	ligature, top and bottom
2	ₜ	lowering sign
-	−	minus sign
|	ᛁ	pipe (lower case height)
^\	׀	pipe (upper case height)
^|	|	pipe (full height)
+	+	plus sign
^2	₊	raising sign
;	;	semicolon
^!	ˌ	stroke (inferior), vertical
!	ˈ	stroke (superior), vertical
^%	~	tilde

%'	o̍	nucleus stress
%)	ŏ	breve
%*	o̊	over-ring
%-	o̅	macron
%.	ȯ	over-dot
%/	ó	acute accent
%:	ö	umlaut
%<	ô	circumflex
%>	ǒ	wedge
%?	õ	tilde, superscript
%[	o̅	bridge, superscript
%\	ò	grave accent
%{	o̓	half-ring superscript, left
%}	o̒	half-ring superscript, right
%!	o̩	palatalization hook
%!!	o̧	right hook
%''	o̩	syllabicity mark
%**	o̥	under-ring
%--	o̲	under-bar
%..	ọ	under-dot
%;	o̧	cedilla
%;;	ǫ	polish hook
%>>	o̬	wedge, subscript
%??	o̰	tilde, subscript
%[[	o̱	bridge, subscript
%{{	o̔	half-ring, subscript left
%}}	o̕	half-ring, subscript right
%%(	o͡o	ligature, long, top
%%-	o̅o̅	macron, long
%%))	o͜o	ligature, long, bottom
%%--	o̲o̲	underbar, long

Display types, printed appearance

Start -- end **bold**:	#F+ bzw. #F-
Start -- end *cursive*:	#/+ bzw. #/-
Start -- end l e t t e r s p a c i n g :	#S+ bzw. #S-
Start -- end SMALL CAPS:	#K+ bzw. #K-
Start -- end Arabic:	#A+ bzw. #A-
Start -- end Greek:	#G+ bzw. #G-
Start -- end Hebrew:	#H+ bzw. #H-
Start -- end Cyrillic (Church Slavonic):	#C+ bzw. #C-
Start -- end Phonetic:	#P+ bzw. #P-
Start -- end Russian (Cyrillic):	#R+ bzw. #R-
Start -- end Syrian:	#Y+ bzw. #Y-
Start -- end ~~strikeout line~~:	#0+ bzw. #0-
Start -- end single underlineing:	#1+ bzw. #1-
Start -- end double underlining:	#2+ bzw. #2-
Start -- end bold underlining:	#3+ bzw. #3-
Start -- end dotted underlining:	#4+ bzw. #4-
Start -- end subscript underlining:	#5+ bzw. #5-
Start -- end superscript underlining:	#6+ bzw. #6-
End all displays:	#?-

Superscript and subscript

Raise 1/3 line above base line:	#H:
Lower 1/3 line below base line:	#T:
Raise 1/2 line above base line:	#O:
Lower 1/2 line below base line:	#U:
Return to base line:	#G:

Note

Single characters (from the 7-bit TUSTEP character set) can also be raised or lowered 1/3 of a line (either from the base line, or in addition to their initial raised or lowered position) by placing a "#'" (superscript) or "#," (subscript) in front of the character.

Availability of fonts on selected printers

Type	line printer / screen with ...	L	/	A	G	H	K	P	R	Y
ASCII	ASCII character set	O	N	-	-	-	N	-	-	-
DEUTSCH	German character set	O	N	-	-	-	N	-	-	-
EBCDIC	EBCDIC character set	O	N	-	-	-	N	-	-	-
DECMCS	DEC multinational character set	O	N	-	-	-	N	-	-	-
IBM	IBM-EBCDIC character set	O	N	-	-	-	N	-	-	-
IBMPC	IBM-PC character set	O	N	-	-	-	N	-	-	-

Type	dot matrix printers	L	/	A	G	H	K	P	R	Y
PP	IBM Proprinter and compatible	+	+	-	+	+	+	-	+	-
LX	EPSON LX... and compatible	+	+	-	+	+	+	-	+	-
FX	EPSON FX... and compatible	+	+	-	+	+	+	-	+	-
LQ	EPSON LQ... and compatible	+	+	+	+	+	+	+	+	-

Typ	laser printers	L	/	A	G	H	K	P	R	Y
HP	HP w/o font cartridge	+	N	-	-	-	N	-	-	-
HP-10	HP with 10 cpi cartridge	+	+	-	-	-	+	-	-	-
HP-12	HP with 12 cpi cartridge	+	+	-	-	-	+	-	-	-
HP-GM	HP with Greek + math. characters	+	+	-	O	-	+	-	-	-
HP-LP1	HP as line printer (160 * 64)	+	N	-	-	-	N	-	-	-
HP-LP2	HP as line printer (2 * 80 * 64)	+	N	-	-	-	N	-	-	-
HPDJ	HP DeskJet	+	+	+	+	+	+	+	+	+
HP+	HP PLUS w/o font cartridge	+	+	+	+	+	+	+	+	+
HPII	HP II and compatible	+	+	+	+	+	+	+	+	+
PS-10	Postscript with 10 cpi font	+	+	+	+	+	+	+	+	+
PS-12	Postscript with 12 cpi font	+	+	+	+	+	+	+	+	+
PS-Q1	Postscript landscape single-column	+	+	+	+	+	+	+	+	+
PS-Q2	Postscript landscape double-columns	+	+	+	+	+	+	+	+	+
LN03-10	LN03 Laser printer, Courier 10	O	N	-	O	-	N	-	-	-
LN03-12	LN03 Laser printer, Elite 12	O	N	-	O	-	N	-	-	-

Key to symbols:

L	Latin	G	Greek	P	Phonetic
/	Cursive	H	Hebrew	R	Russian
A	Arabic	K	Small caps	Y	Syrian

+	Font available
-	Font not available
N	Font not available; small caps will be written as uppercase letters in the normal font; cursive characters will be written as Roman characters.
O	Font available, but without accents.

Note

In order to determine which characters can be represented on a specific printer, the TUSTEP character set can be printed out on the selected printer with the command #MANUAL (see page 128).

Alphabetical list of special characters

%<	ô	Accent circonflexe (above the letter)
%<<	o̭	Accent circonflexe (below the letter)
%/	ó	Accent aigu (above the letter)
%//	o̩	Accent aigu (below the letter)
%\	ò	Accent grave (above the letter)
%\\	o̖	Accent grave (below the letter)
#.^a	æ	ae ligature
#.^A	Æ	AE ligature
#(AEH)	~	Similar
#.(	ʿ	Ajin, Ain (transcription symbol)
%/	ó	Acute (above the letter)
%//	o̩	Acute (below the letter)
#.)	ʾ	Alef (transcription character)
#.%	ʒ	Alef, double (transcription symbol)
#.^d	ð	Old/Middle English eth (lowercase)
#.^D	Ð	Old/Middle English eth (uppercase)
#.z	ȝ	Old/Middle English yogh (lowercase)
#.Z	Ȝ	Old/Middle English yogh (uppercase)
"	"	Quotes, double
#.'	″	Quotes, double right
#.,	„	Quotes, double base
#.>	»	Guillemet, double
#.<	«	Guillement, double
#.:	›	Guillement, single
#.;	‹	Guillement, single
'	'	Apostrophe (raised comma)
%,	ó	Apostrophe, comma (above the letter)
^'	'	Apostrophe, inverted

!	!	Exclamation point
^!	¡	Exclamation point, inverted (Spanish)
_		Space, hard
^\	\	Backslash
^,	\	Backslash
#(BSL)	\	Backslash
\		Backslash (control character for discretionary hyphen)
^/		Backslash (control character for discretionary hyphen)
%-	o̅	Bar, horizontal (above the letter)
%--	o̲	Bar, horizontal (below the letter)
_		Blank, (hard space)
%(	o	Semicircle above letter (open at bottom)
%((	o	Semicircle below letter (open at bottom)
#(BML)	├	Box: left intersection
#(BMM)	┼	Box: center intersection
#(BMR)	┤	Box: right intersection
#(BOL)	┌	Box: upper left corner
#(BOM)	┬	Box: upper intersection
#(BOR)	┐	Box: upper right corner
#(BUL)	└	Box: lower left corner
#(BUM)	┴	Box: lower intersection
#(BUR)	┘	Box: lower right corner
%;	ç	Cedilla
#(CE)	¢	Cent
#(C)	©	Copyright
#.d	đ	d with cross-stroke (Serbo-Croatian d)
#.D	Đ	D with cross-stroke (Serbo-Croatian D)
%<	ô	Circumflex (above the letter)

%<<	◌̭	Circumflex (below the letter)
#.o	ø	Danish ö (o slash)
#.O	Ø	Danish Ö (O slash)
#(DIV)	÷	Division
^$	$	Dollar
$	$	Dollar (CTC for listing file)
#(DO)	$	Dollar
%"	◌̋	Double acute (above the letter)
%""	◌̏	Double acute (below the letter)
#.%	꞉	Double alef (transcription symbol)
:	:	Colon
#(DP)	″	Double stroke
#.<	«	Double quotes
#.>	»	Double quotes
#.'	”	Double right quotes
#.,	„	Double base quotes
#(DEA)	⟦	Double left bracket
#(DEAG)	⟦	Double left bracket (large)
#(DEZ)	⟧	Double right bracket
#(DEZG)	⟧	Double right bracket (large)
#./	‖	Double vertical bar
#(DSS)	‖	Double vertical bar
#(DF)	‴	Triple stroke
#(DM)	∩	Intersection
#(DMG)	∩	Intersection (large)
#(EOM)	⊃	Proper superset
#(ETM)	⊂	Proper subset
[	[	Bracket, left
^<	[	Bracket, left
#(EAG)	[	Bracket, left (large)

#(DEA)	⟦	Bracket, left double
#(DEAG)	⟦	Bracket, left double (large)
]	]	Bracket, right
^>	]	Bracket, right
#(EZG)	⟧	Bracket, right (large)
#(DEZ)	⟧	Bracket, right double
#(DEZG)	⟧	Bracket, right double (large)
#(EF)	′	Single stroke
#.:	›	Single quotes
#.;	‹	Single quotes
^!	¡	Inverted exclamation point (Spanish)
^?	¿	Inverted question mark (Spanish)
#(EL)	∈	Element
#(ENT)	≙	Corresponds to
#(EG)	⋁	There exists
#(EGE)	∃	There exists (reverse E)
^&	&	Ampersand
&	&	Ampersand (CTC for listing file)
#.^d	ð	Eth, Old/Middle English (lowercase)
#.^D	Ð	Eth, Old/Middle English (uppercase)
#(FUE)	∝	Varies as
_		Hard space
?	?	Question mark
^?	¿	Question mark, inverted (Spanish)
;	;	Question mark, Greek (located between #G+ and #G-)
?	;	Question mark, Greek (located between #G+ and #G-)
#(FA)	⋀	For all
#(FAA)	∀	For all (inverted A)
#(GES)	∩	Intersects
{	{	Brace, left

^(	{	Brace left
#(GAG)	⎧	Brace, left (large)
}	}	Brace, right
^)	}	Brace, right
#(GZG)	⎫	Brace, right (large)
=	=	Equals
#.*	°	Degree
%\	ò	Grave (above the letter)
%\\	o̖	Grave (below the letter)
;	;	Greek question mark (located between #G+ and #G-)
?	;	Greek question mark (located between #G+ and #G-)
!	·	Greek semicolon (located between #G+ and #G-)
#(GR)	>	Greater than
#(GGL)	≥	Greater or equal (with -)
#(GGLD)	≧	Greater or equal (with =)
#.^D	Ð	Uppercase eth (Old/Middle English)
#.P	Þ	Uppercase Thorn (Islandic)
#.Z	Ȝ	Uppercase yogh (Old/Middle English)
%>	ǒ	Haček, caron (above the letter)
%>>	o̬	Haček, caron (below the letter)
%)	ŏ	Semicircle (top open) (above the letter)
%))	o̩	Semicircle (top open) (under the letter)
#'x	ˣ	Superscript character
^1	¹	Superscript digit, small
'	'	Raised comma (apostrophe)
#.i	ı	I, dotless (Turkish i)
#(ID)	≡	Equivalent
#.j	ij	ij ligature
#.J	IJ	IJ ligature
#(INT)	∫	Integral

#(INTG)	∫	Integral (large)
#(IW)	¤	International currency symbol
#.p	þ	Islandic thorn (lowercase)
#.P	Þ	Islandic Thorn (uppercase)
#(KP)	X	Cartesian product
#(KPG)	⨯	Cartesian product (large)
^@	@	At
@	@	At (CTC for listing file)
^1	¹	Small superscript digit
#(KL)	<	Less than
#(KGL)	≤	Less or equal (with -)
#(KGLD)	≦	Less or equal (with =)
#.^d	ð	Lowercase eth (Old/Middle English)
#.p	þ	Lowercase thorn (Islandic)
#.z	ȝ	Lowercase yogh (Old/Middle English)
,	,	Comma
%,,	o̧	Comma (above the letter)
%,	ó	Comma, Apostrophe (above the letter)
#(KGR)	≃	Congruent (with -)
#(KGRD)	≅	Congruent (with =)
%*	o̊	Ring (above the letter)
%**	o̥	Ring (below the letter)
%;;	o̧	Ogonek, hook (under the letter)
#(MKLS)	⩲	Short or long syllable (meter)
#(MKS)	˘	Short syllable (meter)
#.l	ł	l slash (Polish l)
#.L	Ł	L slash (Polish L)
#(MLS)	–	Long syllable (meter)
^|	|	Long vertical bar
^;	|	Long vertical bar

^-	—	Long minus sign
#.s	ſ	Long s
#.z	ʒ	Long z
#(LM)	∅	Empty set
#.^a	æ	Ligature ae
#.^A	Æ	Ligature AE
#.j	ij	Ligature ij
#.J	IJ	Ligature IJ
#.^o	œ	Ligature oe
#.^O	Œ	Ligature OE
#(LN)	¬	Logical Not
#(LO)	∨	Logical Or
#(LU)	∧	Logical And
^*	×	Multiply x
#(MKR)	×	Multiply x
^.	·	Multiply dot
#(MPU)	·	Multiply dot
#(MC)	ℂ	Set C
#(MN)	ℕ	Set N
#(MQ)	ℚ	Set Q
#(MR)	ℝ	Set R
#(MZ)	ℤ	Set Z
#(LM)	∅	Set, empty
#(MKLS)	⏑	Meter: short or long syllable
#(MKS)	◡	Meter: Short syllable
#(MLS)	–	Meter: Long syllable
#(MPL)	∓	Minus or Plus
-	-	Minus sign
^-	—	Minus sign, long
#.-	′	Minute

#(NBL)	▽	Gradient
#(LN)	¬	Negation (logical Not)
#(NEL)	∉	Not an element
#(NID)	≠	Not identical (with /)
#(NIDS)	≢	Not identical (with |)
#(NOM)	⊅	Not superset
#(NTM)	⊄	Not subset
#(LN)	¬	Not, logical
^#	#	Number
#	#	Number (CTC for special characters)
#.o	ø	o slash (Danish ö)
#.O	Ø	O slash (Danish Ö)
#(EOM)	⊃	Superset, proper
#(UOM)	⊇	Superset, reflex
#(LO)	∨	Or, logical
#.^o	œ	oe ligature
#.^O	Œ	OE ligature
%;;	ǫ	Ogonek, hook (below the letter)
#.!	§	Section
#(PA)	§	Section
#(PFB)	↔	Arrow pointing left and right
#(PFBD)	⇔	Arrow pointing left and right (double)
#(PFL)	←	Arrow pointing left
#(PFLD)	⇐	Arrow pointing left (double)
#(PFO)	↑	Arrow pointing up
#(PFOL)	↖	Arrow pointing upper left
#(PFR)	→	Arrow pointing upper left
#(PFRD)	⇒	Arrow pointing right (double)
#(PFU)	↓	Arrow pointing down
#(PFLR)	⇆	Arrows pointing left and right

#(PFOO)	⇈	Arrows pointing up, two
#(PFOU)	⇅	Arrows pointing up and down
#(PF)	£	Pound sterling
#(PLM)	±	Plus or Minus
+	+	Plus
#.l	ł	Polish l (l slash)
#.L	Ł	Polish L (L slash)
#(PR)	∏	Product
#(PRG)		Product (large)
#(PRM)	‰	Per thousand
^%	%	Percent
%	%	Percent (CTC for accent marks)
#(P)	℗	Published
.	.	Period
%.	ȯ	dot (above the letter)
%..	ọ	Dot (below the letter)
%-	ō	Macron, horiz. bar (above the letter)
%--	o̲	Macron, horiz. bar (below the letter)
#(R)	®	Registered
%*	o̊	Ring (above the letter)
%**	ǫ	Ring (below the letter)
^=	●	Bullet
(	(	Parenthesis, left
#(RAG)		Parenthesis, left (large)
)	)	Parenthesis, right
#(RZG)		Parenthesis, right (large)
^s	ß	Double s (German)
^"	▒	Filled box
/	/	Slash
#.=	″	Second

!	·	Semicolon, Greek (when placed between #G+ and #G-)
#(SR)	⊥	Perpendicular
|	|	Vertical bar
^:	|	Vertical bar
#./	‖	Vertical bar, double
#(DSS)	‖	Vertical bar, double
^|	⎪	Vertical bar, long
^;	⎪	Vertical bar, long
#.d	đ	Serbo-Croatian d (d with cross-stroke)
#.D	Đ	Serbo-Croatian D (D with cross-stroke)
^!	¡	Spanish exclamation point
^?	¿	Spanish question mark
<	<	Angle bracket, left
>	>	Angle bracket, right
^+	†	Dagger
*	*	Asterisk
#./	‖	Bar, double vertical
#(DSS)	‖	Bar, double vertical
|	|	Bar, vertical
;	;	Semicolon
#(SUM)	∑	Summation
#(SUMG)	$\sum$	Summation (large)
#(ETM)	⊂	Subset, proper
#(UTM)	⊆	Subset, reflex
#.p	þ	Thorn, Islandic (lowercase)
#.P	Þ	Thorn, Islandic (uppercase)
#,x	ₓ	Subscript characters
%?	õ	Tilde (above the letter)
%??	o̰	Tilde (below the letter)
%:	ö	Diaeresis (above the letter)

%::	ọ̈	Diaeresis (below the letter)
#.i	ɪ	Turkish i (dotless)
#;x	ð	Superior character
^'	ˈ	Inverse apostrophe
^a	ä	Umlaut ä
^A	Ä	Umlaut Ä
^o	ö	Umlaut ö
^O	Ö	Umlaut Ö
^u	ü	Umlaut ü
^U	Ü	Umlaut Ü
#(LU)	∧	And, logical
#(UOM)	⊇	Reflex superset
#(UTM)	⊆	Reflex subset
#(UE)	∞	Infinite
#(UGF)	≈	Approximately equal
#(UGR)	≳	Approximately greater than
#(UKL)	≲	Approximately less than
#(UGL)	≠	Not equal (with /)
#(UGLS)	≠	Not equal (with |)
#!x	ọ̲	Inferior character
^_	_	Underline
_		Underline (CTC for hard space)
#(VER)	∪	Union
#(VM)	⋃	Union
#(VMG)		Union (large)
#..	→	Reference arrow
#(IW)	¤	Currency, international
#(WI)	∢	Angle
#.[	⌜	Upper left corner
#.]	⌝	Upper right corner

#(WU)	√	Radical
#(WUG)	$\sqrt{}$	Radical (large)
#.z	ȝ	Yogh, Old/Middle English (lowercase)
#.Z	Ȝ	Yogh, Old/Middle English (uppercase)
^1	¹	Digits, small superscript
%<	ô	Circumflex (above the letter)
%<<	o̭	Circumflex (below the letter)
#(ZWR)	␣	Intervening space

Code tables

Survey:

General remarks

a) Concerning the character code used in TUSTEP

The input and internal representation of a single character in TUSTEP are based on the international ASCII code (DIN 660003, International Reference Version = 7-bit code according to the ISO-norm 646).

To make sure that all accent marks are coded in a unified manner, the ASCII characters grave accent, circumflex and tilde (in hexadecimal code 60, 5E and 7E) are not used for this purpose. However, the ASCII grave accent character is used in the Editor as a marker character on the screen and may therefore not occur in the data. The ASCII circumflex character is used as a shift character.

For more efficient use of comptuer storage space, the 7-bit ASCII-code has been extended to 8 bits. Characters marked by a preceding "^" (e.g. ^a for ä) on data input are internally represented by setting the 8th bit. The few exceptions to this are described below.

Some keyboards do not feature the characters: vertical bar, backslash, square brackets and curly brackets (braces). In their place, the following substitute codes have been selected.

^: vertical bar	^/ backslash (control character)
^< left square bracket	^(left brace
^> right square bracket	^) right brace

The characters "long vertical bar" and "backslash" (represented as printed characters, not as control characters) are normally endoded as "^|" and "^\", respectively. Substitute codes are also available for keyboards lacking these characters, which are thus entered as:

^; long vertical bar	^, backslash

Not every internal code which can be generated by marking a character with "^" corresponds to a printable character. For example, "^m" does not correspond to any character in the Latin alphabet, whereas in the Hebrew alphabet it represents the "final Mem".

The international EBCDIC code is the basis for the internal representation of characters in TUSTEP under IBM operating systems. This is the code which results from converting ASCII into EBCDIC based on the conversion table used by the IBM operating system MVS when reading ASCII tapes (cf. IBM: MVS/370 Magnetic Tape Labels and File Structure Administration).

b) Concerning the code tables The following tables have been organized separately according to operating systems using the ASCII code and those using the EBCDIC code. The ASCII code is used in TUSTEP versions running under

DOS, UNIX and VMS

while the EBCDIC code is used for the

BS2000, MVS und VM/CMS versions.

The first table shows the codes in which the characters of the "7-bit TUSTEP character set" and the "8-bit TUSTEP character set" (see page 298 and 299) are stored in files. All other TUSTEP characters are stored as a combination of characters from the 7-bit and 8-bit TUSTEP character set.

The second table has been provided merely for reference purposes and has no further significance when using TUSTEP.

The remaining tables show the various codes for screen input/output as set with the command #DEFINE (see page 83). Under certain circumstances they are also used to convert character codes when importing data from SYSTEM files and exporting data to SYSTEM files with the command #CONVERT (see page 70). This code setting has no influence on the code in which the data in TUSTEP files have been stored.

ASCII code tables

1. Internal 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. International ASCII code for comparison

	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=grave 7E=tilde

3. Code "GERMAN" on display devices with German keyboard

	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=grave

4. Code "IBMPC" for screen display on IBM compatible PCs

	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	

The code "IBMPC" contains

- all characters of the 7-bit TUSTEP character set
- all characters of the 8-bit TUSTEP character also found in the extended ASCII code for IBM compatible PCs.

5. Code "CP437" for screen display on IBM compatibler PCs

	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	Ö				┘			09
0A			*	:	J	Z	j	z	è	Ü	¬	‖		┌		·	0A
0B			+	;	K	[	k	{	ï	¢						√	0B
0C			,	<	L	\	l	|	î	£					∞		0C
0D			-	=	M	]	m	}	ì		¡						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	

The code "CP437" contains

- all characters of the 7-bit TUSTEP character set
- all characters of the 8-bit TUSTEP character set which are also defined in code page 437 (see your DOS manual).
- accented letters defined in code page 437.
- all characters encoded with the control character "#" and "#(name)" and which are also defined in code page 437.

6. Code "CP850" for screen display on IBM compatible PCs

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
00			SP	0	@	P		p	Ç	É	á		└	ð	Ó		00
01			!	1	A	Q	a	q	ü	æ	í		┴	Ð	ß	±	01
02			"	2	B	R	b	r	é	Æ	ó		┬	Ê	Ô		02
03			#	3	C	S	c	s	â	ô	ú	│	├	Ë	Ò		03
04			$	4	D	T	d	t	ä	ö	ñ	┤	─	È	õ		04
05			%	5	E	U	e	u	à	ò	Ñ	Á	┼		Õ	§	05
06			&	6	F	V	f	v	å	û		Â	ã	Í		÷	06
07			'	7	G	W	g	w	ç	ù		À	Ã	Î	þ		07
08			(	8	H	X	h	x	ê	ÿ	¿	©		Ï	Þ		08
09			)	9	I	Y	i	y	ë	Ö	®			┘	Ú		09
0A			*	:	J	Z	j	z	è	Ü	¬	‖		┌	Û	·	0A
0B			+	;	K	[	k	{	ï	ø					Ù		0B
0C			,	<	L	\	l	|	î	£					ý		0C
0D			-	=	M	]	m	}	ì	Ø	¡	¢			Ý		0D
0E			.	>	N	^	n		Ä	×	«			Ì			0E
0F			/	?	O	_	o		Å		»	┐	¤				0F
	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	

The code "CP850" contains

- all characters of the 7-bit TUSTEP character set
- all characters of the 8-bit TUSTEP character set that are also defined in code page 850 (see your DOS manual).
- accented letters defined in code page 850. sind
- all characters encoded with the control characters "#" and "#(name)" and which are also defined in code page 850.

7. Code "DECMCS" für VT100 (and compatible) 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			¡	±	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	Œ	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	Ÿ	í	ÿ	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	

The code "DECMCS" contains

- all characters of the 7-bit TUSTEP character set
- all characters of the 8-bit TUSTEP characters set that are also defined in the "DEC Multinational Character Set" (see your VT100 manual).
- Accented letters defined in the "DEC Multinational Character Set".
- all characters encoded with the control characters "#" and "#(name)" and also defined in the "DEC Multinational Character Set".

8. Code "ISO8859" for XTERM windows (in X-Windows)

	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					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	

EBCDIC code tables

1. Internal 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. International EBCDIC code for comparison

	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=grave A1=tilde

3. Code "GERMAN" for display devices with a German keyboard

	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=grave

4. Code "EBCDIC" for display devices with a US EBCDIC keyboard

	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=brave 4A=cent 5F=log.Not 6A=broken line A1=tilde

TUSTEP code conversion from ASCII to 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 code conversion from EBCDIC to 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 sort order in TUSTEP

In the table below, the individual characters are listed in the standard ascending order in which they are sorted (i.e. unless otherwise specified by using parameters). Character adjacent to one another in the same column will be treated equally.

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: (	19: <	29: {	67: z ^z Z ^Z
10:)	20: = ^=	30: | ^|	

A p p e n d i x

Using the PC as a terminal

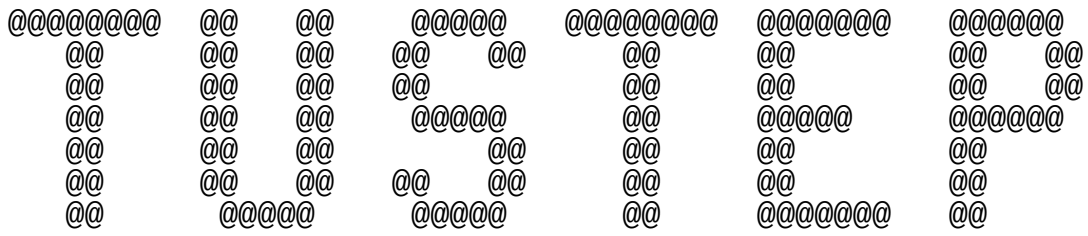
A PC can be employed as a terminal for a work station or mainframe. If TUSTEP is also worked with in this type of local network, it is vital that a matching keyboard layout is used in both of these situations. This means that the respective terminal emulation program must be configured appropriately. Yet not all such programs are flexible enough to provide a completely identical keyboard layout.

One widely-used terminal program is KERMIT. It is capable of handling asynchronous connections as well as connections via Ethernet. With KERMIT, the keyboard layout can be so defined that there are no differences to the one used when working with TUSTEP on a PC.

Each TUSTEP version for a work station or a mainframe features the file "kermit.dos", which contains the necessary definitions for the keyboard layout. This file can be copied to the PC. The TUSTEP keyboard layout can then be activated with the KERMIT command "take kermit.dos". This command can also be written at the end of the file "mskermit.ini" so that it will be automatically executed each time KERMIT is called up.

The file "kermit.dos" must match the TUSTEP version being used at the workstation or mainframe. This file is therefore not included in the DOS version of TUSTEP. Users should note that this file may change with a new version of TUSTEP. Should any problems with the keyboard layout arise, it may due to the fact that the latest version of this file has not been installed on the PC.

If a PC is used to run TUSTEP on another computer via a terminal emulation program, usually the same code can be set as that used when working on the PC locally. For more details, refer to the notes regarding code settings as outlined in the command #DEFINE on page 83.



Tübingen

System of Text Processing Programs

Program

C O L L A T E

1993

TÜBINGEN UNIVERSITY - CENTER FOR DATA PROCESSING

Department of Literary and Documentary Data Processing

Brunnenstrasse 27, D-72074 Tübingen

Survey:

Command:

#COLLATE

Specifications:

SOURCE	= file		Name of the input file containing the basic text
MODE	= -STD-	*	Each CORRECTION file contains the correcting instructions for a single text version
	= CUMULATED		The CORRECTION file contains all correcting instructions for all individual text versions.
ERASE	= -	*	If the LISTING file already contains data, they are to be retained.
	= +		If the LISTING file already contains data, they are to be erased beforehand.
PARAMETER	= file		Name of the file with the parameters
	= *		Parameters follow the command line and are ended by *EOF
CORRECTION	= file		Name of the file containing the correcting instructions (with correction key); for MODE=-STD- more than one file name is allowed.
LISTING	= -STD-	*	The generated listing is to be written into the standard LISTING file.
	= file		Name of the file to which the generated listing is to be written

Features:

With this command, a listing of the differences between one or several text versions and a basic text can be generated for printing in lines synoptic to the basic text. The differences must be provided in form of correcting instructions containing a correction key, as generated by the TUSTEP programs COMPARE and/or PRESORT. The deviant words (variants of the different text versions) are listed in synoptical lines under the respective words of the basic text. Identities between the basic text and the other text versions as well as between the variants themselves will be marked.

General remarks

In order to obtain an overview of the differences (variants) contained in several versions of the same basic text, it can be helpful to collate them into a single list. The differences contained in these versions must be present in the form of correcting instructions including a correction key as generated by the TUSTEP programs COMPARE or PRESORT.

The imput for the program COLLATE are the file containing the basic text and the file(s) containing the variants. Differences are marked in the form of correcting instructions. These correcting instructions must contain a correction key.

The LISTING file will successively list a single line of basic text, followed by the corresponding text of the other text versions. In this list, for each line of the the basic text in which variants have been detected, the variant text is located under the corresponding word or phrase of the basic text. Words which are identical both in the basic text and in the other versions - or words which have identical versions - are marked as such.

Furthermore, if a certain wording is identical in different versions, subsequent occurrences of this wording can be replaced by a reference to the version in which it first appeared.

MODE=-STD-

In this mode, all correcting instructions for each text version are located in a file of their own. Up to 9 files may be specified. The order in which lines containing variants from the basic text will appear depends on the order in which the respective files are specified.

MODE=CUMULATED

In this mode, all correcting instructions for the individual text versions (max. 49) are located in a single file. The correcting instructions must be sorted in ascending order according to the correction key. Parameter SW must specify the sorting values of those text versions that are to be collated. The order in which lines containing variants from the basic text will appear depends on the order in which the sorting values have been specified. In addition, parameter VKZ must provide an identification code for all text versions specified by parameter SW.

Parameters

Values given in [ ] refer to the type of parameter employed. The various types of parameters are described in the "Parameters" chapter of "TUSTEP Basics". Default values are shown in < >.

Selecting data

If the entire input data are to be processed, none of the following parameters need to be specified.

In case only a certain portion of the basic text is to be processed, this can be defined by the following parameters.

BER Specifies an area ("page.line-page.line") or an initial position ("page.line") if the basic text is not to be processed in its entirety. [xI]

MAX For trial runs, this specifies the maximum number of lines in the basic text that are to be processed. [I] <999999>

Parameters for LISTING file settings

Of the following parameters, parameter DRT must always be specified.

DR Printer output control

Four numerical values may be specified here:

1st value: columns <1>

Number of columns to appear side-by-side on each page.

2nd value: left margin <0>

Number of blank spaces to the left of the first column.

3rd value: width <132>

Number of characters per column

4th value: width between columns <0>

Number of blank spaces between columns

DRZ Additional specifications for printer output control [I]

Three numerical values may be specified here:

1st value: header text <3>

Number of lines for the header (including line spacing between the header and body text).

2nd value: column height <60>

Number of lines per column (excluding header and footer lines).

3rd value: footer text <0>

Number of lines for the footer (including blank lines between footer and body text).

KT Character string to be printed as a header at the top of every page [II]
<"file name" xx. xxx. xxxx xx.xx xxxxxx>

To insert the current date, enter "xx. xxx. xxxx" or "xx.xx.xx" at the appropriate position. Positions for the current time may be indicated by "xx.xx" and for the page number by "xxxxxx" (2 to 6 "x"s; but with at least as many positions as required for the page number). If "- xxxxxx -" is specified for the page number, the page number will appear centered between the minus signs; the minus signs to the left and right of the page number will be separated from the page number by one blank space. However, the date, time and page number can be inserted only one time each.

If a character string begins with "*:", the rest of the character string serves as a header for every text column. If a numeral *n* is entered in place of the asterisk, the rest of the character string is used as a header for the *n*th column. If the numeral specified is 0, the rest of the character string is used for the entire line. If a character string does not begin as just described, "0:" is assumed (standard value).

The following rules determine which line of the header is used by the character string: A character string which is designated for an entire line will be printed at the start of a new line (starting with the first line). A character string which is designated for a particular column will be printed in the same line as the preceding character string, unless this line contains text meant for an entire line, for the same column, or for a column further to the right. In this

case, the character string will be printed in the next line.

Each character string can be positioned at three positions in the line with the formatting instructions "@z" and "@/":

flush left @z centered @/ flush right

The individual character strings will appear flush left, centered and flush right on the page. Any single string may be omitted, with the formatting instructions preceding the second and third character string also being omitted.

FT Character string (analogous to parameter KT) to be printed as a footer at the bottom of every page. [II]

The date, time or page number may be specified in the footer only if it has not already been specified for the header.

PR Specifies whether all lines of the basic text are to be recorded in the LISTING file, or only those lines for which at least one correcting instruction is present in the CORRECTION file. [I] <0>

0 = LISTING file will contain only those lines of the basic text for which there is at least one correcting instruction
1 = LISTING file will contain all lines of the basic text

DRT Printing device for which the data are to be prepared. This parameter is obligatory. [XI]

The types of available printers depends on the actual computer being used. To obtain a list of these, use the command

#LIST,PRINTERS.

Formatting the line-by-line synoptic output

In the LISTING file, the corresponding text of the different versions will be displayed under each line of the basic text (including page and line number).

For MODE=-STD-, lines which contain the text of other versions will be preceded by the version's identification code, or, if the correcting instruction does not contain one, the sorting value of this version.

For MODE=CUMULATED, a version's identification code will be printed in front of the lines containing the text of that version. The version identification code is specified in the two following parameters. The correcting instructions are assigned

to the individual text versions in accordance with the sort values contained in the correction key. Each occurring text version must be given a sorting value (using parameter SW) and its own version identification code (using parameter VKZ). Any version identification code given in the correcting instructions will be ignored.

SW Sort values which are contained in the correction key. [I]

Correcting instructions having an unspecified sort value will be skipped. In this case, the corresponding error message will appear at the end of collation.

This parameter is obligatory when MODE=CUMULATED

VKZ Text strings (parallel to parameter SW) to be used as version identification codes.

This parameter is obligatory when MODE=CUMULATED

Positions where the wording of a version is identical to that of the basic text are marked by an equals sign "=" instead of repeating the text of the version.

Should the user desire another character or the actual identical wording instead of the equals sign, this can be specified in the following parameter.

GLT Character to be used to denote identical basic text and version text [xI] <=>

If the parameter GLT is specified without providing any character, the identical wording of a version will be displayed.

Should the output of identical wording be desired for certain versions only (instead of the equal sign or the optional symbol specified in the GLT parameter), use the following parameter.

GTZ Sorting value of the versions whose text should be displayed, even where it is identical to that of the basic text. [I]

Omissions in the text of the versions (relative to the basic text) are displayed by the corresponding number of blanks. If another character is to be used instead of a blank, specify this in the following parameter.

LCK Character used to designate omissions in the version text. [xI] < >

Differing text (variant) is displayed wherever the text of a version does not match that of the basic text. Should more than

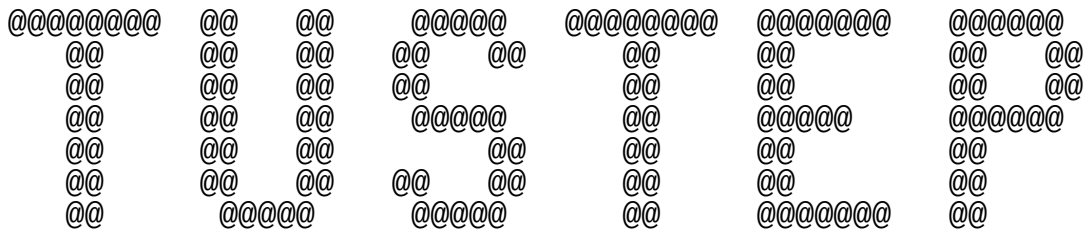
one variant happen to have identical text, the second and any subsequent occurrence of the variant text can be replaced by a reference to the first occurrence (instead of displaying identical version text). This can be specified in the following parameter.

GLV Specifies what should be displayed when variants are identical [I] <0>

0 = Display wording of the variant.
1 = Refer to first occurance of variant text. The version identification code or sorting value of the version containing the first occurance of the variant is placed in pointed brackets at every point where the wording would be repeated.
2 = As in 1. However, if the text version located directly above the present variant has the same wording, a double quotation mark (") is used instead of the reference.

Alphabetical list of parameters

* * * * *



Tübingen

System of Text Processing Programs

Program

C O M P A R E

1993

TÜBINGEN UNIVERSITY - CENTER FOR DATA PROCESSING

Department of Literary and Documentary Data Processing

Brunnenstrasse 27, D-72074 Tübingen

Survey:

Command:

#COMPARE

Specifications:

VERSIONA	= file		Name of the input file containing text version A
VERSIONB	= file		Name of the input file containing text version B
MODE	= T		The differences between the texts are to be listed
	= K		The generated correcting instructions are to be listed
	= ...		Printer for which the listed data containing text differences are to be prepared. The types of available printers depends on the actual computer being used. To obtain a list of these, use the command #LIST,PRINTERS. The printer can also be selected with parameters.
ERASE	= -	*	Do not erase data in the CORRECTION file and in the LISTING file
	= +		Erase data in the CORRECTION file and LISTING file
PARAMETER	= -	*	No parameters
	= file		Name of the file containing parameters
	= *		Parameters to be entered after the COMPARE command and ended with *EOF
CORRECTION	= -	*	Do not record correcting instructions
	= file		Name of the output file to which the correcting instructions are to be recorded
LISTING	= -	*	No LISTING file
	= -STD-		The generated listing is to be written to the standard LISTING file.
	= file		Name of the file to which the generated listing is to be written.

Features:

This command is used to compare two text versions (A and B). The differences are listed in the file given for the specification LISTING. In addition, these differences can be written to the file given in the CORRECTION specification in the form of correcting instructions, using the same conventions for correcting instructions as required for the program CORRECT. If these instructions are used to correct version A, version B will be obtained as a result (line division and numbering will be those of version A).

The line division of the versions may be completely different. Omissions and insertions up to the length of one typed page can be identified by the program automatically. The program can handle omissions and insertions of any length, provided these are identified and indicated by the user. It is also possible to compare only certain parts of a file.

General remarks

A systematic comparison of two or more versions of a text may be desirable for the following reasons:

- To record changes made in a text (eg. a protocol of corrections made when using the TUSTEP program EDIT). In this case, the result of the comparison is a printed list of differences.

- To locate errors in texts which have been written twice for semiautomatic error-detection and correction purposes. In order to carry out semiautomatic error-detection and correction, the program records the differences between the two texts in a file in the form of correcting instructions.

- To locate and reference a variant by tradition for critical text editions. Here the results of the comparison are recorded in more detailed form, allowing the user to merge the results of individual comparisons of different versions to the basic text and process them further.

Mode of operation

Each time COMPARE is activated, two text versions may be compared with one another. If the differences contained in more than two texts are to be ascertained, each additional version must be compared separately with the basic text (version A). Differences recorded by the comparison of two versions can be further processed by other TUSTEP programs.

The two text versions to be compared may vary in line format and contain omissions or insertions. These are automatically recognized by the program (with a corresponding amount of excess CPU time involved) as long as each text difference is no longer than approximately one standard typewritten page. As an alternative, the user may mark them as such and the program will treat them accordingly, with no waste of excess computing time.

The differences found by each comparison of two text versions can be recorded either in the LISTING file for a subsequent printout, or in a text file in the form of correcting instructions. The format of these correcting instructions may be specified by various parameters, depending on how these instructions are to be used later.

Comparison protocol

The comparison protocol will be written to the LISTING file. The protocol method can be set with the specification MODE:

In MODE=T, or when a printer has been given for the MODE specification, the comparison protocol will list the following for every line in which a difference between the two versions occurs: first the text of version A and below that the text of version B. Differences between the two versions are marked between the two lines as follows:

- an omission is marked by a "-" placed under the character in the first line which is omitted in the second one.
- an insertion is marked by a "+" placed above the character in the lower line which has been added to the upper line.
- a replacement is marked as an omission in the upper line and an insertion in the lower line. Positions where "+" und "-" would coincide are marked by a "*"

Example:

```
  1.1      Sample of     2 versions of a short text
      -->            +++   ********      -----
  1.1      Sample of the 2 variants of a        text
```

In MODE=K, for each line in which a difference is noted, the text of version A will be printed. Differences between the two versions are printed below this line in the form of correcting instructions.

Example:

```
  1.1       Sample of two versions of a short text

                 1.1        1.1,2+the
                 1.2        1.1,4=variants
                 1.3        1.1,6-
```

Structure and use of generated correcting instructions

Correcting instructions will be written to the CORRECTION file independently of the selected MODE. If not specified otherwise by additional parameters, the correcting instructions generated by COMPARE contain the components expected by the TUSTEP program CORRECT: the area of the text in version A to be corrected, the type of correction ("-", "+", "=", "*" for "delete", "insert", "replace" and "comment", respectively), and the "corrective text" (text of version B) resulting from the correcting instructions.

Using these correcting instructions with the help of the program CORRECT to correct text version A results in an identical copy of text B (apart from the exact line division and record numbering). If the TUSTEP programm COMPARE is used for semiautomatic error detection and correction, the correcting instructions resulting from the two copies of the text must be altered before the necessary corrections can be carried out in a subsequent CORRECT run. (These alterations will normally consist of deleting those correcting instructions for variants where text version A was the correct one.)

For more complex tasks such as those encountered in the preparation of critical editions, additional elements can be included in the correcting instructions, namely: the correction key, the text of verson A affected by the correcting instructions ("original wording") and its surroundings ("context"), position of corrective text in version B, as well as an identifying label for version B.

The correction key is needed to generate a common list of all differences present in more than two text versions. It can be generated by specifying the respective parameters.

If the TUSTEP program COLLATE is used to generate a synoptic line-by-line printout of more than two text versions, an (arbitrary) version A should be compared with the remaining versions, one after another, each of which is to be specified as version B. The differences must be stored as correcting instructions (with a correction key) in CORRECTION files.

The original wording and context of the variants in text version A are additionally required if, for example, the variants are to be listed in a form commonly used in an apparatus criticus.

To accomplish this, the correcting instructions, which result from comparing the other text versions with the same basic text, must be supplied with correction keys and the other necessary elements. By using the TUSTEP program PREPARE SORT followed by the program SORT, these correcting instructions can be put in the order required for an apparatus criticus: position, wording of text variants, sorting value of the source containing the respective variant. After sorting, text variants can be listed together with the wording of the basic text ("lemma") by using the program GENERATE INDEX [GINDEX].

Hyphenation

Hyphenation is not in effect when texts are compared. A "-" written as the last character in a line (= input record) is used to hyphenate words if the next-to-last character of the line is also a "-", or if the next-to-last character is a letter and the third-to-last character is not a control character ($, &, @, \, _, #, %).

Note for hyphenated German texts: when hyphenation is off, a previously hyphenated "ck" (written as "k-" and "k") will not be restored to "ck".

Parameters

Values in [ ] refer to the type of parameter employed. The various types of parameters are described in the "Parameters" chapter of "TUSTEP Basics".

Values in < > refer to default settings.

In addition to the parameters described below, those used to define character groups and string groups can also be used. [v]

Selecting data

If the entire contents of both files are to be compared in their existing order, none of the following parameters are necessary.

If only a specific area of both versions is to be compared, or if both versions are to be compared from a given position only, the area or the position can be specified by the parameter BER if these specifications are identical for both versions. If the specifications for area or position of the text portions to be compared are not identical in both versions, the parameter ASP must be used to select the individual text portions.

BER Definition of an area ("page.line-page.line") or a starting point ("page.line"). This parameter is only used when not comparing the entire contents of both input files. [xi]

If both input files are segment files and if each has a segment of the same name which is to be compared, the name of the segment can be specified instead of the area.

This parameter can only be used when the record numbers of both files are in ascending order. Only one of the parameters ASP and BER may be used in the same program.

If the two files are to be compared only from certain starting points, or if only certain areas of the two files are to be compared, this can be specified with parameter ASP. This parameter allows different area and position specifications for both versions.

Specifying starting points in a document is also recommended for comparing texts in their entirety if large differences exist between the two files (expecially if they contain extensive omissions or insertions) and thus would cause both a waste of computing time when locating parallel positions, and inaccurate matching of the differences found.

ASP Starting points, or areas of text, to be compared. More than one definition may be given. Definitions are to be separated by an apostrophe. [xI]

A complete definition for an area to be compared in both versions consists of the area definition for version A, and the corresponding area definition for version B, connected by a "=".

An area definition consists of a starting position and an end position, the two positions being connected by a "-" and taking the form p.l[/d][,w]-p.l[/d][,w], where p stands for a page number (up to 6 digits), l for the line number (up to 3 digits), d for the distinction number (up to 3 digits), and w for the word number (up to 2 digits). Entries in [ ] are optional.

The specification for version B (including the preceding "=") can be omitted if the area definitions for version A and B are identical.

When defining two adjoining areas for the same version, the specification for the end position of the first area may be omitted (including the preceding "-"). Likewise, the definition of the end position can be omitted if the area being specified extends to the end of the file. Such specifications without end positions are called starting points.

If only starting points are specified (or pairs of them, each connected by a "="), the two versions will be compared starting with the first (pair of) starting point(s). This means that both files will be broken up into the same number of consecutive areas which will be collated one after the other, as determined by the number of corresponding (pairs of) starting points. This arrangement may be useful for the aforementioned reasons, even though the entire text of both versions is to be compared.

The area definitions are written to the CORRECTION file in the form of a correcting instruction (correction type=comment, correction character = "*") and placed in front of the correcting instructions which apply to the respective area.

This parameter is to be used only with files containing record numbers in ascending order. Only one of the parameters BER and ASP can be used in the same program.

If the text of version B consists only of fragments which are to be compared with the corresponding areas in version A, it may be advisable to define these areas in the text of version B itself instead of using parameters.

KBA Character strings used to denote the area definitions in version B. [XI]

The area specifications must immediately follow the character string specified in the parameter KBA and are to be written with the syntax used for area definitions in the parameter ASP. If text follows in the same line, this text must be separated from the area definition by a blank space. An area of version A text defined in this way will be compared to the version B text which follows the area definition, including all text up to the next area definition. Explicit area definitions for version B text are not expected by the program.

Please note that area specifications denoting the corresponding areas of version A must be present for the entire text of version B. This means that the beginning of the text of version B must therefore be preceded by such an area specification.

The area specifications are to be written into the CORRECTION file in the form of a correcting instruction (correction type= comment, correction character = "*") and placed in front of the correcting instructions which apply to the respective area. The correction character "*" is followed by a number giving the relative position in the parameter KBA of the *character string* used for denoting the respective area specification in version B.

Parameters for LISTING file

If no protocol output is desired (LISTING=-), none of the following parameters are necessary.

If a protocol is desired, the parameter DRT must be specified if not printer has been previously given in the MODE specification.

DR Printer output control [I]

Four numerical values may be specified here:

1st value: columns <1>

Number of columns to appear side-by-side on each page.

2nd value: left margin <0>

Number of blank spaces to the left of the first column.

3rd value: width <132>

Number of characters per column

4th value: width between columns <0>

Number of blank spaces between columns

DRZ Additional specifications for printer output control [I]

Three numerical values may be specified here:

1st value: header text <3>

Number of lines for the header (including line spacing between the header and body text).

2nd value: column height <60>

Number of lines per column (excluding header and footer lines).

3rd value: footer text <0>

Number of lines for the footer (including blank lines between footer and body text).

KT Character string to be printed as a header at the top of every page [II]
<"file name" xx. xxx. xxxx xx.xx xxxxxx>

To insert the current date, enter "xx. xxx. xxxx" or "xx.xx.xx" at the appropriate position. Positions for the current time may be indicated by "xx.xx" and for the page number by "xxxxxx" (2 to 6 "x"s; but with at least as many positions as required for the page number). If "- xxxxxx -" is specified for the page number, the page number will appear centered between the minus signs; the minus signs to the left and right of the page number will be separated from the page number by one blank space. However, the date, time and page number can be inserted only one time each.

If a character string begins with "*:", the rest of the character string serves as a header for every text column. If a numeral *n* is entered in place of the asterisk, the rest of the character string is used as a header for the *n*th column. If the numeral specified is 0, the rest of the character string is used for the entire line. If a character string does not begin as just described, "0:" is assumed (standard value).

The following rules determine which line of the header is used by the character string: A character string which is designated for an entire line will be printed at the start of a new line (starting with the first line). A character string which is designated for a

particular column will be printed in the same line as the preceding character string, unless this line contains text meant for an entire line, for the same column, or for a column further to the right. In this case, the character string will be printed in the next line.

Each character string can be positioned at three positions in the line with the formatting instructions "@z" and "@/":

flush left @z centered @/ flush right

The individual character strings will appear flush left, centered and flush right on the page. Any single string may be omitted, with the formatting instructions preceding the second and third character string also being omitted.

FT Character string (analogous to description in parameter KT) to be printed as a footer at the bottom of every page. [ıı]

The date, time or page number may not specified in the footer if already included in the header.

PR Specifies whether all lines of the basic text are to be recorded in the LISTING file, or only those lines for which at least one correcting instruction is present in the CORRECTION file. [ı] <0>

0 = LISTING file will contain only those lines of the basic text for which there is at least one correcting instruction
1 = LISTING file will contain all lines of the basic text

DRT Printing device for which the data are to be prepared. This parameter is obligatory. [xı]

The types of available printers depends on the actual computer being used. To obtain a list of these, use the command

#LIST,PRINTERS.

Parameters for correcting instructions

If no correcting instructions are to be recorded (CORRECTION=-), the following parameters are not necessary.

If correcting instructions are to be recorded, yet none of the following parameters are specified, the correcting instructions will contain only the position or area of the corresponding text in version A, the correction character (-, = +, or *) and any corrective text involved.

A text position or area is indicated by a single position in the form: p.l[/d][,w], or by a beginning and end position joined by a minus sign (-): p.l[/d][,w]-p.l[/d][,w], where p stands for the page number (up to 6 digits), l for the line number (up to 3 digits), d for the distinction number (up to 3 digits) and w for the word number (up to 2 digits). Elements in [ ] are optional.

Each correcting instruction is written to the CORRECTION file as a single record. Parameter KFZ should be used if a line break that occurs in the corrective text (version B) is to remain recognizable as such in the correcting instruction. This ensures that the correcting instruction is split at the same point where the line break splits the text in version B.

KFZ Specifies whether continuation lines should be generated for correcting instructions whose corrective text contains a line break [I] <0>

0 = No continuation lines are to be generated
1 = Continuation lines are to be generated to match line breaks in version B.

When a correcting instruction is split in this fashion, the continuation lines appearing in the CORRECTION file will contain the correction key (only if the parameter SW has been specified), the correction code "+" (to indicate the continuation of a correcting instruction), and the continuation of the corrective text.

The following parameters must be employed if a more expanded form of correcting instruction is desired. Consult the section "Structure of a data record in the CORRECTION file" for further details concerning what kind of information that may be included in a correcting instruction.

In order to keep track of which version a correcting instruction refers to (even after correcting instructions generated by comparing the basic text with various text versions have been merged into a single file), a marker can be assigned to a correcting instruction. This marker is specified in the parameter VKZ.

VKZ Character string used to mark correcting instructions [II]

The character string specified here will be enclosed in parentheses and inserted after the position indicator of the version A text affected by the correcting instruction.

If the text in version A affected by the correcting instruction ("original wording") and any surrounding text ("context") is

also to be included in the correcting instruction, the parameter UMG must be used.

UMG Specifications for output of original wording and its context. [I]

Three numerical values may be specified here:

1st value: context preceding original wording <0>

Number of words located before the original wording that are to be included in the correcting instruction.

2nd value: Original wording <0>

Maximum number of words of the original wording that will be included in the correcting instruction. If the original wording contains more words than specified here, it will be shortened by omitting surplus words from the middle of the text. If not specified otherwise in the parameter UMK, the omitted words will be marked by ".....".

Third value: Context after the original wording <0>

Number of words located after the original wording that are to be included in the correcting instruction.

In the correcting instruction, the original wording will be separated from the context (assuming the latter has been specifed by the parameter UMG) by "::". If the original wording in the correcting instruction has been shortened (cf. parameter UMG), "....." will be inserted in place of the omitted words. To make any changes in these default settings, the desired character strings can be specified in parameter UMK.

UMK Specifies the character strings used to separate the context from the original wording, and the character string used to denote any necessary omissions in the original wording. [II </::/...../::/>

The first character string separates the original wording from the preceding context; the third character string separates the original wording from the context following it.

If no context is to be included either before or after the original wording (as specified in parameter UMG), the corresponding first and/or third character strings will be omitted from the correcting instruction.

The second character string specified here will be inserted into the correcting instruction if the original wording has to be shortened.

The position indicator of text from version B ("corrective text") will only be displayed if this is requested in the following parameter. This option enables the text to be used later for reference purposes (for example, for compiling corresponding indexes).

STB Specifies whether the position indicator from version B (corrective text) is to be included in the correcting instruction. [I] <0>

0 = No position indicator of corrective version text
1 = Include position indicator of corrective version text

The position indicator of the corrective text, surrounded by angle brackets, will be placed before the correction code in the same form as that used to indicate the position of the original wording.

The correction key will be included in the correcting instruction only if so specified in parameter SW. This is necessary if the correcting instructions are to be sorted, or if they are to be processed for printing with the TUSTEP program COLLATE.

SW Specifies a sorting value [I] <0>

The program expects a number from 0 to 99 as a sorting value. It is a component of the correction key and can be used together with other sorting criteria to arrange correcting instructions in any desired order.

When this parameter is specified, the correction key will be inserted into the correcting instruction. A schematic description of the structure of a correction key can be found in the section "Structure of the correction key" on page 390.

Parameters for comparing text versions

When differences between two text versions are encountered during the comparison, the program must be able to mark off lengthy text additions or omissions while at the same time keeping those parts of the two versions which correspond to each other, despite their differences, in a word-to-word order whenever possible.

If the differences between the two versions are limited to orthographic idiosyncrasies or to the use of abbreviations in the text, the program is able to carry out a better correlation of the compared text of the two versions if the following parameters are used.

Specified in these parameters are characters which may be regarded as equivalent, be ignored, or stand for abbreviations. However, any differences based on these characters will always be included in the output.

GLZ Characters or groups of characters [vI]

Characters which are to be treated as equivalent must first be defined as a character group (parameter type v). The identification for each character group must be specififed in the parameter GLZ.

For example, if i and y are to be considered equivalent and, at the same time, c, g, k and q are also to be equivalent, two separate character groups must be defined. The first character group contains i and y, the second one c, g, k and q. By entering separate group identifications in the parameter GLZ, the characters i and y will represent one group of equivalent characters, and the characters c, g, k and q will represent a second group of equivalent characters.

Furthermore, this parameter can be used to specify characters which should be either ignored or treated as abbreviations. If this is the case, the corresponding specifications must be given in the parameters IGN and ABK respectively.

IGN Position number specifying which character (or character group ID) given in the parameter GLZ that is to be ignored during comparison. A character group identification will be counted as one character. [I]

The parameter IGN expects one number only. Therefore, if more than one character is to be ignored, all characters to be ignored must first be defined as a character group; the identification of this character group must then be specified in parameter GLZ.

ABK Position number specifying which character (or character group ID) given in parameter GLZ that is to be regarded as an abbreviation character. A character group identification will be treated as one character. [I]

The parameter ABK expects one number only. Therefore, if more than one character is to be used as an abbreviation character, all such characters must first be defined as a character group; the identification of this character group must then be specified in the parameter GLZ.

Alphabetical list of parameters

Structure of a data record in the CORRECTION file

```
| KS StA (VKZ) n [ KL :: OWL :: KR ] <StB> KZ KTxt

    ***                                      ** ****
SW  ++
VKZ      +++++
UMG            ++++++++++++++++++++
STA                                +++++
```

*** Data which are always generated
+++ Data generated by specifying the parameters listed in the left column

Abbreviations:

KS	Correction key
StA	Position indicator in version A
VKZ	Correcting instruction marker
n	Number of "]" in the context and original wording
KL	Context to the left of the original wording
OWL	Original wording (from version A)
KR	Context to the right of the original wording
StB	Position indicator of version B
KZ	Correction code
KTxt	Corrective text (from version B)

The original wording and its context are surrounded by brackets. If this text includes right brackets, the number of such brackets contained in the context and original wording is shown by the number *n* preceding the left bracket marking the context. In this way, the program is able to recognize the right bracket which marks the end of the original wording and its context. If the context and original wording contain no right brackets, *n* is omitted.

If the number representing *n* occurs immediately after the position indicator StA -- this being the case when the correcting instructions are not identified by the character string in parentheses (see parameter VKZ page 384) -- the character string "()" will be inserted before *n* in order to separate it from the preceding position indicator.

The character string"::" located between the context and the original wording can be substituted by any other character string (see: parameter UMK, page 385).

Structure of a correction key

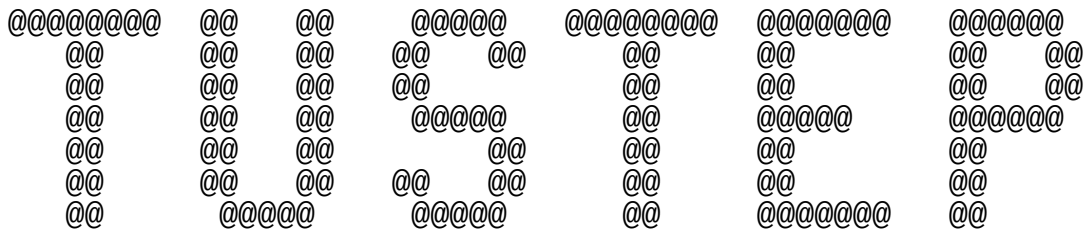
The correction key consists of a total of 44 characters and is structured as follows:

APO	EPO	KA	SW	FNR	POS

APO	1	17	Starting position of the corrective text
	1	6	page number
	7	3	line number
	10	3	distinction number
	13	2	word number
	15	3	character number
EPO	18	17	End position of the corrective text
	18	6	page number
	24	3	line number
	27	3	distinction number
	30	2	word number
	32	3	character number
KA	35	2	Type of correction:
	35	1	0 = error 1 = page-line-word-character 2 = page-line-word 3 = page-line
	36	1	0 = error 1 = comment (*) 2 = delete (-) 3 = replace (=) 4 = insert (+)
SW	37	2	Sorting value (version number)
FNR	39	3	Continuation number (for continuation lines)
POS	42	3	Position of the correction code (*, -, =, +) in the correcting instruction

The first number specifies the position of each character within the correction key, the second number specifies the number of characters.

The character number is not used by the TUSTEP program COMPARE; this number is used by the PREPARE SORT program to create the correction key.



Tübingen

System of Text Processing Programs

Program

C O P Y

1993

TÜBINGEN UNIVERSITY - CENTER FOR DATA PROCESSING

Department of Literary and Documentary Data Processing

Brunnenstrasse 27, D-72074 Tübingen

Survey:

Command:

#COPY

Specifications:

SOURCE	= file		Name of the file containg the data to be copied.
	= -STD-		The standard TEXT file contains the data to be copied.
DESTINATION	= file		Name of the file to which the data are to be copied. More than one file name is allowed.
	= -STD-		The data are to be copied into the standard TEXT file.
MODE	= -STD-	*	Retain numbering of records if possible.
	= +		Renumber the records
	= -		Retain numbering of records in every case
	= S		DATA file contains sort units.
ERASE	= -	*	If the DESTINATION file already contains data, they are to be retained.
	= +		If the DESTINATION file already contains data, they are to be erased beforehand.
PARAMETER	= -	*	No parameters
	= datei		Name of the file containg parameters
	= *		The parameters follow the command and are ended by *EOF.
DATA	= -	*	SOURCE file contains records in their entirety.
	= file		Name of the file containing the text part of each record.
	= -STD-		The standard DATA file contains the text part of each record.
LISTING	= -	*	No trace
	= +		Trace is to be written into the journal.
	= -STD-		Trace is to be written into the standard LISTING file.
	= file		Name of the file into which the trace is to be written.

Features:

With this command files can not only be copied, but also processed in a various number of ways.

Using parameters it is possible to:

- examine data

- select data

- rearrange and supplement individual text parts

- replace character strings

- process calendar dates

m,2- calculate numerical values contained in the text

- control the form of output

Parameters

Values in [ ] refer to the type of parameter employed. The various types of parameters are described in the "Parameters" chapter of "TUSTEP Basics".

Values in < > refer to initial settings.

In addition to the following parameters, other parameters can be used to define character groups and strings. [v]

In certain parameters, text parts can be selected for further processing by means of beginning and/or end markers as well as markers that serve as left and right parentheses for selecting the desired part. The specific functions of these parameters are also described at the end of the "Parameters" chapter in "TUSTEP Basics".

For parameters which have to be specified explicitly for each pass, the number of the pass must be entered flush-right in columns 6 and 7; if this number is not given in such a parameter, pass 1 will be assumed.

Unless specified otherwise, parameters will be processed in the order they are described below. When more than one pass is to be processed, the pass number sets the priority order if the program flow has not been defined otherwise with the appropriate parameters. Thus, the parameters specified when starting the program may be given in any order. However, if continuation lines are used in a parameter, these must be entered in logical order immediately after the first line of the parameter. To avoid any mistakes or oversights, it is recommended to specify parameters in the order they are to be processed.

INITIALIZATION

Defining the special selector switches

SWS Special selector switches [I]

Five numerical values (for selector switches 1 to 16) may be specified here:

1st value: selector switch for input data <0>

The specified selector switch is canceled upon every input and set when the input data are exhausted.

2nd value: selector switch for partitioning basic text <0>

The specified selector switch is canceled prior to every partitioning of the basic text and is set when the basic text is exhausted.

3rd value: selector switch for test runs (trace) <0>

The control output (trace) for each individual part of the program will be printed only when the specified selector switch has been set.

4th value: selector switch for repeating basic text <0>

If the specified switch is set when jumping to program part 0, the same part of the basic text as used in the previous pass again becomes the working text. The selector switch will be cancelled automatically. If the basic text (or part of it) has not yet become the working text, i.e. immediately after it has been read from the SOURCE file, the selector switch will be ignored.

5th value: selector switch for double passes <0>

When the end of the SOURCE file is reached for the first time, the file will be read once more if the specified selector switch has been set.

Note: This feature can be used, for example, to add up the individual numerical values in the first pass (without having the data written to the DESTINATION file) to establish their total number. The corresponding procent figures can then be calculated for each numerical values and added to the text during the second pass.

Determining the initial setting for selector switches

WSG Selector switches which are to be set. [I]

Up to 16 numerical values (numbers of the selector switches 1 to 16) may be specified. Selector switches which remain unspecified will be cancelled.

Retrieving selector switch settings from the command level

WSH Selector switches whose initial settings are to be determined by the settings of the command level selector switches (these are selector switches which can be either set or cancelled with the command #SWITCH). [I]

Up to 7 numerical values (numbers of the selector switches 1 to 7) may be specified. Selector switches not specified here will be set or cancelled according to the specifications given in the parameter WSG.

Determining the initial value of the running number

LNB Specification in case the running number, which can be inserted with parameter LNR, is not to begin with 1. [I]

Two numerical values can be specified:

1st value: Value to be added to the output number (= 1 + the number of text units already outputted). <0>

2nd value: Value to be added to the (separate) running number. <0>

Determining the initial stored comparison texts

VTV Initial stored comparison text. The text part specified here is stored in memory at the start of the program; it will serve as the stored comparison text for the pass for which this parameter has been given. [II]

Determining the initial contents for replacement texts

ETV Initial replacement text. The replacement text part specified here is stored in memory at the start of the program; it will serve as the replacement text for the pass for which this parameter has been given. [II]

Determining initial values for variables

R Compute statements for inital variable settings

Described in the section "Compute statements" on page 435 ff.

Redefining program flow

The program's logical operation is schematically illustrated in the table "Logical program structure" on page 446. Program flow can be redefined by using either the parameters SPW, SPN, SPJ and SPn (n=0,1,3,...,8,9) or the parameter SPR. If parameter SPR is used, none of the other parameters may be used. Parameter SPR must be used instead of the others if more than 20 passes are defined.

SPW Jump table for a jump in case a selector switch specified in WS+ is set, or in case a selector switch specified in WS- is not set. [I]
<see "Logical program structure">

SPN Jump table for a jump in case not all comparison conditions have been met. (no). [I]
<see "Logical program structure">

SPJ Jump table for a jump in case all comparison conditions have been met (yes). [I]
<see "Logical program structure">

SPn Jump table for jump after completion of program part n (n=0,2,3,...,8,9). [I]
<see "Logical program structure">

SPR Jump table for all provided jumps in a single pass. [I] <see "Logical program structure">

This parameter can be used to specify in the nth pass (where n is entered flush-right in columns 6 and 7 of the parameter) those values that could be specified individually as an nth value with the parameters SP0, SPW, SPN, SPJ, SP2, SP3, ..., SP9. Here care should be taken to observe all jumps involved and to specify them in their correct sequence.

PROGRAMM PART 0 (Input only, no run specifications):

Selecting data on input

If the entire file is to be processed, none of the following parameters are necessary.

BER Definition of an area ("page.line-page.line") or a starting point ("page.line"). This parameter is only used when not processing the entire input file. [xI]

If a segment of a segment file is to be processed, the name of the segment can be substituted for the area.

This parameter can only be used when all record numbers of the file are in ascending order. Moreover, only one of the parameters BER, NR+ and NR- can be used in the same program.

NR+ Page-line numbers (also areas) of records to be processed. More than one definition may be given; definitions are to be separated from one another by an apostrophe. [xI]

Records will be read in the same order as given in the page-line numbers (or areas). The specified records may overlap or be repeated any number of times.

This parameter is only to be used when the record numbers in the file are in ascending order. Moreover, only one of the parameters BER, NR+ und NR- may be used at one time.

NR- page-line numbers (also areas) of records to be ignored. Area definitions must be separated from one another by an apostrophe. [xI]

The page-line numbers (or areas) must be given in ascending order. Areas may not overlap.

This parameter can only be used when the record numbers in the file are in ascending order. Moreover, only one of the parameters BER, NR+ und NR- may be used in the same program.

MAX For test runs, specifies the maximum number of text units to be either read or included in the output. [I]

Two numerical values may be given here:

1st number Maximum number of text units to be read
<999999>

2nd number Maximum number of text units for output <999999>

Organizing records into text units

In case each input record contains a complete text unit, the following parameters should not be used. Otherwise, the parameters ANR, AA and/or AE can be used to organize more than one record into a text unit.

If one of the following four parameters is specified, any blanks located either before or after the input record will be eliminated before the parameter is evaluated.

When records are being organized into a text unit, a blank will be inserted between each input record; no blank will be inserted at positions where hyphenation protect has been specified (see parameter STR).

ANR Specifies whether successive records, whose record numbers either partially or completely match, are to be organized into a text unit. [I] <0>

One numerical value may be specified:

0 = Do not organize records into a text unit on the basis of record numbers.
1 = Organize successive records having the same page number into a text unit.
2 = Organize successive records having the same page and the same line number (regardless of the distinction number) into a text unit.
3 = Organize all successive records with the same record number into a text unit.

If 0 is specified (default setting), records will be organized into text units on the basis of the two following parameters only. If one of the values 1 to 3 is specified, the resulting text units may be broken down further on the basis of the two following parameters.

AA Character strings placed at the beginning of a record (after leading blanks have been eliminated) which mark the start of a text unit. [VIIIa]

AE Character strings placed at the end of a record (after any trailing blanks have been eliminated) which mark the end of a text unit. [VIIIb

STR Hyphenation [I] <0>

0 = Input data are not hyphenated
1 = Rejoin hyphenated words during input

Here a hyphen is considered to be a "-" which (after trailing blanks have been eliminated) is the last character in an input record if the second-to-last character is also a "-" or a letter and the third-to-last character is not a control character ($, &, @, \, _, #, %).

When hyphenation is turned off, a hyphenated "ck", which according to German hyphenation is written as "k-" and "k", will not be restored to its "ck" form.

Replacing character strings on input

X Pairs of character stings (and exception strings). On input, the first character string of a pair will be replaced in the text by the pair's second character string. [x]

Replacement is carried out separately in each input record, even if a text unit consists of more than one record (see parameters ANR, AA and AE).

Before the character string is replaced, a blank is added to both the beginning and end of the record (after any previous blanks have been removed); both blanks are removed after replacement.

Before replacement takes place, the program first checks to see whether the record begins with the character string specified in the parameter AA, or whether it ends with the character string specified in the parameter AE.

If hyphenated words are to be rejoined (parameter STR), the program checks whether the record ends with a hyphen after replacement.

PROGRAMM PART 0 (basic text ==> working text):

Partitioning the basic text

GTU Character strings marking the positions where the basic text is to be partitioned. The respective character string is considered part of the text which follows it. [ɪx]

PROGRAMM PART 1 (inquiry + comparison):

Branching according to a selector switch position

WS+ The jump condition is met if one of the specified selector switches is set (see also: parameter SPW). [I]

Up to 16 numerical values (i.e., the numbers of the selector switches 1 to 16) may be specified.

WS- The jump condition is met if one of the specified selector switches is not set (see also: parameter SPW). [I]

Up to 16 numerical values (i.e. the numbers of the selector switches 1 to 16) may be specified.

Branching according to a variable

VSP Jump addresses to be skipped to according to variable S8 (cf. page 436). If the value of S8 is less than 1 or larger than the number of specified jump addresses, no jump will be carried out. [I]

Branching according to a marker

PV Starting position for KEN and the comparison text (or the starting position from which the comparison text is to be extracted from the working text), in case this is not equal to 1, or differs from the starting position specified in the parameter POS. [I] <1st numerical value given in the parameter POS>

KEN Character strings which serve as (beginning) markers for text units. If the working text begins with a specified character string, the jump will be made to the corresponding jump address specified in the parameter KSP. If this address is missing, or the text unit begins differently, no jump will take place. [VIIIa]

KSP Jump address corresponding to the markers given in KEN [I]

Preparing the comparison text

If the working text is to used in its entirety as the comparison text, the following selection parameters (PV to VI) are not necessary.

PV Beginning position for the comparison text.

(Described above under parameter PV on page 403)

AV Character strings marking the beginning of the text part of the working text which is to be used as the comparison text. [ɪx]

EV Character strings marking the end of the text part of the working text which is to be used as the comparison text. [ɪx]

(V Character strings which serve as a left parenthesis when selecting the comparison text (if AV and/or EV have not been specified) or when eliminating text parts already selected by AV and/or EV. [ɪx]

)V Character strings which serve as right parenthesis when selecting the comparison text (if AV and/or EV have not been specified) or when eliminating text parts already selected by AV and/or EV. [ɪx]

VI Index for the parameters AV, EV, (V and)V [ɪ]

Two numerical values may be specified:

1st value: Specification analogous to parameter AEI <1>
2nd value: Specification analogous to parameter KLI <0>

XV Pairs of character strings (and exception strings). The first character string of a pair will be replaced in the comparison text by the pair's second character string. [x]

Branching according to the frequency of character strings

ZFZ Character strings which are to be counted in the comparison text. The number is saved in the variable S5 (cf. page 436). For the character string which first occurs in the comparison text, its position in this parameter is stored in the variable S6 (cf. page 436). [Ix]

ZSP Jump addresses to be skipped to according to the number of character strings counted in parameter ZFZ. If this number is 0, or larger than the number of specified jump addresses, no jump will be carried out. [I]

Selecting the text units

In case certain text units are to be selected for further processing, the following parameters are used to specify the criteria for selection.

Selection is carried out in five steps. In order for a text unit to be considered selected at the end, it must be selected in every step for which parameters have been specified.

S t e p 1

In this step, the comparison text can be checked to see whether it consists of only certain characters or character strings.

ZFP Character strings which may comprise the comparison text. [Ix]

If the comparison text contains characters or character strings which cannot be composed from those specified in the parameter, the text unit will not be selected.

S t e p 2

In this step, the comparison text can be checked to see whether it
- consists/does not consist of a particular character string
- contains/does not contain a particular character string
- starts/does not start with a particular character string
- ends/does not end with a particular character string

If the comparison text matches one of the text parts specified in the parameters T+ or T+U, the text unit will be selected. In

this case, the text unit will not be checked further on the basis of the parameters remaining in this step. If no other parameters have been specified for this step, only these text units will be selected.

T+ Text parts, one of which must match the comparison text if the text unit is to be selected. [III]

No distinction is made between uppercase and lowercase letters.

T+U Text parts, one of which must match the comparison text if the text unit is to be selected. [III]

A distinction is made between uppercase and lowercase letters.

If the comparison text matches one of the text parts specified in the parameter T- or T-U, the text unit will not be selected. In this case, the text unit will not be checked any further on the basis of the remaining parameters in this step.

T- Text parts, none of which may match the comparison text if the text unit is to be selected. [III]

No distinction is made between uppercase and lowercase letters.

T-U Text parts, none of which may match the comparison text if the text unit is to be selected. [III]

A distinction is made between uppercase and lowercase letters.

The parameters ZF+, TA+ and TE+ can be used to specify conditions under which a text unit is to be selected. If one or more of these parameters are specified, the comparison text must meet at least one of the stated conditions if the text unit is to be selected.

ZF+ Character strings, of which at least one must occur in the comparison text if the text unit is to be selected. [IX]

TA+ Character strings, of which at least one must match the beginning of the comparison text if the text unit is to be selected. [VIIIa]

TE+ Character strings, of which at least one must match the end of the comparison text if the text unit is to be selected. [VIIIb]

The parameters ZF-, TA- and TE- can be used to specify conditions under which a text unit is not to be selected. If one or more of these parameters are specified, only one of the stated conditions need be met in order that the text unit will not be selected.

ZF- Character strings, none of which may occur in the comparison text if the text unit is to be selected. [IX]

TA- Character strings, none of which may match the beginning of the comparison text if the text unit is to be selected. [VIIIa]

TE- Character strings, none of which may match the end of the comparison text if the text unit is to be selected. [VIIIb]

If one or more of the parameters ZF+, TA+ and TE+ as well as one or more of the parameters ZF-, TA- and TE- are specified, the comparison text must fulfill at least one of the conditions stated in the parameters ZF+, TA+ and TE+ and may not meet any of the conditions stated in parameters ZF-, TA- and TE- in order for the text unit to be selected.

S t e p 3

In this step, the comparison text may be checked for the order and frequency of specified character strings.

ZF Character strings whose occurrence in the comparison text is to be checked for their order and frequency. [IX]

The character strings specified here must occur in the comparison text in the same order as given in this parameter. Otherwise the text unit will not be considered selected.

The parameters ZFM and/or ZFH can be used to specify the minimum or maximum number of times character strings specified in the parameter ZF may occur in the comparison text. Such character strings which occur more than once in the comparison text may be separated in the comparison text by text of any kind except for other character strings specified in the parameter ZF.

Variable S4 (cf. page 416) will record the relative position of the first character string in the comparison text in terms of its position as specified in parameter ZF. However, if one of the conditions in parameters parameters ZF, ZFM and ZFH is not met,

variable S4 will record the position of the first character string where this occurs.

ZFM Specifies the minimum frequency required for each character string in the comparison text. Frequencies must be given in the same order as the corresponding character strings in parameter ZF. [I] <0,0,0,...>

If one of the character strings fails to meet this condition, the text unit will not be selected.

ZFH Specifies the maximum frequency permitted for each character string in the comparison text. The frequencies must be given in the same order as the corresponding character strings in the parameter ZF. [I] <1,1,1,...>

If one of the character strings fails to meet this condition, the text unit will not be selected.

S t e p 4

In this step, the comparison text can be compared with a comparision text which has been previously stored. This can be used to check the respective alphabetical order of the two texts, or whether they are identical. The stored comparison text can be specified at the start of the program with parameter VTV. It can also be redefined (stored) in program part 5 each time the program is run.

VGL Specifies how the comparison text and the stored comparison text are to be compared with one another. [I]

One numerical value can be specified:

0 = Both texts must be identical.
1 = Both texts must be identical or the comparison text's alphabetical location must precede that of the stored comparison text.
2 = The comparision text's alphabetical location must precede that of the stored comparison text.
3 = Both texts must be identical or the comparison text's alphabetical location must follow that of the stored comparison text.
4 = The comparision text's alphabetical location must follow that of the stored comparison text.
n = Both texts may deviate by n percent of their characters.

VTB Comparison table for comparing the comparison text with the stored comparison text [VII]

S t e p 5

In this step, the first number occurring in the comparison text can be tested to see whether it has been provided with its corresponding check digit.

PZP Specifies the type and form of the check digit. [I]

Two numerical values can be specified:

1st number: Type of check digit

0 = Do not test check digit in the comparison text

1 = Check digit = Difference between the sum of the weighted digits and the next multiple of 11

2 = Check digit = Difference between the sum of the weighted digits and the previous multiple of 11

2nd number: check digit syntax

0 = Check digit immediately follows the number.

1 = Check digit is separated from the number by a "-"

For a description of how check digits are calculated, see "Check digits" (page 434).

PROGRAMM PART 2 (Processing):

Setting and canceling selector switches

WSS selector switches which are to be set. [I]

Up to 16 numerical values (selector switch numbers 1 to 16) may be specified.

WSL selector switches which are to be canceled. [I]

Up to 16 numerical values (selector switch numbers 1 to 16) may be specified.

WSU selector switches which are to be toggled. [I]

Up to 16 numerical values (selector switch numbers 1 to 16) may be specified.

Defining the working text

T Text which is to be used as the working text. [II]

If this parameter is given, the specified text will become the working text. Any position specifications given in the parameter POS will be ignored, i.e. the previous working text will be completely replaced starting at position 1.

PK Starting position from which the individual text parts are to be selected, if this value does not equal 1 or differs from the starting position specified in parameter POS. That part of the text unit located before this position will be copied in its original form. The first character string specified in the parameter ERG (if any) will be inserted at this point. [I] <1st numerical value of the parameter POS>

AKn Character strings which mark the beginning of the nth part (n=1,2,...,9) of the new working text. [IX]

EKn Character strings which mark the end of the nth part (n=1,2,...,9) of the new working text. [IX]

AEI Index for the parameters AKn and EKn
[I] <1,1,1,1,1,1,1,1,1>

One to nine numerical values can be specified:

1 = Selects the first text part marked with A/E (beginning with the beginning marker and ending before the following end marker, or at the end of the text unit).
If only A has been specified, the selected text part ends at the end of the text unit; if only the parameter E has been specified, the selected text part starts at the beginning of the text unit.
0 = Selects that part of the text unit which would be excluded by choosing 1.
3 = As in 1. However, this selects not only the first text part marked with A/E, but all text parts marked in this way (with the second text part starting with the beginning marker which follows the end of the first text part marked with A/E).
If only A is specified, the selected text part will begin at the last beginning marker ocurring in the text unit, and will end at the end of the text unit. If only E is specified, the selected text part will will start at the beginning of the text unit and end before the last end marker of the text unit.
2 = Selects that part of the text unit which would be excluded by choosing 3.

If one of the values 0 to 3 is specified, each beginning marker is counted as part of the text following it, while the end marker is not counted as part of this text. This treatment of beginning and end markers can be reversed by adding either 10 or 20 to the number chosen. If the value of 10 is added (i.e. by entering a number from 10 to 13), each beginning marker will not be counted as part of the following text; if 20 is added, each end marker will be counted as part of the preceding text. If 30 is added, the beginning marker is not counted as part of the following text and the end marker is counted as part of the preceding text.
When 2 or 3 is chosen, the program is instructed to search the text for more than one beginning marker, and will look for a new beginning marker starting at the first position after the end of the preceding text part. Therefore, for the values 2 and 3, the search for a new beginning marker starts at the first character of the preceding text's end marker, since the end marker is not part of the preceding text. Beginning and end markers may thus overlap in the text. If 20 or 30 is added to these values, the next beginning marker is searched from the character which follows the last position of the most recently found end marker, since this marker is counted as part of the preceding text.

(Kn Character strings serving as left parentheses for selecting the nth (n=1,2,...,9) part of the new working text (in case AKn and/or EKn has not been specified) or for eliminating text parts from the text part already selected with AKn and/or EKn. [IX]

)Kn Character strings serving as right parentheses for selecting the nth (n=1,2,...,9) part of the new working text (in case AKn and/or EKn has not been specified) or for eliminating text parts from the text part already selected with AKn and/or EKn. [IX; MSCAN]

KLI Index for the parameters (Kn and)Kn
[I] <0,0,0,0,0,0,0,0,0>

One to nine numerical values may be specified:

0 = Eliminates the parts of the text in parentheses (including the parentheses themselves). Missing parentheses are added logically at either the beginning or end of the text unit or text part which has already been selected.
1 = Selects all text parts which would be eliminated by option 0.
2 = As in 0, but unpaired parentheses are ignored instead of being logically provided with a complementary parenthesis.
3 = Selects those text parts which would be eliminated by option 2.

If the values 0 to 3 are specified, the parentheses themselves are considered part of the text in parentheses and are thus either eliminated along with the text or are kept with it. This treatment of left and right parentheses can be reversed by adding either 10 or 20 to the value chosen. If 10 is added (i.e. by entering a value from 10 to 13), each left parenthesis will not be counted as part of the text in parentheses. If 20 is added, each right parenthesis will not be counted as as part of the text in parentheses. If 30 is added, neither parenthesis will be counted as part of the text in parentheses.

ERG Text parts to be added at the beginning of the text unit, between the individual text parts selected by the parameters AKn,EKn and/or (Kn,)Kn, and at the end of the the text unit. If for a text part no selection parameter has been specified, the entire text unit (or that part of the text unit starting with the beginning position specified in the parameter POS or PK) will be copied. [II]

Replacing text parts

ATT Character strings marking the beginning of the text parts to be replaced. [IX]

ETT Character strings marking the end of the text parts to be replaced. [IX; MSCAN]

(TT Character strings to be used as a right parenthesis when selecting the text parts to be replaced (or after selecting such text parts with ATT and/or ETT). [IX]

)TT Character strings to be used as a left parenthesis when selecting the text parts to be replaced (or after selecting such text parts with ATT and/or ETT). [IX]

TTI Index for ATT, ETT, (TT and)TT [I]

Two numerical values may be specified:

1st number: Specification analogous to parameter AEI <1>
2nd number: Specification analogous to parameter KLI <0>

TTT Pairs of text parts [IV]

The first text part of each pair is checked to see if it is identical with the text parts chosen on the

basis of the above parameters. Here no distinction will be made between uppercase and lowercase letters. Each such text part of the working text will be replaced by the second text part of the respective text-part pair.

The parameters TTT and TUT cancel each other out when used in the same pass.

TUT Pairs of text parts. [IV]

As in parameter TTT, but a distinction is made between uppercase and lowercase when the check for matching text is made.

The parameters TTT and TUT cancel each other out when used in the same pass.

Inversion of text parts

AUM Character strings marking the beginning of the text parts to be inverted. [IX]

EUM Character strings marking the end of the text parts to be inverted. [IX]

NUM Character strings which are not to be inverted should they occur in the text parts to be inverted. [IX]

Calculating printing positions

DPB Type of output device (printer) for which the length of the working text is to be calculated in terms of the number of printing positions. [XI]

The types of available printers depends on the actual computer being used. To obtain a list of these, use the command
#LIST, PRINTERS.

The number of calculated printing positions is stored in the special variable S1 (see page 436 ff.).

Note to DOS version: due to memory requirement limitations, this parameter cannot be used.

Reading numerical values from the working text

PL Starting position from which the text part containing numerical values to be read either begins or is to be selected, if this value does not equal 1 or differs from the starting position specified in the parameter POS. [I] <1st numerical value of the parameter POS>

AL Character strings marking the beginning of the text part from which numbers are to be read. [Ix]

EL Character strings marking the end of the text part from which numbers are to be read. [Ix]

(L Character strings serving as left parenthesis for selecting the text part containing numbers to be read (in case AL and/or EL have not been specified), or for eliminating text parts already selected with AL and/or EL. [Ix]

)L Character strings serving as right parenthesis for selecting the text part containing numbers to be read (in case AL and/or EL have not been specified), or for eliminating text parts already selected with AL and/or EL. [Ix]

LI Index for parameters AL, EL, (L and)L [I]

Two numerical values may be specified:

1st number: Specification analogous to parameter AEI <1>
2nd number: Specification analogous to parameter KLI <0>

XL Pairs of character strings (and exception strings). The first character string of a pair will be replaced in the text part from which numbers are to be read by the pair's second character string. [x]

LIV I-variables into which numbers are to be read. [I]

A minus sign placed directly before a number to be read will be treated as a mathematical sign.

If fewer numbers are found than the number of I-variables specified in this parameter, the remaining I-variables will be set to 0.

LDN Specifications parallel to LIV, indicating how many decimal places located after the decimal point are to be read. [I] <0>

LIZ Specifications parallel to LIV, indicating whether Arabic numerals, Roman numbers or hexidecimal numbers are to be read. If Roman or hexidecimal numbers are selected, the corresponding specification in the parameter LDN will be ignored. [I] <0>

One numerical value can be given for each I-variable specified with paramter LIV:

0 = Number in Arabic numerals
1 = Roman numerals (letters: I,V,X,L,C,D,M)
22 = two hexidecimal numbers (= 1 Byte)
42 = four hexidecimal numbers (= 2 Bytes)

Executing compute statements

RR Instructions for compute statements, conditions and jumps.

Please refer to "Compute statements" page 435 ff.

Inserting numerical values in the working text

PE Starting position from which substitution/insertion (of numerical values, calculation numbers, the replacement text, the running number, the page number) is to be carried out, in case this is not equal to 1, or differs from the starting position specified in the parameter POS. [I] <1st numerical value given in the parameter POS>

EIN Character strings marking the beginning of the text part in which the numbers are to be replaced. [IX]

EIB Character strings marking the end of the text part in which numbers are to be replaced. [IX; MSCAN]

EIV I-variables, whose values are to replace the old numbers. [I]

A minus or plus sign located directly before a number to be replaced will be removed.

If less numbers are found in the working text (or in the text part defined with EIN and/or EIB) than variables specified, the numbers will be appended to

the end of the working text or inserted at the end of the defined text part (see, however, parameter EIK).

EDV Specifications parallel to EIV, indicating how many positions before the decimal point are to be filled out with blanks. [I] <0>

If the specified positions are to be filled out with leading zeros instead of blanks, this can be specified in parameter EIZ.

EDN Specifications parallel to EIV, indicating how many digits are to be positioned after the decimal point. [I] <0>

The values of I-variables can be inserted into the working text in different forms: as normal numbers (Arabic numerals), Roman numbers, or as a number with a check digit. The desired form can be selected with the following parameter.

EIZ Specifications parallel to EIV, indicating in which form the respective numerical value is to be inserted. [I] <0,0,0,...>

One numerical value can be specified for each I-variable specified with parameter EIV:

```
 0 = normal number (in Arabic numerals)
 1 = Roman numbers in lowercase letters
 2 = Roman numbers in uppercase letters
x3 = number immediately followed by its check digit
x4 = number separated from its check digit by a "-"
 5 = as in 0, except that positive numbers will be
     filled out with leading zeros instead of blanks,
     should this has been specified by the parameter
     EDV.
26 = two hexidecimal numbers (= 1 Byte)
46 = four hexidecimal numbers (= 2 Bytes)
```

If a number with a check digit is to be inserted, the type of check digit being employed must also be specified. (See: "Check digits" page 434). Thus, when the values 3 or 4 are chosen, 10 or 20 must always be added to them: 10 is added when the check digit = the difference between the sum of weighted digits and the next multiple of 11; 20 is added when the check digit = the difference between the sum of weighted digits and the previous multiple of 11.

For information on calculating check digits, please refer to "Check digits" (page 434).

EIK Character strings marking the positions after which a number is to be replaced by a new value [Ix]

If the character string found in the working text has been specified as the nth character string with parameter EIK, each following number will be replaced by the I-variable given as the nth value in parameter EIV. A character string may occur more than once in the text, with the corresponding I-variable being inserted after each character string.

If no number is found in the working text up to the next character string after which a number is again to be inserted, the value of the nth I-variable will be inserted immediately before this character string. Correspondingly, this value will be inserted at the end of the working text (or at the end of the text part defined with EIN and/or EIB) if no such character string follows.

If parameter EIK has been specified, the following applies: I-variables specified with parameter EIV but for which no corresponding character string has been specified with parameter EIK will not be inserted. If the nth character string specified in parameter EIK is not found in the working text (or in a text part defined with EIN and/or EIB) the nth I-variable specified with parameter EIV will not be inserted either. The same applies when the nth character specified with parameter EIK is not a search string, but an exception string.

Replacing character strings

PX Beginning position from which replacement is to be carried out, in case this is not equal to 1, or differs from the beginning position specified in the parameter POS. [I] <1st numerical value of the parameter POS>

AXX Character strings marking the beginning of the text part in which character strings are to be replaced (permissible only with the parameter XX). [Ix]

EXX Character strings marking the end of the text part in which character strings are to be replaced (permissible only with the parameter XX). [Ix]

(XX Character strings serving as left parenthesis when defining a text part in which no character strings are to be replaced (after it has been defined with the parameters AXX and/or EXX, if present). (Permissible only with parameter XX) [Ix]

)XX Character strings serving as right parenthesis when defining a text part in which no character strings are

to be replaced (after it has been defined with the parameters AXX and/or EXX, if present). (Permissible only with parameter XX) [Ix]

XXI Index for the parameters AXX, EXX, (XX and)XX [I]

Two numerical values can be specified:

1st number Specification analogous to parameter AEI <1>
2nd number Specification analogous to parameter KLI <0>

XX Pairs of character strings (and exception strings). The first character string of a pair will be replaced by the pair's second character string. [x]

Replacing character strings with conditions

PX Starting position for replacing character strings

(Described in parameter PX on page 417)

ERN Character strings, after which replacement is to take place. [Ix]

ERB Character strings, up to which (exclusively) replacement is to take place. [Ix]

ERS Character strings to be replaced (not permissible in combination with parameter XXB; parameter EZF is required here). [Ix]

EZF List of text parts which are to replace those specified in ERS. In the text unit, the nth found character string specified in ERS will be replaced by the nth (in case variable S7 does not equal 0: by the S7+nth) text part in EZF, or by the last character string in EZF if fewer character strings have been specified. [II]

XXB Pairs of character strings (and exception strings). The first character string of each pair will be replaced by the pair's second character string. [x]

ERZ Additional specifications for replacement [I]

Four numerical values may be specified:

1st number: Number of character strings to be replaced. <9999>

2nd number: Specifies how often a character string given in ERN is to be searched (with each search beginning after the previously found character string) in order to replace the number of character strings specified with the first number. If ERN has not been specified, each replacement starts at the beginning of the text part. <1>

3rd number: Indicates how many character strings in ERN must have been found before replacement is carried out for the first time. <1>

4th number: Indicates whether a character string specified in ERS is to replaced by one specified in EZF, or whether it is to be eliminated, in case another character string specified in ERS immediately follows in the text unit, or in case the character string is located at the end of the text unit <0>

0 = character string to be replaced, not eliminated
1 = character string to be eliminated

Inserting previously defined text parts

The following parameters can be used to insert a previously defined "replacement text" into the working text. The replacement text can be preallocated with parameter ETV at the start of the program; it can be redefined for each run of the program in program part 6.

PE Starting position for inserting the replacement text.

(Described for parameter PE on page 415)

EEN Character strings after which the replacement text is to be inserted. [IX]

EEB Character strings, up to which (exclusively) the replacement text is to be inserted. [IX]

ETE Character strings which are to be either replaced by the replacement text or after which the replacement text is to be inserted. [IX]

EEZ Additional specifications for inserting the replacement text. [I]

Two numerical values can be specified:

In the following, the term "found character string" refers to a character string that has been given in parameter ETE as a search string and which has been found in the working text (or in a text part defined by EEN and/or EEB).

1st number: How often and where to insert <0>

0 = The replacement text is inserted once (for the first character string found). Parameter ETE is obligatory here. If no character string specified in ETE is found, the working text remains unchanged.

1 = as in 0, with the exception that if no character string specified in ETE is found, or the parameter ETE has not been specified, the replacement text will be inserted at the beginning of the working text, or at the beginning of the text part defined with the parameter EEN.

2 = as in 1, except that the replacement text will be inserted at the end of the working text, or at the end of the text part defined with the parameter EEB.

3 = as in 0, except that the replacement text will replace every found character string (instead of just one replacement).

4 = A portion of the replacement text will be inserted for every found character string: the first portion for the first character string found, the second portion for the second string found, etc. The separator character for subdividing the replacement text must be specified with parameter ETR.

2nd number: Replace or insert <0>

0 = The replacement text is to replace the respective character string.
1 = The replacement text is to be inserted behind the respective character string.

ETR Separator characters used to subdivide the replacement text. [IX]

If 4 has been specified as the first numerical value in parameter EEZ, the following applies:

A portion of the replacement text will be inserted for every found character string: the first portion for the first character string found, the second portion for the second string found, etc. Should the replacement text consist of less portions than the number of found character strings, blank strings will be inserted; if it consists of more portions than found character string, the remaining portions will be ignored.

For all other cases:

If the replacement text consists of more than one part, for each additional part the text unit will be copied to the end and the corresponding part of the replacement text will be inserted.

Inserting a running number

PE Starting position for inserting a running number.

(Described under parameter PE on page 415)

LNR Character strings, after which the running number is to be inserted. [ɪx]

The initial value of the running number can be set with parameter LNB (see page 398).

If a separate running number is to be inserted instead of the output number (= 1 + number of text units already outputted), this must be specified with the second numerical value in parameters LNZ below.

LNZ Additional specifications for LNR [ɪ]

Four numerical values can be specified:

1st number: <0>

0 = The running number is to be inserted once (in case parameter LNR has been specified).
1 = The running number is to be inserted any number of times.
2 = As in 1, except the running number is to be increased by 1 before being inserted a further time (not possible when the second number of this parameter =0).

2nd number: <0>

0 = Instead of the running number, the output number (= 1 + number of text units already outputted) will be inserted.
1 = Increase running number (before insertion) by 1 if it is to be inserted into the text unit or would be inserted by being a multiple of this parameter's fourth numerical value.
2 = Do not alter running number
3 = Increase running number by 1 (regardless of whether it will be inserted or not; if so, it will be increased before insertion)
4 = Set running number to 0 (before any insertion)
5 = Set running number to 1 (before any insertion)

3rd number: Number of decimal places <0>

Fill out running number to the left with the specified number of zeros.

4th number: Step size <1>

MOD Running number is to be inserted only if it is a multiple of the number specified here.

Inserting / reading the page-line number

PE Starting position for inserting the page-line number.

(Described under parameter PE on page 415)

SNR Character strings, after which the page and, if applicable, line number is to be inserted. [ɪx]

SNZ Additional specifications for SNR [ɪ] <0,0,0>

Three numerical values can be specified:

1st number: Insert how often? <0>

0 = The page(-line) number is to be inserted once.
1 = The page(-line) number is to be inserted any number of times.

2nd number: What to insert <0>

0 = Insert page number only
1 = Insert page-line number
2 = Insert page-line-distinction number

3rd number: Number of the output file <0>

If this number is not equal to 0, the page(-line) number will not be inserted, but read from the text. This number will be given to the next record to be outputted to the nth file (n = the number specified here), provided that the last record already outputted to this file does not have a higher number.

Defining positions (columns)

POS Positions, in case the text unit is not to be processed starting from the first character position, or in case characters are to be eliminated from certain character positions. [I]

Three numerical values can be specified:

1st number: Starting position <1>

Starting position from which the text unit is to be processed.

2nd number: End position <99999>

End position, up to which characters are to be either saved or eliminated (no effect when this parameter's third value =0).

3rd number: Mode <0>

0 = The characters from position 1 (inclusively) to the position specified by this parameter's 1st number (exclusively) are to be left unchanged.
1 = The characters preceding the position specified by this parameter's 1st number and following the position specified by this parameter's 2nd number are to be eliminated after processing is completed.
2 = The characters at the position specified by this parameters 1st number up to and including those at the position specified by this parameters 2nd number will be eliminated after processing is completed.

Note: The first number of this parameter is also used as the default value for parameters PV, PK, PL, PX, PE. If different settings are desired, the desired position specifications (e.g. 1) must be additionally made with each of these parameters.

PROGRAMM PART 3 (Output, with pass specification):

Printing messages

MLD Message (text part) to be written to the LISTING file together with the text unit. [xI]

Each text part will be printed in a separate line. The first character of each text part must be a feed character ("-" for new page, "1", "2", to "7" for line feeds of 1, 2 to 7 lines).

If the character "@" immediately follows the feed character in the first text unit, this will be regarded as an error message. The program will nevertheless continue, but will end with errors upon completion. This is of consequence when the error stop option is in effect (see the command #ERROR STOP).

If the message contains x...x, the x-characters will be replaced by the value of variable S9 (cf. page 437).

In case this parameter has been specified, the text unit will not be written to the DESTINATION file if ZD has not also been specified.

Determining the DESTINATION file

ZD Destination file to be used for output in the present pass [I] <1>

PROGRAMM PART 3 (Output, without pass specification, valid for all passes):

Determining the beginning and end of lines

During output, the working text can be divided into lines. For this purpose, character strings can be specified, before or after which a line break is to be carried out.

ZA Character strings, before which a new line is to be started (beginning of line). [Ix]

ZE Character strings, after which a new line is to be started (end of line). [Ix]

If lines created with the help of parameter ZA and ZE are so long that they must be subdivided into more than one record on the basis of the specifications concerning record length in parameter SL (see below), the following parameters can be used to specify certain character strings before or after which lines may be subdivided. A line will be divided at such a character string only when division at the next character string would fail to produce a line corresponding to the desired maximum line length.

ZAB Character strings marking the start of a new line in case subdivision is necessary. Subdivision will be carried out before the character string. [IX]

ZEB Character strings marking the end of a line if a new line is necessary. Subdivision will be carried out after the character string. [IX]

Determining the beginning and end of pages

During output, record numbering can be advanced to the next page for lines starting with a certain marker or after lines ending with a certain marker.

SA Character strings marking lines whose record number is to be advanced to the next page number. [VIIIa]

The marker is effective only when it is located directly at the beginnning of the line. If necessary, parameter ZA can be used to accomplish this.

SE Character strings marking the lines after which the next record number is to be advanced to the next page number. [VIIIb]

The marker is effective only when it is located directly at the end of the line. If necessary, parameter ZE can be used to accomplish this.

If a new page is to be started for a line on the basis of both parameters SA and SE, the line number will be advanced by one page only, instead of two.

Inserting blank lines during output

During output, a blank line may be inserted before lines starting with a certain marker, or after lines ending with a certain marker.

LZV Character strings used to mark lines before which a blank line is to be inserted. [VIIIa]

The marker is effective only when located directly at the beginning of the line. If necessary, parameter ZA can be used to accomplish this.

LZN Character strings used to mark the lines after which a blank line is to be inserted. [VIIIb]

The marker is effective only when it is located directly at the end of the line. If necessary, parameter ZE can be used to accomplish this.

If a blank line is to be inserted between two lines on the basis of both parameters LZV and LZN, only one blank line (instead of two) will be inserted.

Replacing character strings during output

XXX Pairs of character strings (and exception strings). On output, the first character string of a pair will be replaced by the pair's second character string. [x]

Eliminating blanks at the beginning and end of lines

BLU Specifies whether blanks at either the beginning or end of a line are to be eliminated before output. [I] <0,0>

Two numerical values can be specified:

1st number: Blanks at the beginning of a line

0 = Do not eliminate blanks at the beginning of a line
1 = Eliminate blanks at the beginning of a line

2nd number: Blanks at the end of a line

0 = Do not eliminate blanks at the end of a line
1 = Eliminate blanks at the end of a line

Eliminating blank lines on output

LZU Specifies whether blank lines are to be eliminated. [I] <0>

One numerical value can be specified:

0 = Include blank lines in the output
1 = Eliminate blank lines

Blank lines inserted with parameters LZV or LZN will always be printed.

Determining record length

SL Length of output records [I]

In case this parameter is not specified and the line division of the input data cannot be retained, lines will not be subdivided.

Two numerical values can be specified:

1st number: Maximum length of lines to be subdivided because they are longer than the second number specified in this parameter. <99999>

LNG2 Maximum length of lines which are not to be subdivided. <100 or value of 1st number if larger than 100>

Before output, a line having more characters than that specified with the second number will be subdivided into output records. The line will be subdivided into records having a maximum length specified by this parameter's first number. The line is divided at any blank which is not preceded by a "-" (exception: a blank preceded by " -", since this "-" represents a dash and can therefore not be confused with a hyphen). If such a division point cannot be found within the number of characters specified by this parameter's first number, the next possible division point will be selected.

Determining record numbering

NR Specifications for numbering output records [I]

Three numerical values can be specified:

1st number: Page number with which output is to start (999999 for the next available page). <MODE=-STD-: 999999; MODE=+: 1>

2nd number: Maximum number of records per page.
<MODE=-STD-: 1000000; MODE=+: 60>

3rd number: Increment for numbering when a new record number must be allocated.
<MODE=-STD-: 10; MODE=+: 1000>

PROGRAMM PART 4 (working text ==> basic text):

Redefining the basic text

No parameters

PROGRAMM PART 5 (comparison/working text ==> stored comparison text)

Defining stored comparison texts

VTZ Specifies for which passes the comparison text stored in this pass is to be valid. [I]

PROGRAMM PART 6 (working text ==> replacement text):

Defining replacement texts

ETZ Specifies for which passes the replacement text stored in this pass is to be valid. [I]

PROGRAMM PART 7 (working text ==> stored text):

Replacing/supplementing the stored text by the working text

MTD Specifications for defining the stored text [I] <0>

One numerical value can be specified:

0 = Stored text to be replaced by the working text
1 = Working text to be appended to the existing stored text.
2 = Working text to be inserted before the existing stored text.

PROGRAMM PART 8 (stored text <==> working text):

Exchanging stored text and working text

No parameters

PROGRAMM PART 9 (stored text ==> working text):

Replacing/supplementing the working text by the stored text

MTH Specifications for retrieving stored text [I] <0>

Four numerical values can be specified:

0 = Stored text to replace working text
1 = Stored text to be appended to existing working text.
2 = Stored text to be inserted before the existing working text.

If 3 is added to these values (i.e., if 3, 4 or 5 is specified instead), the stored text will be erased after it has been moved.

FINAL STEP

Printing calculated numerical values

RRR Compute statements for calculating the final results, which can be outputted using the following parameters.

For a more detailed description, see "Compute statements", page 435 ff.

DDD Text to be outputted to the LISTING file. A new line will be started at each separator character; the first character after the separator character will be interpreted as a line feed character. [II]

DIV I-variables to be inserted in the text specified in parameter DDD (at the position x...x) before printing. [I]

Values will be inserted in the specified order at positions marked by a sequence of at least 2 "x"s. Each position must have enough "x"s to accomodate the value of the corresponding I-variable; if too few "x"s have been provided, they will remain in the text. If more I-variables have been specified than the number of positions marked with "x"s, the excess I-variable values will be ignored.

DDN Specification parallel to I-variables specified in parameter DIV: establishes how many digits are to be positioned after the decimal point. [I] <0,0,0,...>

Saving selector switch settings to the command level

WSR Selector switches whose settings are to be saved to the selector switches at the command level (i.e., selector switches which can be either set or cancelled using the command #SWITCH). [I]

Up to 7 numerical values (number of the selection switch 1 to 7) may be specified. Selector switches not specified here remain unaltered.

Alphabetical List of Parameters

The character "n" in the parameter abbreviation stands for integers 1 to 9 (e.g. AKn stands for AK1, AK2 to AK9).

Check digits

Check digits are calculated on the basis of module 11 with a weighting of 10 to 2. In other words, the last digit (ones) of a number is multiplied ("weighted") by 2, the next-to-last digit (tens) is multiplied by 3, etc. The resulting products are then added. Type 1 check digits are arrived at by calculating the difference of this sum and the next higher multiple of 11. Type 2 check digits are arrived at by calculating the difference of this sum and the preceding multiple of 11. If the sum of the products itself is divisible by 11, it has the check digit 0. If the check digit has the value 10, the Roman number X is used.

Example:

```
  Number:                          1    9    8    5
  Weight:  10    9    8    7    6    5    4    3    2
 Product:   0  + 0 +  0 +  0 +  0 +  5 + 36 + 24 + 10 = 75
```

The type 1 check digit for the number 1985 is thus 2 (77-75), the type 2 check digit for the same number is 9 (75-66).

Compute statements

Compute statements must be separated from each other by a semicolon. Blanks are insignificant and may be inserted at any place to make the compute statements more readable. A compute statement may also be interupted at any place and continued on the next line (with the same parameter identification).

In the following, a few terms will be defined which are subsequently used for defining individual compute statements.

Variables

A variable is a storage location which is identified by a name and contains an integer value. Each of these variables has a unique name by which the contents of the variable can be accessed. This access is made by placing the name of the variable at the location where its numerical value is defined (cf. "Value assignment") or required (cf. "Arithmetic expression"). The names of the variables are predefined and may not be chosen freely. At the beginning of the program, all variables are assigned the value zero. The largest numerical value that may be saved as a variable depends on the computer being used; in most cases this is the value 2 147 483 647.

- H-variables

  10 H-variables (Help variables) are defined with the names H0, H1 to H9.

- I-variables

  I-variables (100 storage locations) can be addressed in two ways: either as a simple variable with the names I0, I1 to I99, or as an indexed variable in the form of I(index). In this case, the index may be any arithmetic expression (see below). However, the value of the arithmetic expression must lie between 0 and 99 (inclusively).
  These variables differ from H and B variables in that they can be assigned numerical values not only by value assignment (see below), but also by reading numbers located in the data to be processed (cf. "Reading numerical values" page 414). In addition, the numerical values stored in these variables can not only be used in arithmetic expressions (see below), but can also be inserted into the data to be processed (cf. "Inserting numerical values" page 415 f).

- B-variables

  These variables may only be addressed as indexed variables in the form B(index). In this case, the index may be any arithmetic expression (see below). However, the value of the arithmetic expression must lie between 0 and 99 (inclusively).

- S-variables

Each S-variable (special variables) has its own significance:

S0 The contents of this variable show the present length of the working text. If the value n is subtracted from S0, the last n characters of the working text will be eliminated. Please note that the contents of S0 may not be smaller than 0. The working text cannot be enlarged.

S1 A numerical value is stored in this variable every time the parameter DPB (see page 413) is processed. It represents the number of printing positions needed by the working text when it is printed. Altering the contents of this variable has no significant effect.

S2 Current value of the running number (cf. "Inserting a running number" page 421). If necessary, this variable may be altered.

S3 The contents of this variable show the length of the last comparison text. The comparison text is defined each time program part 1 is executed as long as this program part has not already been exited using one of the parameters SPW (or the corresponding jump address of parameter SPR), VSP or KSP. Altering the contents of this variable has no significant effect.

S4 A numerical value is stored in this variable each time the parameter ZF (cf. page 407) is processed. Its contents show the parameter position of the character string first found in comparison text. If, however, the order of character strings found in the comparison text does not correspond to the order of character strings in the parameter, or if the conditions required by parameters ZFM and ZFH are not met, the contents of variable S4 show the position in parameter ZF of the character string which either violated the character string order or did not meet the given conditions. Altering the contents of this variable has no significant effect.

S5 A numerical value is stored in this variable each time the parameter ZFZ (cf. page 405) is processed. Its contents show the number of character strings found in the comparison text. Altering the contents of this variable has no significant effect.

S6 A numerical value is stored in this variable each time the parameters ZFZ (cf. page 405) is processed. Its contents show the position in the parameter of the character string first found in the comparison text. Altering the contents of this variable has no significant effect.

S7 The contents of this variable are evaluated each time the parameters ERS and EZF (cf. page 418) are processed.

S8 The contents of this variable are evaluated each time the parameter VSP (cf. page 403) is processed.

S9 The numerical value stored in this variable can be included in a message generated by parameter MLD (cf. page 424).

S10 The contents of this variable can be used to define which record number (page-line-distinction number) is to be stored in the variables S11 and S12 before executing the parameter RR. If S10 has the value 0, S11 and S12 will contain the record number of the present text unit. If S10 has the value n, S11 and S12 will contain the record number of the record last written to the nth DESTINATION file. If, however, the record number for the next record of this file has already been defined (see S13), then S11 and S12 will contain this predefined record number. If the numerical value in S10 is smaller than zero or greater than the number of DESTINATION files specified when calling up the program, the program will terminate and display the corresponding error message.

S11 This variable contains that part of the record number which specifies the page number (see also S10 and S13).

For a description of what a record number is and how it is written, please refer to the "Files" chapter in "TUSTEP Basics".

S12 This variable contains that part of the record number which specifies the line and distinction number (see also S10 and S13).

For a description of what a record number is and how it is written, please refer to the "Files" chapter in "TUSTEP Basics".

S13 The contents of this variable can be used to define which record number (page-line-distinction number) is to be replaced by the contents of the variables S11 and S12 after executing the parameter RR. If S13 has the value 0, the record number of the present text unit will be replaced. If S13 has the value n, S11 and S12 will be used to define the record number of the next record to be written to the nth DESTINATION file. If the numerical value in S10 is smaller than zero or greater than the number of DESTINATION files specified when calling up the program, the program will terminate and display the appropriate error message.

S14 The contents of this variable show how characters are to be eliminated from the beginning of the working text. Please note that the contents of S14 may not be larger than that of S0 (length of working text). Characters will be eliminated at the completion of the compute statements in which these variables have been assigned a value. Afterwards, the value of variable S14 will be set to zero automatically.

- Selector switches

16 variables with the names WS1, WS2 to WS16 have been defined as selector switches which may contain the numerical values 0

and 1 only. A selector switch is cancelled if it contains the numerical value 0, it is set if it contains the numerical value 1. In a value assignment (see below) where a selector switch is specified to the left of the equals sign, the arithemtic expression (see below) to the right of the equals sign may not be an arbitrary one, but must be one of the numbers 0 and 1. A selector switch can be altered by means of a value assignment, as well as with the parameters WSL, WSS and WSU (see page 409). In addition, a selector switch can be used not only in arithmetic expressions, but also with the parameters WS+ and WS- (cf. page 403).

Functions

Functions are available as an aid in carrying out calculations. A function is called with its name and one or more arguments. Arguments are separated from each other by a comma and are entered in brackets after the function name. They may take the form of any arithmetic expression (see below). If a function with these arguments is called, a numerical value - the function value - is calculated according to the function's specifications.

The following functions are available:

(For simplicity's sake, numbers are used for the individual arguments in the following examples. In actual practice, integer variables or the names of macro variables are more commonly used instead of numbers.)

- Calculating the absolute value: IABS (arg)

  The function value obtained is the absolute value of arg.

  Example: The function value of IABS(-4) is 4.

- Calculating the minimum: MIN (arg1, arg2)

  The function value obtained is the smaller of the two numerical values arg1 and arg2.

  Example: The function value of MIN(-5,+3) is -5.

- Calculating the maximum: MAX (arg1, arg2)

  The function value obtained is the larger of the two numerical values arg1 and arg2.

  Example: the function value of MAX(-5,+3) is +3.

- Calculating the division remainder: MOD (arg1, arg2)

  The function value obtained is the remainder resulting from the division of arg1 by arg2.

  Example: the function value of MOD(234,10) ist 4.

- Interval function: IV (arg, arg1, arg2, arg3, ...)

  This function can be used to determine which interval the numerical value arg belongs to. A function value of zero is obtained if arg is less than arg1; the value 1 is obtained if arg is greater than or equal to arg1 but less than arg2; the value 2 if arg is greater than or equal to arg2 but less than arg3, etc. The number of arguments is not limited for this function. However, arg1 must be less than arg2, arg2 must be less than arg3 etc.

  Example: the function value of IV(16,1,10,100,1000) is 2.

- Calculating a check digit: IP (arg)

  A function value is obtained which is a number between 0 and 10 (inclusively). It is calculated according to the following rule: The last digit of the numerical value of arg is multiplied by 2, the next-to-last by 3, the third-to-last by 4, etc. The resulting products are then added. If this sum is a multiple of 11, then the function value is zero. Otherwise the difference between the sum and the next multiple of 11 serves as the function value. If the numerical value of arg is less than 1 or greater than 999999999 the function value is -1, for in this case the check digit is not defined.

- Date function: ID (day, month, year, number, mode)

  With this function it is possible to calculate and convert calendar dates. This function is controlled by the argument mode.

  To facilitate the calculation of calendar dates, the days are numbered in succession. Thus each day bears a unique number, the day number.

  mode=0: actual date
  Input: none
  Output: current date in the arguments day, month, year and current day number in the argument number

  mode=1: Calculating the day number by giving the date
  Input: Date in the arguments day, month, year
  Result: Day number in the argument number

  mode=2: Calculating the date by giving the day number
  Input: Day number in the argument number
  Result: Date in the arguments day, month, year

mode=3: Calculating the date of Easter
Input: Year number in the argument year
Result: Date of Easter in the arguments day, month, year and the corresponding day number in the argument number

mode=4: Calculating the interval between two calendar dates
Input: Date of the first calendar date in the arguments day, month, year and day number of the second (later) calendar date in the argument number
Result: The interval between the two dates in years, months and days (in the arguments year, month and day) as well as in days (in the argument number)

The function value so obtained - independent of the value of the argument mode - is the day of the week (1=Monday, 2=Tuesday, 3=Wednesday, 4=Thursday, 5=Friday, 6=Saturday, 7=Sunday) whose date is given in the arguments day, month and year. If there is an illegal input, the function value is zero (e.g. if mode has a value not defined, or if an improper date is given, such as the 29th of February in a non-leap year).

Calculations are based on the Gregorian calender. However, if a date occurs before October 15, 1582, the Julian calendar is used. If the Julian calendar is to be used for later dates as well, the respective negative value is to be given as the argument mode.

Example: Calculating the date of Pentecost/Whitsuntide for the year 1982

```
H0 = ID (H1, H2, 1982, H4, 3);
H0 = ID (H1, H2, H3, H4+49, 2);
```

First, the date of Easter is calculated. It does not matter which numerical values H1, H2 and H4 contain before the date function is called up. Because 3 has been given as the mode, only the argument year is evaluated. The day of the week, day and month of Easter are stored in H0, H1 and H2, but are not used for further processing. The day number of Easter is stored in H4. Because the interval between Easter and Pentecost is exactly 7 weeks (= 49 days), the day number of Pentecost is obtained by adding 49 to the day number of Easter. This day number is then converted into the respective calendar date. It does not matter which numerical values H1, H2 and H3 contain before the date function is called up. Because 2 has been given for the mode, only the argument number is evaluated. The day of the week (7 for Sunday) is stored in H0, day, month and year of the date of Pentecost are stored in H1, H2 and H3.

Arithmetic expressions

An arithmetic expression is a rule for computing by which a numerical value is defined. It consists of operands, arithmetic operators and pairs of brackets.

An operand may be an (integer) number, a (simple or indexed) variable or a function call. In the simplest case, an arithmetic expression consists of only one of these three operands.

The arithmetic operators for the four basic mathematical operations are:

+ Addition	* Multiplication
- Subtraction	/ Division

When the arithmetic expression is evaluated, multiplication and division are executed prior to addition and subtraction. If there are several consecutive multiplication and division operations, they are carried out from left to right. The same holds true for consequtive addition and subtraction operations. Deviations from this rule can be made by placing brackets at the appropriate positions.

Please note that division is carried out with integers only: remainders will be lost and there is no rounding. For example, 3/2 results in the value 1 (not 1.5); 3/2*4 will give the value 4 (not 6).

Relation conditions

A relation condition is where two numerical values are compared. It consists of two arithmetic expressions which are connected by a relation operator:

arithm. expression relation operator arithm. expression

There are six relation operators:

.EQ. equal	.NE. not equal
.GT. greater than	.LT. less than
.GE. greater or equal	.LE. less or equal

A relation condition is either satisfied, resulting in the logical value TRUE, or it is not satisfied, resulting in the logical value FALSE.

Examples: I1.EQ.0	is satisfied if I1 contains the numerical value 0, otherwise it is not satisfied.
MOD(I1,10).LT.5	is satisfied if the division of I1 by 10 results in a remainder which is less than 5, otherwise it is not satisfied.
I1+I2.EQ.4*I3	is satisfied if the addition of I1 and I2 results in a value 4 times

larger than I3, otherwise it is not satisfied.

Logical expressions

A logical expression consists of relation conditions which may be connected by logical operators. In its simplest case, a logical expression may also have only one relation condition.

The logical operators are:

.AND. logical AND
.OR. logical OR

A logical expression is evaluated in a manner analogous to an arithmetic expression. The result is either the logical value TRUE or the logical value FALSE. When the logical expression is evaluated, the logical AND is executed prior to the logical OR. This order may be changed by the appropriate use of brackets.

Examples: I1.EQ.I2	has the logical value TRUE if I1 contains the same numerical value as I2, otherwise it has the logical value FALSE.
I1.GE.1 .AND. I1.LE.8	has the logical value TRUE if I1 contains a numerical value from 1 and 8 (inclusively), otherwise it has the logical value FALSE.
I1.EQ.0 .AND. (I2.EQ.1 .OR. I2.EQ.2)	has the logical value TRUE if I1 contains the numerical value zero and I2 contains the numerical values 1 or 2, otherwise it has the logical value FALSE.

Value assignments

A value assignment has the form:

variable = arithmetic expression

A value assignment causes the arithmetic expression to be evaluated. The result is stored in the integer variable (simple or indexed) placed to the left of the equals sign. In this statement, the equals sign has the function of an assignment operator and thus differs from its usual mathematical function.

Examples: H1 = 0; I1 = I1 + 1; I(I1) = MAX (H2,H3);
I1 = I1 * (H1 + H2); I0 = MOD (S12, 1000);

Jump statements

A jump statement has the form:

GO TO arithmetic expression

The use of a jump statement causes the arithmetic expression to be evaluated. The result is interpreted as the jump destination and the corresponding position in the program is jumped to. The value of the ones position in the result designates the program part; the hundreds and tens positions show the number of the pass.

Examples: GO TO 0; GO TO H1 * 10 + 2; GO TO B(I1)

In addition, there are two special jump statements:
The statement

GO ON

results in a jump which skips all following compute statements of the same pass.
The statement

EXIT

may be located only between the statements LOOP and END LOOP (cf. loop statements) and results in a jump which skips all following statements until the next END LOOP.

Conditional statements

A conditional statement lets the user specify that one or more successive instructions are to be executed only under certain conditions. These conditions are specified by a logical expression. There are three forms of conditional statements:

1.) If a single instruction (in this case, this may be only a value assignment or a jump statement) is to be executed only under the conditions specified in the logical expression, the following form may be chosen:

IF (logical expression) instruction

The instruction specified after the brackets is executed only if the logical expression in brackets has the value TRUE. Otherwise the instruction is skipped.

```
Examples: IF (I1.EQ.0) I2 = I2 + 1;
          IF (MOD(S12,1000).EQ.0) GO TO 3;
```

2.) If the following form of the conditional statement is chosen, one or more instructions of any type may be executed on the basis of the conditions specified in the logical expression.

```
IF (logical expression) THEN;
   instructions;
END IF
```

The instructions located between THEN and END IF are executed only if the logical expression in brackets has the value TRUE. Otherwise, these instructions are skipped.

Examples:
```
IF (I1.EQ.60) THEN; S11=S11+1; S12=1000; END IF;
IF (WS1.NE.0) THEN; I2=I2+1; GO TO 0; END IF;
```

3.) The third form of the conditional statement is an extension of the second form. With it, two sequences of instructions may be given, of which one is executed and the other one is skipped.

```
IF (logical expression) THEN;
   instructions;
ELSE;
   instructions;
END IF
```

If the logical expression in brackets has the value TRUE, the instructions located between THEN and ELSE are executed and the instructions between ELSE and END IF are skipped. If the logical expression has the value FALSE, the instructions between THEN and ELSE are skipped and the instructions between ELSE and END IF are executed.

Loop statements

Loop statements make it possible to run a series of instructions as often as required. This series of instructions is initiated by the statement

```
LOOP
```

and terminated by the statement

```
END LOOP.
```

There are three ways to exit from this series of instructions:
- The statement EXIT will cause a jump to the following END LOOP.
- The statement GO TO causes a jump to a specified program part.
- The statement GO ON can be used to terminate the execution of compute statements, i.e. all subsequent statements of this pass (including those located after a END LOOP statement) are skipped.

Example: The variables I(1) to I(32) are to be set to zero. Here the variable H0 is used as the counter and as the index.

```
H0 = 1;
LOOP;
```

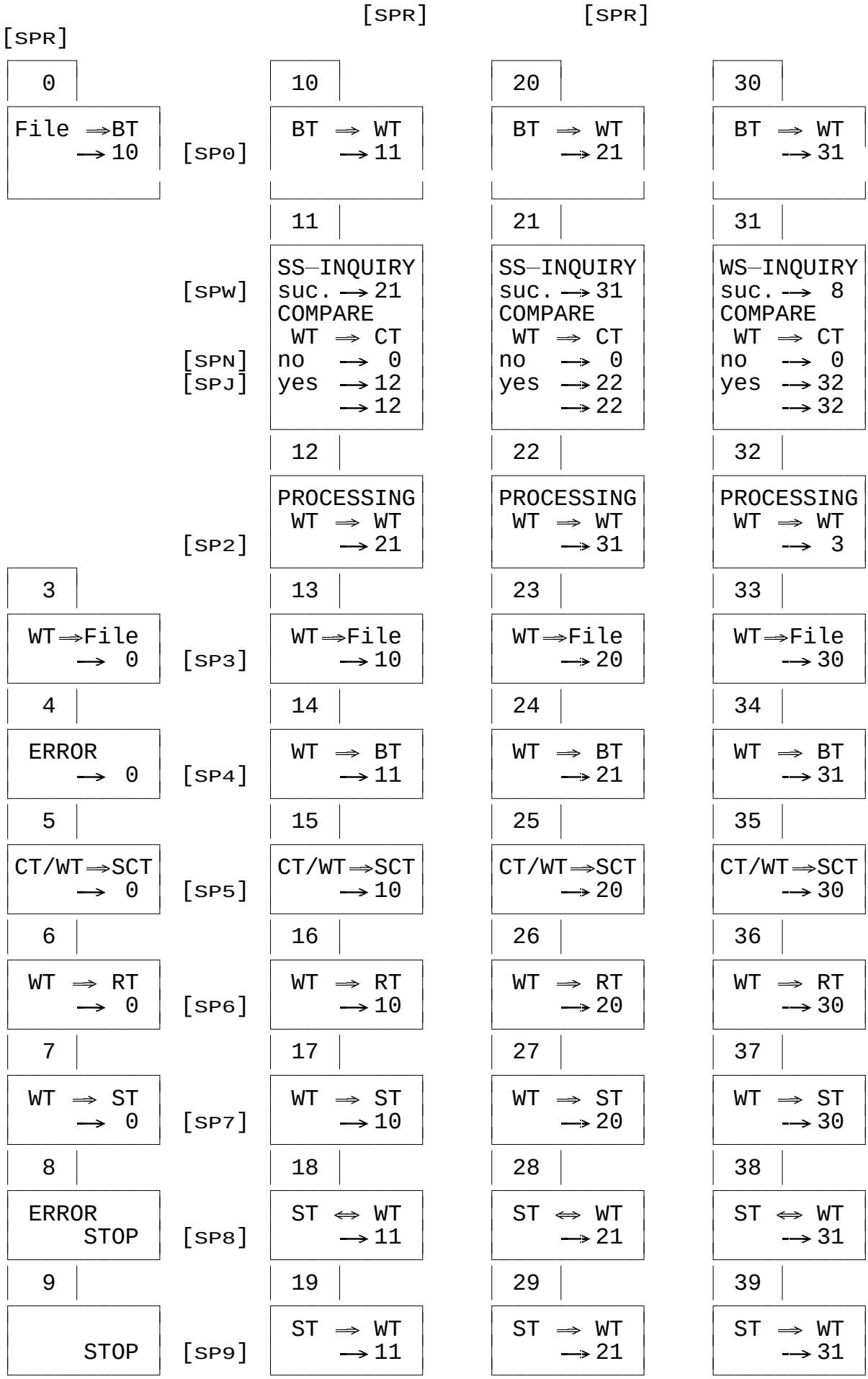
```
      I(H0) = 0;
      IF (H0.EQ.32) EXIT;
      H0 = H0 + 1;
   END LOOP;
```

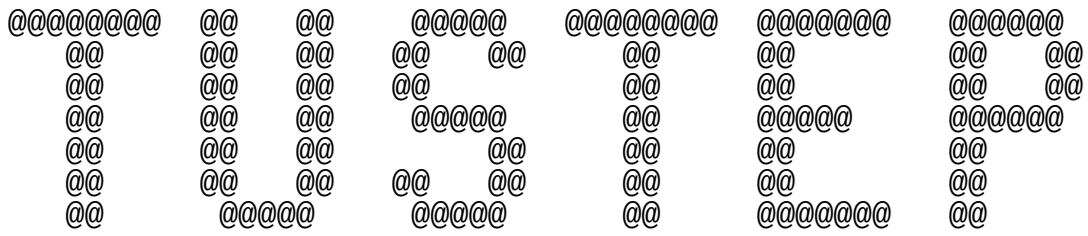
Logical program structure

The following page shows the logical program structure as used for three passes. The default values for parameters SPR (or SP0, SPW, SPN etc.) have also been specified for three passes. Each time a pass is carried out, the default values for the following pass are increased by 10. An exception is parameter SPN, whose default value remains 0 for all passes, and the two parameters SPW and SP2. SPW and SP2 follow the same rules for default values as outlined above for all passes except for the final one: here the default value for parameter SPW is 8; for SP2 it is 3.

Key to chart abbreviations:

BT	basic text
CT	comparison text
RT	replacement text
SCT	stored comparison text
SPJ	jump table for jump when conditions are met (yes)
SPN	jump table for jump when not all conditions are met (no)
SPn	jump table for jump after program part n
SPW	jump table based on selector switch settings
SS	selector switch
ST	stored text
suc.	successful
WT	working text


[SPR]
[SPR]
[SPR]
0
File ⇒BT
⟶ 10
[SP0]
10
BT ⇒ WT
⟶ 11
20
BT ⇒ WT
⟶ 21
30
BT ⇒ WT
⟶ 31
11
21
31
[SPW]
[SPN]
[SPJ]
SS—INQUIRY
suc. ⟶ 21
COMPARE
WT ⇒ CT
no ⟶ 0
yes ⟶ 12
⟶ 12
SS—INQUIRY
suc. ⟶ 31
COMPARE
WT ⇒ CT
no ⟶ 0
yes ⟶ 22
⟶ 22
WS—INQUIRY
suc. ⟶ 8
COMPARE
WT ⇒ CT
no ⟶ 0
yes ⟶ 32
⟶ 32
12
22
32
[SP2]
PROCESSING
WT ⇒ WT
⟶ 21
PROCESSING
WT ⇒ WT
⟶ 31
PROCESSING
WT ⇒ WT
⟶ 3
3
13
23
33
WT⇒File
⟶ 0
[SP3]
WT⇒File
⟶ 10
WT⇒File
⟶ 20
WT⇒File
⟶ 30
4
14
24
34
ERROR
⟶ 0
[SP4]
WT ⇒ BT
⟶ 11
WT ⇒ BT
⟶ 21
WT ⇒ BT
⟶ 31
5
15
25
35
CT/WT⇒SCT
⟶ 0
[SP5]
CT/WT⇒SCT
⟶ 10
CT/WT⇒SCT
⟶ 20
CT/WT⇒SCT
⟶ 30
6
16
26
36
WT ⇒ RT
⟶ 0
[SP6]
WT ⇒ RT
⟶ 10
WT ⇒ RT
⟶ 20
WT ⇒ RT
⟶ 30
7
17
27
37
WT ⇒ ST
⟶ 0
[SP7]
WT ⇒ ST
⟶ 10
WT ⇒ ST
⟶ 20
WT ⇒ ST
⟶ 30
8
18
28
38
ERROR
STOP
[SP8]
ST ⇔ WT
⟶ 11
ST ⇔ WT
⟶ 21
ST ⇔ WT
⟶ 31
9
19
29
39
STOP
[SP9]
ST ⇒ WT
⟶ 11
ST ⇒ WT
⟶ 21
ST ⇒ WT
⟶ 31



Tübingen

System of Text Processing Programs

Program

C O R R E C T

1993

TÜBINGEN UNIVERSITY - CENTER FOR DATA PROCESSING

Department of Literary and Documentary Data Processing

Brunnenstrasse 27, D-72074 Tübingen

Survey:

Command:

#CORRECT

Specifications:

SOURCE= file		Name of the file containing the data to be corrected
= -STD-		The standard TEXT file contains the data to be corrected
DESTINATION = file		Name of the file to which the corrected data are to be written
= -STD-		The corrected data are to be written to the standard TEXT file
MODE = -STD-	*	Retain numbering of records if possible
= +		Renumber records (in text mode)
ERASE = -	*	Do not erase data in the DESTINATON file and LISTING file
= +		Erase data in the DESTINATION file and LISTING file
PARAMETER = file		Name of the file containing parameters
= *		Parameters follow the command and are ended by *EOF
CORRECTION = file		Name of the file containing the correcting instructions
LISTING = -	*	No protocol to be recorded; only error messages will be written into the journal
= -STD-		The correction protocol is to be written into the standard LISTING file
= file		Name of the file to which the correction protocol is to be written

Features:

This program is used to correct texts without using the Editor. The correcting instructions, which have been recorded in a file, allow the user to replace, erase or insert

- lines (eg. line 2 on page 3)

- words (eg. the second word in line 3 on page 4)

- characters (eg. the second character in line 3 on page 4
or the second character in the third word in line 4 on page 5).

A protocol of the executed corrections is provided upon request.

Note

The correcting instructions for this program are usually created with the command #COMPARE and then processed further.

Correcting instructions

A correcting instruction consists of the following components:

- Area of text to be corrected
- Type of correction to be carried out. This is indicated by one of the correction codes "-" ("delete"), "+" ("insert"), "=" ("replace"), or "*" ("comment").
- Corrective text to be used in corrections (for "insert" and "replace").

Corrective text should be entered immediately after the correction code (i.e. not separated by any blank spaces).

Correcting instructions must be sorted in ascending order, according to the text areas that are being corrected. Corrections which are added later must be placed at the proper position.

Each correcting instruction should start at the beginning of a new line. If the correction text is too long for one input line, it may be continued in the following line. Such continuation lines are to be marked with a "+" at the beginning of the line. Hyphenation should not be used in correcting instructions which take up more than one line.

When correcting with "insert", you are expected to enter a position (line, word or character) and not a text area. The corrective text will be inserted after this position. This position entry takes the same form as that part of the area specification which precedes the "to" dash.

Specifying the text areas to be corrected

Areas of the text to be corrected are specified by entering page and line numbers. More limiting specifications are made by including word and/or character numbers. An area entry uses the following syntax:

p1.l1[/d1][,w1][:c1]-p2.l2[/d2][,w2][:c2]

where:

p1 and p2 = page number
l1 and l2 = line number
d1 and d2 = distinction number
w1 and w2 = word number
c1 and c2 = character number

Elements in square brackets are optional. Both parts of an specification must limit the text area in the same manner (i.e. by either "line" or "word" or "line:character" or "word:character"). Otherwise, you only have to specify those elements that are sufficient in defining a unique text area.

- If a distinction number is not entered after a line number, a distinction number of 0 is assumed.
- If you are correcting one or more whole lines, word and character numbers should be omitted.
- If the corrections are limited to whole words, the character number should be omitted.
- If the area to be corrected is defined in terms of individual characters, each of which are counted from either the end or beginning of a line, the word number should be omitted. Characters may also be counted from the beginning or end of a word, only in this case all components should be included in the area specification.

Short cuts for specifying a text area

The page number can be omitted if it is the same as the most recently named page number.

The page number can also be written on a separate line and preceded by a "=". For example:

```
=1256
2-3=replacement for lines 2 and 3 on page 1256
5,2+insertion after the second word in line 5 on
page 1256
```

You may leave out all elements, including separators (. / , :), in the second half of the area definition which, starting from the left, match those in the

first half (Exception: a "/" located in front of a distinction number must always remain). For example:

p1.l1/d1,w1:c1[-[[[p2.]l2[/d2],]w2:]c2]

Especially when using the functions "delete" and "replace", the first half of the area specification (ie. that part located before the "to" dash) alone is sufficient for deleting or replacing a single line, word or character.

Examples of equivalent expressions:

1.2-1.2	and	1.2
1.2-1.3	and	1.2-3
1.2-1.2/3	and	1.2-/3
1.2,3-1.2,3	and	1.2,3
1.2,3-1.2,4	and	1.2,3-4
1.2:3-1.2:4	and	1.2:3-4
1.2,3:4-1.2,4:6	and	1.2,3:4-4:6

When specifying "word" and "character", it is also possible to count words or characters starting from the right. This is done by choosing an exponent of ten (10, 100, 1000, 10000, 100000) that is greater than the sum of words (or letters) in a line (or greater than the number of letters in a word) plus the position of the word or character as counted from the right. A word or character is thus specified by counting how far it is from the right margin and then subtracting this number from the chosen exponent.
Example: 100-2=98 for the next-to-last word or character of a line containing less than 98 words or characters; writing 998, 9998 or 99998 has the same effect.

Notes:

When counting words, each character string surrounded by the beginning and/or the end of a line and/or by one or more blank spaces is a considered one word. Thus, punctuation marks and special characters are either part of a word or, when separated from it by blank spaces, a separate word. A hard space ("_") does not count as a blank.

When counting characters in a line, blanks at the beginning of the line must also be counted.

If one or more words or characters are to be inserted at the beginning of a line, the beginning of the line can be addressed by word no. 0 or character no. 0.

If the text area is defined using word numbers (without an additional character number), blank spaces

after a word are considered part of the word. If this type of definition is used when correcting with the functions "insert" and "replace", a blank space is appended to the end of the corrective text. If the the corrective text is to be placed at the end of a line (using the operation "insert after last word") then a blank is inserted before the corrective text at the end of the line.

If for "replace" and "insert" instructions the correction area is defined using character numbers, and no other characters follows the correction characters "=" or "+", then a blank will be inserted there when the correction is executed.

Blanks located at the end of a line are removed during text input, input of correcting instructions and text output.

Correction protocol

The LISTING file shows all changes that have been carried out by the correction process. Each line that was modified is listed in its original and corrected forms. The changes that have been carried out are marked between these two lines in the following manner:

- a deletion is marked with a "-" under the character deleted in the original line (ie. upper line in the LISTING file).
- an insertion is marked with a "+" placed over the inserted character in the corrected line (ie. lower line in the LISTING file).
- a replacement is marked as a combination of deletion and insertion.

When "-" and "+" coincide (for example, when a replacement is to be marked), an "*" will appear at this position.

Parameters

Values in [ ] refer to the type of parameter employed. The various types of parameters are described in the "Parameters" chapter of "TUSTEP Basics".

Values in < > refer to default settings.

Selecting data

If the entire file is to be processed, none of the following parameters are necessary.

BER Definition of a single area ("page.line-page.line") or a starting point ("page.line"). This parameter is only used when not processing the entire input file. [xI]

Parameters for LISTING file settings

DR Specifications for printer output control [I]

Four numerical values can be specified:

1st number: Number of columns <1>

Number of columns (printed side by side) on each page

2nd number: Left margin <0>

Number of blanks to the left of the first column

3rd number: Column width <132>

Number of characters per column

4th number: Space between columns <0>

Number of blank spaces between columns

DRZ Additional specification for printer output control [I]

Three numerical values can be specified:

1st number: Header text <3>

Number of lines for the header (including blank lines)

2nd number: Column height <60>

Number of lines per column (excluding lines for the header and footer)

3rd number: Footer text <0>

Number of lines for the footer (including blank lines)

KT Text parts to be printed at the top of every page as a header [II]
<"file name" xx. xxx. xxxx xx.xx xxxxxx>

To insert the current date, enter "xx. xxx. xxxx" or "xx.xx.xx" at the appropriate position. Positions for the current time may be indicated by "xx.xx" and for the page number by "xxxxxx" (2 to 6 "x"s, but at least as many positions as necessary for the page number). If "- xxxxxx -" is entered for the page number, it will appeared centered on the page between two minus signs, with each minus sign separated from the page number by a space having the width of up to one whole blank. However, the date, time and page number can be inserted only one time each.

If a text part begins with a "*:", the rest of the text part serves as a header for every text column. If a numeral n is entered in place of the asterisk, the rest of the text part is used as a header for the nth column. If the numeral given equals 0, the rest of the text part is used for the entire line. If a text part does not begin as just described, "0:" is assumed (standard value).

The following rules determine which line of the header is taken up by the specified text parts: A text part which is designated for an entire line will be printed at the start of a new line (starting with the first line). A text part which is to appear above a particular column should be entered in the same line as the preceding text part, unless this line contains text meant for an entire line, for the same column, or for a column further to the right. In this case, the text part will be printed in the next line.

Each of the specified text parts can be arranged in three parts using the formating

instructions "@z" and "@/":
left-aligned @z centered @/ right-aligned
The individual parts will be inserted left-aligned, centered and right-aligned. An individual part may be omitted; in this case the formatting instructions in front of the second and third parts may also be omitted.

FT Text parts (analogous to that described in parameter KT) to be printed as a footer at the bottom of every page. [ıı]

The date, time or page number may not be entered in the footer if already specified to appear in the header.

DRT Printing device for which the data are to be prepared. [xı]

This parameter is obligatory if a listing file is to be created.

The types of available printers depends on the actual computer being used. To obtain a list of these, use the command.
#LIST,PRINTERS

Output record length and numbering

NR Specifications for numbering output records [ı]

Three numerical values can be specified:

1st number: Page number with which output is to start (999999 for selecting the next available page).
<MODE=-STD-: 999999; MODE=+: 1>

2nd number: Maximum number of records per page.
<MODE=-STD-: 1000000; MODE=+: 60>

3rd number: Increment for numbering when a new record number must be allocated.
<MODE=-STD-: 10; MODE=+: 1000>

SL Length of output records [ı]

Two numerical values can be specified:

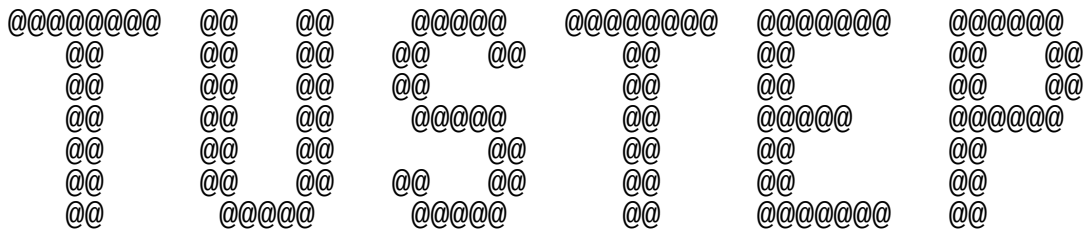
1st number: Maximum length of lines to be subdivided because they are longer than the second number specified in this parameter. <99999>

2nd number: Maximum length of lines which are not to be subdivided. <100 (or the value specified in the first number if this is greater than 100)>

Before output, a line that contains more characters than specified by this parameter's second number will be subdivided into two or more output records. The line will be subdivided into records having a maximum number of characters as specified by this parameter's first number. The line is divided at blanks which are not preceded by a "-" (exception: a blank preceded by " -", since this "-" represents a dash and can therefore not be confused with a hyphen). If such a division point cannot be found within the number of characters specified by the first number, the next possible division point will be selected.

Alphabetical list of parameters

* * * * *



Tübingen

System of Text Processing Programs

Program

F O R M A T

1993

TÜBINGEN UNIVERSITY - CENTER FOR DATA PROCESSING

Department of Literary and Documentary Data Processing

Brunnenstrasse 27, D-72074 Tübingen

Survey:

Command:

#FORMAT

Specifications:

SOURCE	= file		Name of the file containing the data (with formatting instructions) to be formatted
	= -STD-		The standard TEXT file contains the data (with formatting instructions) to be formatted
DESTINATION	= -	*	Output only to the LISTING file
	= file		Name of the file to which the data (with formatting instructions) are to be written observing the new page-line division.
	= -STD-		The formatted data (with formatting instructions) are to be written into the standard TEXT file observing the new page-line division.
MODE	=		Type of printer for which the data are to be prepared. The types of printers available depends on the actual computer being used. To obtain a list of these, use the command #LIST, PRINTERS The printer can also be specified with parameters.
ERASE	= -	*	If the DESTINATION file or the LISTING file already contains data, they are to be retained.
	= +		If the DESTINATION file or the LISTING file already contains data, they are to be erased beforehand.
PARAMETERS	= -	*	No parameters
	= file		Name of the file containing parameters
	= *		The parameters follow the command and are ended by *EOF.
PREFIX	= -	*	No user-defined default settings
	= file		Name of the file containing formating instructions for user-defined default settings.
	= *		The formatting instructions for user-defined default settings follow the command (and any parameters) and are ended by *EOF.

LISTING	= -STD- *	The formatted data are to be written to the standard LISTING file.
	= file	Name of the file to which the formatted data are to be written.
	= -	No listing output of formatted data

Features:

With this command texts can be prepared for printing. The text is automatically made up into lines and pages (including hyphenation, line justification and footnotes). The layout can be controlled by instructions which are included in the text.

Notes:

Data in the SOURCE file always remain unaltered. The formatted results of the program (i.e. the data prepared for printing) are written to the LISTING file. These can be then printed (sent to a printer) with the command #PRINT.

Data are written to the DESTINATION file as they are found in the SOURCE file (with the exception of character strings which have been replaced according to the appropriate parameters). The line division and the page-line numbers, however, are adjusted to the formatted result.

The DESTINATION file is therefore ideal for making further corrections. It has the advantage over the SOURCE file in that its page-line numbering corresponds to that of the printed results, making it much easier to find the exact position in the Editor where the text must be corrected.

If an index is to be compiled from the formatted text, the DESTINATION file must be used so that the references correctly reflect the updated page positions of the formatted text.

Parameters

Values in [ ] refer to the type of parameter employed; the various types of parameters are described in the "Parameters" chapter of "TUSTEP Basics".

Values in < > refer to default settings.

In addition to the following parameters, other parameters can be used to define character groups and character strings. [v]

Selecting data

If the entire file is to be processed, none of the following parameters are necessary.

BER Definition of a single area ("page.line-page.line") or a starting point ("page.line"). This parameter is only used when not processing the entire input file. [xI]

If a segment of a segment file is to be processed, the name of the segment can be substituted for the area.

This parameter can only be used when the record numbers of the file are in ascending order.

MAX For test runs, this specifies the maximum number of records that are to be read/outputted, or the maximum number of pages that are to be prepared for printing. [I]

Three numerical values can be specified:

1st number: the maximum number of records that are to be read <999999>
2nd number: the maximum number of records for output <999999>
3rd number: the maximum number of pages for output <999999>

Parameters for LISTING file settings

DR Printer output control [I]

Four numerical values can be specified:

1st number: Columns <1>

Number of columns appearing side-by-side on every page <1>

2nd number: Left margin <10>

Number of blanks to the left of the first column

3rd number: Column width <64>

Number of characters per column

4th number: Space between columns <0>

Number of blank spaces between columns

DRZ Additional specification for printer output control [I]

Seven numerical values can be specified:

1st number: Header <3>

Number of lines for the header (including blank lines)

2nd number: Height of column <60>

Number of lines per column (excluding lines for the header and footer)

3rd number: Footer <0>

Number of lines for the footer (including blank lines)

4th number: Columns <1>

Number of columns per page

5th number: Repetition of footer text

Number of lines of the page's footer text (counting from the first line down) which are to be repeated after every column (except for the last column).

6th number: blank lines <0>

Number of blank lines between individual columns

7th number: Repetition of header text <0>

Number of lines of the page's header text (counting from the bottom line to the top) which are to be repeated before every column (except for the first column).

KT Text parts to be printed at the top of every page as a header [II]
<"file name" xx. xxx. xxxx xx.xx xxxxxx>

To insert the current date, enter "xx. xxx. xxxx" or "xx.xx.xx" at the appropriate position. Positions for the current time may be indicated by "xx.xx" and for the page number by "xxxxxx" (2 to 6 "x"s, but at least as many positions as necessary for the page number). If "- xxxxxx -" is entered for the page number, it will appeared centered on the page between two minus signs, with each minus sign separated from the page number by a space having the width of up to one whole blank. However, the date, time and page number can be inserted only one time each.

If a character string begins with a "*:", the rest of the character string serves as a header for every text column. If a numeral n is entered in place of the asterisk, the rest of the character string is used as a header for the nth column. If the numeral given equals 0, the rest of the character string is used for the entire line. If a character string does not begin as just described, "0:" is assumed (standard value).

The following rules determine which line of the header is used by the character string: A character string which is designated for an entire line will be printed at the start of a new line (starting with the first line). A character string which is designated for a particular column will be printed in the same line as the preceding character string, unless this line contains text meant for an entire line, for the same column, or for a column further to the right. In this case, the character string will be printed in the next line.

Each of the specified character strings can be arranged in three parts by using the formatting instructions "@z" and "@/":

left-aligned @z centered @/ right-aligned

The individual parts will be inserted left-aligned, centered and right-aligned. An individual part may be omitted; in this case the formatting instructions in front of the second and third parts may also be omitted.

KTZ Specifies whether the header text is to be printed on the first page. [I] <0>

0 = Suppress header text
1 = Print header text

FT Text parts (analogous to the description for parameter KT) to be printed as a footer at the bottom of every page. [II]

Date, time and page number may not be included in the footer if already specified to appear in the header.

DRT Printing device for which the data are to be prepared. [XI]

This parameter is obligatory if no printer has been given in the specification MODE (and may only be specified here if this is the case).

The types of available printers depends on the actual computer being used. To obtain a list of these, use the command
#LIST,PRINTERS.

Parameters for numbering

NR Initial page number for output [I] <1>

ROM Specifies whether Arabic numerals or Roman numbers are to be used for the page number in the header or footer text. [I] <0>

0 = Arabic numerals
1 = Roman numbers in lowercase letters
2 = Roman numbers in uppercase letters

FNN Number to be used as the first footnote number. [I] <1>

Replacing character strings

X Pairs of character strings (and exception strings). The first character string of a pair will be replaced by the pair's second character string upon input of the SOURCE file. [x]

Replacement is carried out in the input record. Before the character string is replaced, a blank is added to both the beginning and end of the record and is removed as soon as replacement has been carried out. This can be used, for example, to convert a sequence of abbreviated instructions into its full form.

If hyphenated words are to be rejoined, the program checks whether the record ends with a hyphen after replacement has been carried out.

XPR Pairs of character strings (and exception strings). The first character string of a pair will be replaced by the pair's second character string upon output to the LISTING file. [x]

Replacement is carried out word by word (blanks can therefore not be replaced).

XXX Pairs of character strings (and exception strings). The first character string of a pair will be replaced by the pair's second character string upon output to the DESTINATION file. [x]

Replacement is carried out in the output record for the DESTINATION file. A sequence of instructions, for example, can thus be converted back to its original shortened form.

Margin justification

REZ Specifies whether the margin of a single line (i.e. the first line of a paragraph containing only two lines) is to be justified. [I] <1>

0 = Justify margin
1 = Do not justify margin

Inverting text parts

Normally, none of the following parameters need to be specified, since the default settings are sufficient for standard use.

Arabic, Hebrew and Syrian texts must be printed from right to left. However, the #PRINT command can only print from left to right. For this reason, such text parts have to be inverted. At the same time, care must be taken that numbers, control characters and codes which consist of more than one character are not inverted along with the text.

AUM Character strings marking the beginning of the text parts to be inverted. [IX]

Default settings:

```
>1Z       AHY
AUM       '#>1+'
```

EUM Character strings marking the end of the text parts to be inverted. [IX]

Default settings:

```
>1Z       CGKPR
>2Z       AHY?CGKPR
EUM       '#>1+'#>2-'
```

NUM Character strings which are not to be inverted should they occur in the text parts to be inverted. [IX]

Default settings:

```
>1Z       =+-
>2Z       ',;!."
>3Z       +-:
>4Z       ()[]{}<<>>
>5Z       !"'()*,-./:;<<>>?[\]{}
>5S       '%>5'%>5>5'
>6S       '</'#.</'_'
NUM       '<>>/'#>1>/>/>/'#>2<%'
NUM       '#(<></)'#>%>3'>4'<>>5>6
```

XUM Pairs of character strings (and exception strings). The first character string of a pair will be replaced by the pair's second character string within the character strings which are not to be inverted (as specified with the parameter NUM). [X]

Default settings:

```
>1Z       0123456789/SF
>5Z       !"'()*,-./:;<<>>?[\]{}
>5S       '%>5'%>5>5'
XUM       '#>1+'#>=02-'#>1-'#>=02+'
```

```
XUM       '#.:'#.;'#.;'#.:'
XUM       '(')')'('[']']'['{'}'}'{'<<'>>'>>'<<'
XUM       ''>5'#"<<'#">>'#(<></)''
```

Alphabetical list of parameters

Character set

A complete list of character sets (with input codes) can be found in "TUSTEP Basics", as well as the various possibilities of displaying characters in superscript and subscript, and the available display types (underlining, letter-spacing, bold, etc.).

If the characters "#", "$", "%", "&", "@", "\", "_" are to be printed, they must be written as "^#", "^$", "^%", "^&", "^@", "^\", "^_". Information concerning the coding of other special characters is described in "TUSTEP Basics".

Instructions:

An instruction also serves as a separator character between two words. Therefore, it does not matter whether an instruction is surrounded by blanks or not. If, however, one of the instructions 2 to 40 is immediately followed by a word starting with one or more digits, at least one blank must be located between the instruction and the word; otherwise the digit(s) would be interpreted as being part of the instruction.

Letters in the instructions may be written in either lowercase or uppercase. The specification n stands for an integer with no sign. It is ended by an arbitrary character (except a digit) placed to its right. The specification "n = :" in the description refers to the case when n is missing, i.e. when no number has been given.

The specification or definition of instructions marked with a "+" or "-" remain in force until they are canceled or altered by a corresponding instruction. The specification or definition of instructions which are marked by a "+" are in addition stored for the "format" currently valid and will be reset when another format is selected. All instructions which are unmarked affect only their immediate surroundings.

0 Blank = Space between words. More than one blank will be treated as 1 blank unless specified otherwise in instruction 55 (@-).

1 Blank line: n successive blank lines in the input data correspond to instruction 2 ($n, n = number of blank lines).

$n 2 Start of paragraph (cp. "Paragraphs" on page 479): new line preceded by n blank lines. These blank lines will be suppressed at the start of a new page or column. A succession of n blank lines in the input data correspond to the instruction $n. If more than one instruction for blank lines is given (with no intervening text), only the instruction requesting the largest number of blank lines will be carried out.

n = 0: 1/2 blank line
n = : no blank line

Line spacing for blank lines is the same as that used for lines to be printed: the default setting is 1/6 inch (meaning 6 lines per inch), unless specified otherwise in instruction 5 (&Vn).

$$n 3 (not yet defined)

$$$n 4 Start of paragraph (cp. "Paragraphs" on page 479): new line; new column (i.e. new page for single-column layout) if less than n lines are available on the current page. Any blank lines which are to be placed before the next line to be printed are not counted. If

this instruction is given more than once in succession (with no intervening text), only the instruction with the largest value of n will be carried out. However, the instruction for page break ($$$0) has precedence.

n = 1: as in $$$ if the following text up to the next line break instruction fits into 1 line; otherwise as $$$2.
n = 0: New page
n = : New line only, no start of paragraph

The amount of space taken up by the n lines which must still be available on the page depends on the line spacing setting. The default setting is 1/6 inch per line (meaning 6 lines per inch), unless specified otherwise with instruction 5 (&Vn).

&Vn 5 + Set line spacing to n (max. 4) lines (default setting n=1).
n = 0: 1 1/2 line spacing (4 lines per inch)
n = 1: single spacing (6 lines per inch)
n = 2: double spacing (3 lines per inch)
n = 3: triple spacing (2 lines per inch)
n = 4: quadruple spacing (1 1/2 lines per inch)
n = : (not yet defined)

&$n 6 + Indent paragraph continuation lines n characters from the left margin at the start of a paragraph coded by instruction 2 (default setting n=0).

n = : Indent as at the start of a paragraph

&Fn 7 + Indent the following n characters from the left margin (Instructions 8 and 9 have precedence; default setting: n=0).

n = : Indent as usual for continuation lines

&:n 8 Indent the rest of the line and all following lines (until the next line break instruction) n characters from the left margin. (Has precedence over instruction 7).

n = : (not yet defined)

&;n 9 Indent all following lines (until the next line break instruction) n characters from the left margin (has precedence over instruction 7).

n = : Indent following lines to the current position (relative to the left margin)

&Ln 10 + Define a left margin of n characters (default setting: n=0).

n = : Define current position as left margin

&Rn 11 + Define a right margin of n characters (default setting n=0).

n = : Define current position as right margin

&An 12 as instruction 10, in addition: move to left margin
n = : Move only to left margin

&En 13 as instruction 11, in addition: move to right margin.
n = : Move only to right margin

&=n 14 Move to position n

n = 0: Mark current position
n = : Move to the position marked by instruction &=0

&+n 15 Move n positions to the right
n = 0: Mark position (relative to left margin)
n = : Move to position marked by instruction &+0

&-n 16 Move n positions to the left
n = 0: Mark position (relative to right margin)
n = : Move to position marked by instruction &-0

&Tn 17 Move to tabulator n (max. 20). If this has not been defined (with instruction 18), the current position will not change.

n = 0: Move to the next physical tabulator stop (i.e. the tabulator stop located at the current position or farther to the right and whose number follows that of the last tabulator addressed in the same line, or which has the smallest number in case a tabulator is addressed for the first time in the line). If such a tabulator does not exist, move to the right margin. If all tab stops are in ascending order, this instruction has a function similar to that of the tabulator key on a typewriter.
n = : Move to the next logical tabulator stop (i.e. the tab stop whose number is 1 higher than that of the last tab stop addressed in the same line, or the tab with the number 1 in case a tab stop is being addressed for the first time in the line). If this has not yet been defined (with instruction 18), the current position will not change.

&Mn 18 + Set tabulator n (max. 20) at the current position

n = 0: Cancel all tabulators (default setting)
n = : Set next tabulator (i.e. the tabulator with the number 1, if a tabulator is being addressed for the first time in the line) at the current position

&Zn 19 + Center field in front of tabulator n (max. 20). (Takes effect when positioning to tabulator n.)

n = 0: No field to be centered (default setting)
n = : Center all fields

&/n 20 + Field in front of tabulator n (max. 20) to be printed right-aligned (Takes effect when positioning to tabulator n.)

n = 0: No fields to be printed right-aligned (inital setting)
n = : All fields to be printed right-aligned

&.n 21 + Print field in front of tabulator n (max. 20) right-aligned, with dot leader (Takes effect when positioning to tabulator n.)

n = 0: The above is not valid for any field (default setting)
n = : The above is valid for all fields

&Dn 31 Inserting the current date.
n = : (not yet defined)
n = 1: Date written as xx.xx.xx (e.g. 23.01.90)
n = 2: Date written as xx. xxx. xxxx (e.g. 23. Jan. 1990)
n = 3: Date written as xx. xxxxxxxx xxxx (e.g. 23. Januar 1990)

&"n 35 Reference to the following footnote is not to be inserted at the end of the preceding word but n characters (print positions, not characters of input coding) farther to the left.

n = 0: Footnote reference to be inserted immediately after the last printed character of the preceding word (e.g. between "d" and "#1-" in "#1+Word#1-").
n = : (not yet defined)

&,n 36 Set word spacing before the next word to n blanks.

n = : (not yet defined)

&Xn 37 + If a total of more than n blanks must be added to a line to justify it, cancel justification for this line (default setting 999).

n = 0: (not yet defined)
n = : (not yet defined)

&Yn 38 + If more than n blanks must be inserted at any single position in the line to justify it, cancel justification for this line (default setting: 999).

n = 0: (not yet defined)
n = : (not yet defined)

&Sn 39 + Hyphenate only (besides positions marked by a "\") when more than n characters are available in the line for the word to be hyphenated (default setting: n=2). This instruction covers instruction 51 (@S+).

n = 1: Hyphenate only (in addition to positions marked by a "\") when justification of a

line would require more than one blank to be inserted at any single position.
n = 0: Hyphenate only at positions marked by a "\".
n = : (not yet defined)

&Wn 40 - Switch to the format having the number n (max. 9).

n = 0: Restore default settings
n = : Use settings from previously-used format.

@W+ 41 + Switch to the format having the next higher number

@W- 42 + Switch to previously-used format

@Z 43 Center following text (up to the next instruction for positioning or line break, or up to one of the instructions 43 to 46).

@/ 44 Align following text to the right (up to the next instruction for positioning or line break, or up to one of the instructions 43 to 46).

@. 45 as instruction 44, but insert dot leader in intervening space

@W 46 Repeat following text (up to the next instruction for positioning or line break, or to one of the instructions 43 to 46) as often as necessary to fill up intervening space (up to specified position or to the right margin). Caution: If the text to be repeated is terminated by one of the instructions 43 to 46, text will be repeated up to the right margin and a new line will be started.

@\+ 47 + If a word contains a space (blank; this can be either a blank coded as "_" or one that has been inserted as a result of replacement using parameter XPR and is thus not regarded as a separator between words), a new line may be started at this position if the word does not fit into the line.

@\- 48 + If a word contains a space (blank; this can either be one coded as "_" or one that has been inserted as a result of replacement using parameter XPR and is thus not regarded as a separator between words), a new line may not be started at this position. (default setting)

@\ 49 Do not hyphenate following word (except at positions marked with a "\")

@S- 50 + Hyphenation off (i.e. hyphenate only at positions marked with a "\")

@S+ 51 + Hyphenation on (default setting)

@R- 52 + Justification off

@R+ 53 + Justification on. However, solitary lines (i.e. a paragraph consisting of one line only) will not be justified (default setting)

@R 54 - Justify current line (However, instructions 43 to 46 have precedence over this instruction).

@- 55 + Keep extra blanks; line breaks in the input data will have the same effect as the instruction $$$. This means that the program attempts to preserve the text format of the input file. However, any instructions contained in the text will be carried out.

@+ 56 Cancels instruction 55 (default setting)

@D 57 Do not print following word. This instruction cancels instruction 58, if present.

@D- 58 - Print off (following text will not be printed, but the corresponding space will be left free)

@D+ 59 - Print on (default setting)

@I 60 Ignore next word or instruction. This allows the user to insert, for example, markers (which will not be printed) for index entries.

@I+ 61 - Ignore following words. For example, this allows the user to skip over index entries included in the text.

@I- 62 - Cancel instruction 61 (default setting)

@$ 63 New line, in case the following (absolute) positioning instruction cannot be executed as desired.

@$+ 64 - as instruction 63, but is valid for all subsequent positioning instructions. In this case, instruction 63 cancels instruction 64 for the next positioning to be carried out.

@$- 65 - Cancels instruction 64 (default setting)

@M 66 The line in which the following word is located is to be marked; this instruction cancels instruction 67, if it has been given.

@M+ 67 - Turns line marking on. The marking (vertical line in the right margin) can be used, for example, to mark additions to the text made in a new version.

@M- 68 - Turns line marking off (default setting)

@F 69 Footnote numbering to be reset to 1 (starting with the next instruction 70).

@F+ 70 Beginning of footnote

@F- 71 End of footnote

List of formatting terms:

The numbers refer to the related instructions

Notes:

Hyphenation

Hyphenated words present in the input data will be rejoined. Here a hyphen is considered to be a "-" which is the last character in a line (= input record) if the next-to-last character is also a "-", or if the next-to-last character is a letter and the third-to-last character is not a control character ($, &, @, \, _, #, %).

If a "-" located at the end of a line is not to be interpreted as a hyphen, a "\" should be added directly after the "-". The character "\" will not appear in the printout.

When words are rejoined, a hyphenated "ck" which has been written as "k-" und "k" (according to German hyphenation rules) will not be restored as "ck".

The program uses German hyphenation rules when hyphenating. In addition, a "\" can be used within a word to mark the position where hyphenation should preferably be carried out; "\\" can be used to mark a position which should not be hyphenated under any circumstances. If "\" is written at the beginning of a word, it can only be hyphenated at its "\" positions.

Any hyphenation carried out will not be reflected when the file is written to the DESTINATION file. A hyphenated word in the LISTING file will appear in its unhyphenated form in the DESTINATION file.

Paragraphs

The beginning of a paragraph also includes the end of the preceding paragraph. The program makes sure that neither the first line of a paragraph appears as the last line on a page (or column) nor that the last line of a paragraph appears as the first line on a page (or column). This may result in a page (or column) having one line less than its full length when printed, with one or two lines being carried over to the next page (or column) in order to meet these requirements.

If, however, the last line of a paragraph contains a footnote reference number, this line may appear at the top of a new page.

Continuation lines

Continuation lines are lines that are started by automatic line breaks, as opposed to those started by a line-break instruction (instruction 2-4).

Footnotes

Footnotes must be surrounded by the instructions @F+ and @F- (instructions 70 and 71). During formatting, they will be automatically numbered and placed at the bottom of the respective page. The footnote reference number (footnote number) will be printed directly after the preceding word. If the footnote number is not to be inserted at the end of the word but further to the left (e.g. before a comma at the end of a word), instruction 35 (e.g. &"1) can be given before instruction 70 in order to specify how many characters (print positions, not input character positions) farther to the left the footnote number is to be inserted.

Positioning

If a positioning instruction is to be executed (instructions 8 and 12-17) but the current position is to the right of the desired position, a blank will be inserted instead (unless instructions 63 or 64 have been given, specifying that in this case a new line is to be started and positioning is to take place in this new line). If, however, blanks are already present immediately to the left of the current position, the positioning instruction will be attempted once more, starting at the last non-blank position.

Hard spaces

If a space (blank) is not to be enlarged during justification, a "_" (underline dash) should be used instead of a blank. Such a hard space will not be considered as a separator character between two words.

Formats

A "format" consists of all format settings (e.g. left and right margins, tabulators) which may be altered by instructions marked with a "+". There are 9 formats, numbered from 1 to 9.

When the program is started, all formats will contain the default settings, and format 1 will be selected. Default settings are described along with the instructions marked with a "+". When another format is selected (using one of the instructions 40-42), the settings last defined for the respective format will take effect.

Formats can be helpful when working with tables whose layouts are often repeated. The necessary instructions for each table can be stored in a separate format; tabulators thus must be set only once for each table format.

Limitations:

Line markings

Lines may be marked (instructions 66-67) for single-column printing only (1st numerical value in the parameter DR = 1).

Footnotes

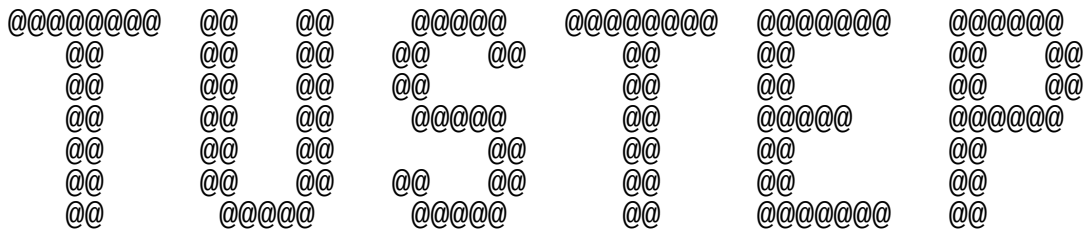
Footnotes (instructions 69-71) are possible for single-column printing only (1st numerical value of the parameter DR = 1).

1 1/2-line feed

A line feed of 1 1/2 is possible for single-column printing only (1st numerical value of the parameter DR = 1). In addition, such a line feed is possible only with dot-matrix and laser printers, not with line printers.

This limitation affects instructions 2 and 5. In case a 1 1/2-line feed is not possible, the instruction $0 will be treated as $, and the instruction &v0 as &v1.

* * * * *



Tübingen

System OF Text Processing Programs

Program

G E N E R A T E F O R M S

1993

TÜBINGEN UNIVERSITY - CENTER FOR DATA PROCESSING

Department of Literary and Documentary Data Processing

Brunnenstrasse 27, D-72074 Tübingen

Survey:

Command:

#GFORMS

Specifications:

SOURCE	= file		Name of the file containing the data from which forms are to be generated
	= -STD-		The standard TEXT file contains the data from which forms are to be generated
MODE	= -	*	The sequence of data in the LISTING file is to be the same as in the input file
	= -STD-		The input data is to be sorted in such a way as to yield sequential stacks of forms after cutting.
ERASE	= -	*	If the LISTING file already contains data, they are to be retained.
	= +		If the LISTING file already contains data, they are to be erased beforehand.
PARAMETER	= file		Name of the file containing parameters
	= *		The parameters follow the command and are ended by *EOF
FORM	= -	*	No form to be used as a mask
	= file		Name of the file containing a form to be used as a mask
LISTING	= -STD-	*	The generated forms are to be written into the standard LISTING file
	= file		Name of the file into which the generated forms are to be written

Features:

This command can be used to prepare data for printing in a given format (e.g. address labels, standardized letters, catalog cards, office forms). It is possible to specify:

- the size of the form
- standard text for each form (which can also depend on the occurrence of specific text parts in the respective form)
- line and character positions for each text part
- text parts which are to be repeated on continuation forms if there is not enough space for the text on one form

Parameters

Values in [ ] refer to the type of parameter employed. The various types of parameters are described in the "Parameters" chapter of "TUSTEP Basics".

Values in < > are default settings.

In certain parameters, beginning and end markers (as well as markers which serve as left and right parentheses) can be used to select text parts for further processing. The manner in which these parameters function is also described at the end of the "Parameters" chapter in "TUSTEP Basics".

In addition to the following parameters, other parameters can be used to define character groups and character strings. [v]

Parameters marked with a "+" must contain a 2 in column 7 if they are meant for a continuation form.

Selecting data

If the entire file is to be processed, none of the following parameters are necessary.

BER Definition of a single area ("page.line-page.line") or a starting point ("page.line"). This parameter is only used when not processing the entire input file. [xi]

If a segment of a segment file is to be processed, the name of the segment can be substituted for the area.

This parameter can only be used when the record numbers of the file are in ascending order.

MAX For test runs, this specifies from how many text units forms are to be generated, or the maximum number of forms (including continuation forms) or pages that are to be generated. [i]

Three numerical values can be specified here:

1st number: the maximum number of text units to be generated <999999>
2nd number: the maximum number of forms to be generated <999999>
3rd number: the maximum number of pages to be generated <999999>

Parameters for LISTING file settings

DR Specifications for printer output control [I]

Seven numerical values can be specified:

1st number: Columns <1>

Number of columns (forms) to be printed side by side on each page)

2nd number: Margin <10>

Number of blanks to the left of the first column

3rd number: Width <64>

Number of characters per column (form)

4th number: Number of blanks between columns (forms) <0>

5th number: Indent for continuation lines <0>

Number of blanks at the start of continuation lines before a line break forced by one of the character strings specified in parameter ZW

6th number: Indent after line break <0>

Number of blanks at the start of a line after a line break forced by one of the characters specified in parameter ZW.

7th number: Blanks at the start of continuation lines <0>

Blanks at the beginning of a continuation line after a line break has already been forced with one of the characters specified in parameter ZW.

DRZ Additional specification for printer output control [I]

Seven numerical values can be specified:

1st number: Header <3>

Number of lines for the header (including blank lines)

2nd number: Column height <60>

Number of lines per column (form) (excluding lines for the header and footer)

3rd number: Footer <0>

Number of lines for the footer (including blank lines)

4th number: Columns <1>

Number of forms per column

5th number: Repetition of footer text <0>

Number of lines of the page's footer text (counting from the first line down) which are to be repeated after every form column (except for the last column) <0>

6th number: Space between forms <0>

Number of blank lines between each row of forms

7th number: Repetition of header text <0>

Number of lines of the page's header text (counting from the botom line to the top) which are to be repeated before every form column (except for the first column).

KT Text parts to be printed at the top of every page as a header [II]
<"file name" xx. xxx. xxxx xx.xx xxxxxx>

To insert the current date, enter "xx. xxx. xxxx" or "xx.xx.xx" at the appropriate position. Positions for the current time may be indicated by "xx.xx" and for the page number by "xxxxxx" (2 to 6 "x"s, but at least as many positions as necessary for the page number). If "- xxxxxx -" is entered for the page number, it will appeared centered on the page between two minus signs, with each minus sign separated from the page number by a space having the width of up to one whole blank.). However, the date, time and page number can be inserted only one time each.

If a character string begins with a "*:", the rest of the character string serves as a header for every text column. If a numeral n is entered in place of the asterisk, the rest of the character string is used as a header for the nth column. If the numeral given equals 0, the rest of the character string is used for the entire line. If a character string does not begin as just described, "0:" is assumed (standard value).

The following rules determine which line of the header is used by the character string: A character string which is designated for an entire line will be printedat the start of a new line (starting with the first line). A character string which is designated for a particular column will be printed in the same line as the preceding character string, unless this line contains text meant for an entire line, for the

same column, or for a column further to the right. In this case, the character string will be printed in the next line.

Each of the specified character strings can be arranged in three parts by using the formating instructions "@z" and "@/":

left-aligned @z centered @/ right-aligned

The individual parts will be inserted left-aligned, centered and right-aligned. An individual part may be omitted; in this case the formatting instructions in front of the second and third parts may also be omitted.

FT Text parts (analogous to that described in the parameter KT) to be printed as a footer at the bottom of every page. [ıı]

Date, time and page number may not be entered in the footer if already specified to appear in the header.

SM Specifies that a page with cutting marks is to be printed after every n pages. [ı] <100>

DRT Printing device for which the data are to be prepared. This parameter is obligatory. [xı]

The types of available printers depends on the actual computer being used. To obtain a list of these, use the command

#LIST, PRINTERS

Specifications for numbering

NR Initial page number for output [ı] <1>

Organizing records into a text unit

In case each input record contains a complete text unit (i.e. all data for a form including any necessary continuation forms), the following parameters should not be used. Otherwise the parameters AA and/or AE can be used to organize more than one record into a text unit.

If one of the following four parameters is specified, any blanks located either before or after the input record will be eliminated before the parameter is evaluated.

When records are being organized into a text unit, a blank will be inserted between each input record; no blank will be inserted at positions where hyphenation protect has been specified (see parameter STR).

ANR Specifies whether successive records, whose record numbers either partially or completely match, are to be organized into a text unit. [I] <0>

One numerical value may be specified:

0 = Do not organize records into a text unit on the basis of record numbers.
1 = Organize successive records having the same page number into a text unit.
2 = Organize successive records having the same page and the same line number (regardless of the distinction number) into a text unit.
3 = Organize all successive records with the same record number into a text unit.

If 0 is specified (default setting), records will be organized into text units on the basis of the two following parameters only. If one of the values 1 to 3 is specified, the resulting text units may be broken down further on the basis of the two following parameters.

AA Character strings placed at the beginning of a record (after leading blanks have been eliminated) which mark the start of a text unit. [VIIIa]

AE Character strings placed at the end of a record (after any trailing blanks have been eliminated) which mark the end of a text unit. [VIIIb

STR Hyphenation [I] <0>

0 = Input data are not hyphenated
1 = Rejoin hyphenated words during input

Here a hyphen is considered to be a "-" which (after trailing blanks have been eliminated) is the last character in an input record if the second-to-last character is also a "-" or a letter and the third-to-last character is not a control character ($, &, @, \, _, #, %).

When hyphenation is turned off, a hyphenated "ck", which according to German hyphenation is written as "k-" and "k", will not be restored to its "ck" form.

Replacing character strings

XX Pairs of character strings (and exception strings). The first character string of a pair will be replaced in the text unit by the pair's second character string. [x]

Supplementing a text part

ERG Text part which is to be supplemented before every text unit. [II]

Here standard text for forms can be specified in advance. The markers contained therein are used to designate the position in the form where it is to be printed.

Dividing the text unit into form fields

TTK Character strings marking the individual text parts. The text parts will be stored in the form in the same order as their corresponding markers are given in this parameter. Should a marker occur more than once in the data, the text parts which it marks will be organized into a single text unit. The markers will not show up in the form field. [Ix]

TTF Specifies (parallel to TTK) whether a text part is to be stored in its own form field, or whether it is to be appended to another text part. [I] <0>

0 = Do not append text part to any other text part
n = Text part is to be appended to the nth text part (i.e. the text part marked by the nth character string in the parameter TTK). The latter can in turn be appended to another text part.

This process is carried out from left to right (in the order specified in TTK). A text part can be appended to any text part whose marker (as given in TTK) is located further to the right. If its marker is located to the left, the text part may be appended only if this text part has not yet been appended to another text part.

EGA / EGE Text parts (parallel to TTK) which are to be added to the beginning or end of the respective text part. Depending on the occurence of certain text parts (markers), standard form text may be supplemented. [II]

Text parts inserted in this manner will not be checked for any markers defined with parameter TKK. If such markers happen to be present, they will be treated as text.

AZN + Beginning line numbers (parallel to TTK) of the individual form fields. [I] <0>

0 = Ignore text part
n = Store text part after the nth line. If the nth line is already occupied, the text part may be stored in the next available line, provided that the corresponding value of the parameter EZN allows for this.

In this parameter, the value 0 can be given for text parts which are to be appended to another text part as specified with parameter TTF. In this case, the 0 is merely used as a temporary marker so that the following numerical values can be properly assigned to the character strings specified in parameter TTK.

EZN + Final line number (parallel to AZN) of the individual form field. [I] <0>

0 = Text part may be stored in one line only (namely, the one specified in the parameter AZN).
n = Text part may be stored anywhere, starting from the line specified in the parameter AZN up to the nth line specified here.

DPA + Beginning printing position (parallel to AZN) of the individual form fields. [I] <1>

This defines the first position from which the text part in the individual lines (as specified in AZN and EZN) may be stored.

DPE + End printing position (parallel to AZN) of the individual form fields. <3rd numerical value of parameter DR>

This defines the last position up to which the text part in the the individual lines (as specified with AZN and EZN) may be stored.

ZEN + Specifications (parallel to AZN) regarding the centering of text in individual form fields. [I] <0>

0 = left-aligned
1 = centered
2 = right-aligned
3 = justified

If vertical centering is also desired, these values are to be added to as follows:

00 = up
10 = in the middle
20 = down

ZW Character strings which mark the start of a new line. None of these character strings will be printed. [Ix]

ZA / ZE Text parts are to be divided into lines so that the character strings specified in ZA and ZE are located at the beginning and end of a line, respectively. [Ix]

ZAB / ZEB If a text part has to be divided into more than one line (in addition to the division required by the parameters ZW, ZA, ZE), division should preferably be carried out so that character strings specified in ZAB and/or ZEB will be located at the beginning and end of a line, respectively. [Ix]

TTS Specifies (parallel to TTK) under what conditions hyphenation should be attempted in case the corresponding text part must be divided into more than one line (in addition to division required by the parameters ZW, ZA, ZE, ZAB, ZEB). [I] <2>

0 = Hyphenate only at points marked by a \
1 = Hyphenate at points other than those marked by a \ only when justification would require that more than one blank space be added.
n = Hyphenate at points other than those marked by a \ only when more than n characters are available in the line.

TAB Character strings marking a jump to the next (logical) tabulator position. The individual character string will not be printed. If the tabulator position has already been passed, no jump will be carried out. [Ix]

TNR Specifies (parallel to TAB) that whenever a character string is encountered which has been specified in the parameter TAB, a jump may be made not only to the next tabulator position but also to at least one other specific tab stop (namely, the tabulator position given here). In case the number of the next (logical) tabulator is larger than the number specified here, a jump will be made to this tabulator. [I]

TPO Tabulator positions, i.e. starting positions of individual tabulator stops [I]

TFZ Specifications (parallel to TPO) for positioning text at the individual tab stops. [I] <0>

0 = left-aligned
1 = centered
2 = right-aligned

TTV Specifies (parallel to TTK) the number of form fields (of identical size) into which the text part is to be divided. Here the data for the 2nd and following form fields are treated as overlay data, i.e. they are stored in the form field specified in the parameter UEF. Please note that these form fields must always have the same size. [I]

UEL + Specifies (parallel to TTK) what steps should be taken in case a text part does not fit in the specified form field. [I] <0>

0 = Insert "..." at the last 3 positions accompanied by an error message
1 = Insert "..." at the last 3 positions with no error message
2 = The rest is to be stored in the form field specified in parameter UEF
3 = Generate continuation form
4 = Ignore the rest, with error message
5 = Ignore the rest, with no error message

UEF + Specifies (parallel to TTK) which form field the rest of the text part is to be stored in. (cf. parameter UEL).

n = Number of the form field

Here n may be larger than the number of character strings (markers) specified in parameter TTK. In this case, the corresponding specification should also be made in parameters AZN, EZN, DPA, DPE, ZEN and UEL.

UEW Specifies (parallel to TTK) which text part is to be repeated in the continuation form. [I] <0>

0 = Do not repeat text part
n = Repeat text part. If hyphenation characters have been specified in the parameter UTR, the text part will be repeated only up to the point where the nth hyphenation character is located. Any selections made on the basis of the parameters AUL, EUL, (UL and)UL will be carried out beforehand.

AUL / EUL Character strings marking the beginning / end of the part of a text part to be repeated in the continuation form. [Ix]

(UL /)UL Parenthesis for selecting a part of the corresponding text part (in case AUL and/or EUL has not been specified) or for eliminating parts of a text part already selected with AUL/EUL which is to be repeated in the continuation form. [Ix]

ULI Additional specifications for AUL, EUL and (UL,)UL [I] <1,0>

Two numerical values can be specified:

1st number: Specification analogous to parameter AEI for COPY
2nd number: Specification analogous to parameter KLI for COPY

UTR Separator character string, up to which point a text part is to be repeated in continuation forms (see also parameter UEW) [Ix]

UTE Text part to replace text eliminated by parameter UTD. [II] </ .../>

FNR Specifies at which point in the form a running form number is to be inserted. Continuation forms are not counted here; they are assigned the same number as their corresponding initial form. [I]

Four numerical values can be specified here:

1st number: Line number
2nd number: First character position
3rd number: Last character position
4th number: Number of the first form <1>

NFF Specifies at which point in a form a running number is to be inserted in case continuation forms must be generated. Please note that the specified field should be large enough to accomodate character strings specified in the parameter NFE. [I]

Three numerical values can be specified here:

1st number: Line number
2nd number: First character position
3rd number: Last character position

NFE Text parts to be inserted at the beginning/end of the field specified in parameter NFF, in case continuation forms must be generated. [II] </-/-/>

MFF Maximum number of continuation forms to be generated (for one text unit). [I] <2>

If the number specified here is exceeded, the rest of the text unit will not be processed and the appropriate error message will appear.

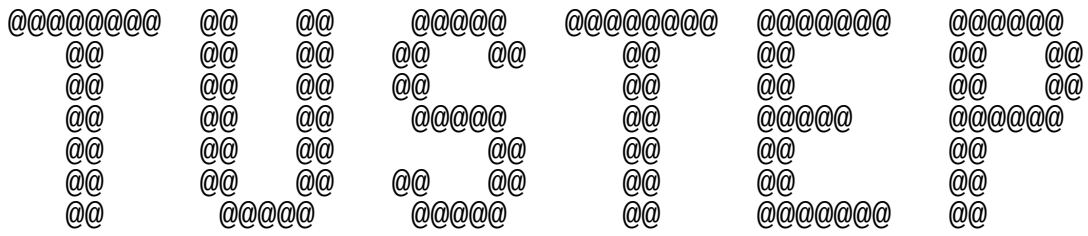
LCH Specifies which position in the form must always remain blank (e.g. to leave space for binder holes in catalog cards). This position will be skipped when it is located within a form field where a text part is to be printed. [I]

Four numerical values can be specified here:

1st number: First line
2nd number: Last line
3rd number: First character position
4th number: Last character position

Alphabetical list of parameters

* * * * *



Tübingen

System of Text Processing Programs

Program

G E N E R A T E I N D E X

1993

TÜBINGEN UNIVERSITY - CENTER FOR DATA PROCESSING

Department of Literary and Documentary Data Processing

Brunnenstrasse 27, D-72074 Tübingen

Survey:

Command:

#GINDEX

Specifications:

SOURCE	= file		Name of the file containing the index entries to be edited or containing the entries for the KWIC index.
	= -STD-		The standard TEXT file contains the index entries to be edited or the entries for the KWIC index.
DESTINATION	= -	*	Output to the LISTING file only
	= file		Name of the file to which the generated index is to be written
	= -STD-		The generated index is to be written to the standard TEXT file.
MODE	= +	*	Generate index; the input data contain a reference field
	= -		Generate index; the input data do not contain a reference field
	= KWIC		Generate KWIC index
ERASE	= -	*	If the DESTINATION file or the LISTING file already contains data, they are to be retained.
	= +		If the DESTINATION file or the LISTING file already contains data, they are to be erased beforehand.
PARAMETER	= file		Name of the file containing parameters
	= *		The parameters follow the command and are ended by *EOF.
DATA	= -	*	SOURCE file contains the index entries in their entirety.
	= file		Name of the file containing that part of each index entry or of each KWIC index entry that was not required for sorting
	= -STD-		The standard DATA file contains that part of each index entry or of each KWIC index entry that was not required for sorting.
LISTING	= -STD-	*	The formated index is to be written to the standard LISTING file.
	= file		Name of the file to which the formated index is to be written.

= - No output of formated index

Features:

With this command index entries and text units can be processed and combined into an index (list of word forms; KWIC index) or directory (e.g. bibliographies). The SOURCE file contains the index entries or text units which have been generated and prepared for sorting with PINDEX or PRESORT and have then been sorted with SORT. Other data can be processed also, provided each input record contains a text unit.

Among other features, this program lets the user add control characters and markers, choose any format for the print output, generate titles and running headers, select index entries and text units, structure index entries hierarchically into entries and subentries, and calculate absolute and relative frequencies.

Notes:

For printing the generated index or directory with the command #PRINT, the LISTING file must be used, *not* the DESTINATION file. The DESTINATION file is needed when the index or the directory is to be used for further processing (e.g. for photo composition); in this case, it is not possible to generate a LISTING file simultaneously with this program.

Parameters

Values in [ ] refer to the type of parameter employed; they are defined in the "Parameters" chapter of "TUSTEP Basics".

Values in < > refer to default settings.

In addition to the following parameters, other parameters can be used to define character groups and character strings. [v]

An "n" located after the parameter identification stands for a digit in column 7 of the parameter. It designates the text part or the reference part referred to by the respective parameter.

Notes concerning supplementary text based on entry type

When generating an index, the user can also insert additional character strings at different positions of an index entry. These character strings can be specified with the appropriate parameter. The description of each of these parameters will specify that such inserted character strings are added in a "type-dependent" manner. This means that for these parameters a corresponding character string can be specified for each type of index entry possible. The first character string is valid for type 1 index entries, the second string for type 2 entries, etc. It may be necessary to specify empty character strings for missing entry types in order to keep all assignments in their proper order. If a character string (even an empty one) is not specified for all entry types occurring in such parameters, the last character string given in the parameter will be inserted for the missing character string. Therefore, if the same character string is to be inserted regardless of the type of index entry, this character string has to be specified only once with the appropriate parameter.

Notes concerning the KWIC index

For the purposes of this program, a KWIC index is, in a strictly logical sense, an index of word forms (with each index entry consisting of a single text part), where

- the text part consists of the respective key word,
- references are not grouped together,
- each reference starts in a new line,
- each reference is followed by the context, in which the key word appears.

Thus the same parameters can be used for a KWIC index as for an index of word forms. However, some parameters are irrelevant for generating a KWIC index, such as those used to combine

references. Furthermore, additional parameters are available for an entry's context.

Parameters for LISTING file settings

DR Printer output control [I]

Four numerical values can be specified:

1st number: Columns <1>

Number of columns appearing side-by-side on every page

2nd number: Left margin <10>

Number of blanks to the left of the first column

3rd number: Column width <64>

Number of characters per column

4th number: Space between columns <0>

Number of blank spaces between columns

DRZ Additional specifications for printer output control [I]

Seven numerical values can be specified:

1st number: Header <3>

Number of lines for the header (including blank lines)

2nd number: Height of column <60>

Number of lines per column (excluding lines for the header and footer).

3rd number: Footer <0>

Number of lines for the footer (including blank lines)

4th number: Columns <1>

Number of columns per page

5th number: Repetition of footer text

Number of lines of the page's footer text (counting from the first line down) which are

to be repeated after every column (except for the last column).

6th number: blank lines <0>

Number of blank lines between individual columns

7th number: Repetition of header text <0>

Number of lines of the page's header text (counting from the bottom line to the top) which are to be repeated before every column (except for the first column).

KT Text parts to be printed at the top of every page as a header [II]
<"file name" xx. xxx. xxxx xx.xx xxxxxx>

To insert the current date, enter "xx. xxx. xxxx" or "xx.xx.xx" at the appropriate position. Positions for the current time may be indicated by "xx.xx" and for the page number by "xxxxxx" (2 to 6 "x"s, but at least as many positions as necessary for the page number). If "- xxxxxx -" is entered for the page number, it will appeared centered on the page between two minus signs, with each minus sign separated from the page number by a space having the width of up to one whole blank. However, the date, time and page number can be inserted only one time each.

If a character string begins with a "*:", the rest of the character string serves as a header for every text column. If a numeral n is entered in place of the asterisk, the rest of the character string is used as a header for the nth column. If the numeral given equals 0, the rest of the character string is used for the entire line. If a character string does not begin as just described, "0:" is assumed (standard value).

The following rules determine which line of the header is used by the character string: A character string which is designated for an entire line will be printed at the start of a new line (starting with the first line). A character string which is designated for a particular column will be printed in the same line as the preceding character string, unless this line contains text meant for an entire line, for the same column, or for a column further to the right. In this case, the character string will be printed in the next line.

Each of the specified character strings can be arranged in three parts by using the formatting instructions "@z" and "@/":
left-aligned @z centered @/ right-aligned
The individual parts will be inserted left-aligned, centered and right-aligned. An individual part may be omitted; in this case the formatting instructions in

front of the second and third parts may also be omitted.

KTZ Specifies whether the header text is to be printed on the first page. [I] <0>

0 = Suppress header text
1 = Print header text

FT Text parts (analogous to those described in parameter KT) to be printed as a footer at the bottom of every page. [II]

Date, time and page number may not be inserted in the footer if already specified for the header text.

DRT Printing device for which the data are to be prepared. [XI]

This parameter is obligatory.

The types of available printers depends on the actual computer being used. To obtain a list of these, use the command
#LIST,PRINTERS.

NR Initial page number for output [I] <1>

ROM Specifies whether Arabic numerals or Roman numbers are to be used for the page number in the header or footer text. [I] <0>

0 = Arabic numerals
1 = Roman numbers in lowercase letters
2 = Roman numbers in uppercase letters

DRE n Printer output control for a new entry starting with text part n. [I]

Six numerical values can be specified:

1st number: Line feed before entry <1>

Number of line feeds (= blank lines + 1) before a new entry. When specified between two entries the larger of this parameter's 1st and 3rd numbers will be used for line feed control.

2nd number: Remaining lines <1>

Minimum number of lines which must be available in a column for a new entry to be started in this column. Otherwise the entry will be started in a new column.

3rd number: Line feed after an entry <1>

Number of line feeds (= blanks lines + 1) after an entry. If specified between two entries, the larger of this parameter's 1st and 3rd numbers will be used for line feed control.

4th number: Indentation for new entry <0>

Number of blanks placed before a new entry

5th number: Indentation of continuation lines <4>

Number of blanks placed before continuation lines.

6th number: Line feed before continuation lines <1>

Number of line feeds (= blank lines + 1) before continuation lines.

DRA n Printer output control for a change in the initial letter of an entry beginning with text part n. [I]

Specifications correspond to those in parameter DRE.

Input data format The specifications entered for the following four parameters must have the same values as those used for preparing the data with the programs PINDEX or PRESORT.

One exception, however, applies to parameters SNL and SSL: If the sort number and/or the sort key have been eliminated during a sort of the prepared data (as specified with the DELETE parameter of the command #SORT), the parameters SNL and/or SSL must be used to specify the value zero.

Parameter SSL is obligatory; all others may be omitted if they were not specified when the data were prepared.

IRL Internal reference length [I]

SNL Length of sort number [I]

SSL Length of sort key [I]

HFL Length of frequency specification [I]

Selecting input data

MAX For trial runs, this specifies the maximum number of entries to be either processed or outputted, or the maximum number of pages to be prepared. [I]

Three numerical values can be specified:

1st number: maximum number of entries to be read <999999>
2nd number: maximum number of entries for output <999999>
3rd number: maximum number of pages for output <999999>

Specifying type of index entry

Every index entry is assigned to an index type by which it can be classified.

However, when MODE=- index entries are not specified according to their type; here the index entries will be treated as if they were all assigned to entry type 1.

TYP Redefining entry type 0 as another type [I] <1>

TXT Specifies whether the entry - depending on its type - (including any references it may have) is to be outputted (1) or not (0). [I] <1,1,...>

Each selection is based on the entry's original type; evaluation of parameters TFT and TFR, if specififed, takes place after this parameter is processed.

REF Specifies whether the reference (depending on its type) is to outputted (1) or not (0). [I] <1,1,...>

Each selection is based on the entry's original type; evaluation of parameter TFR, if specified, takes place after this parameter is processed.

Identical index entries or identical text parts of index entries will not be compiled unless they have been assigned to the same entry type. Any redefinitions of the original entry type as specified with parameter TFT will be taken into account. The same also applies to the grouping of identical references or reference parts, with any redefinitions specified in parameter TFR being taken into account.

If the program is to group together index entries, which for example were originally assigned different types yet whose references differ in terms of their entry type (e.g. italics or

supplemented text, cp. "Notes concerning type-dependent insertions" page 503), the following steps should be taken.

- Assign the same type to each text part of these index entries using parameter TFT.
- Assign a different type to each reference part using parameter TFR if it has been specified.

TFT n Redefining index types for text part n. [I] <1,2,3,...>

TFR n Redefining index types for reference part n. [I] <1,2,3,...>

Defining the text parts of index entries

Index entries consisting of more than one text part (e.g. main and secondary terms) can be subdivided with the use of separator characters. In this case, parameter TT must specify the maximum number of text part divisions, and parameter TR must specify all separator strings. If index entries are not to be subdivided, none of the following parameters need to be specified.

TT Maximum number of text parts (up to 9) which may comprise an index entry [I] <1>

TR Character strings used to separate text parts [Ix]

TRN Specification parallel to TR: the given character string shall count as a separator character for at least the nth level (separator number) [I] <1,1,...>

TRV Specification parallel to TR: the given character string is to be used as a separator character from the nth level only [I] <1,1,...>

TRB Specification parallel to TR: the given character string is to be used as a separator character up to and including the nth level [i] <9,9,...>

TRU Specification parallel to TR whether the separator character should be suppressed (1) or not (0). [I] <0,0,...>

Regardless of the specification made in this parameter, the separator character between two text parts will always be suppressed when a new line is started at this position during output of the index entry due to the parameters NZB and NZ described below.

Inserting/suppressing line breaks

NZB New line after every text part (with omission of the separator character) up to and including the text part whose number is entered here. [I] <0>

NZ n New line (with omission of the separator character) after text part n: 0 = no, 1 = yes [I] <0>

NZU Suppress new line starting with the specified text part m: if the first m-1 text parts in successive entries are identical, no line break will be carried out in front of such entries. [i] <10>

Selecting index entries and key words

If not index entries (when MODE=KWIC all key words) are to be included in the output, the following parameters (in addition to the selection parameter TYP) can be used to select those index entries which should be included in the output.

If the nth text part of an index entry (when MODE=KWIC: of the key word) matchs the string specified in Parameter T+ or T+U, the entry will be included in the output. In this case the entry will not be subject to any more checks based on the remaining parameters described in this section. If no other parateters in this section have been specified, only these index entries will be outputted.

T+ n Text parts, of which one must match text part n of an index entry or key word for the index entry to be included in the output. [III]

No distinction is made between uppercase and lowercase letters.

T+U n Text parts, of which one must match text part n of an index entry or keyword for the index entry to be included in the output. [III]

A distinction will be made between uppercase and lowercase letters.

If the nth text part of an index entry (when MODE=KWIC of the key word) does not match the string specified in parameter T- or T-U, the index entry will not be included in the output. In this case the entry will not be subjected to any more checks based on the remaining parameters in this section.

T- n Text parts, none of which may match text part n of an index entry or keyword for the entry to be included in the output. [III]

No distinction is made between uppercase and lowercase letters.

T-U n Text parts, none of which may match text part n of an index entry or keyword for the entry to be included in the output. [III]

A distinction will be made between uppercase and lowercase letters.

The parameters ZF+, TA+ and TE+ can abe used to specify conditions under which an index entry is to included in the output. If one or more of these parameters are used, at least one of the specified conditions must be fulfilled if the index entry is to be included in the output.

ZF+ n Character strings, of which at least one must occur in the nth text part of the index entry or in the key word if the index entry is to be outputted to the index. [IX]

TA+ n Character strings, of which at least one must match the nth text part of the index entry, or the beginning of the key word, if the index entry is to be outputted to the index. [VIIIa]

TE+ n Character strings, of which at least one must match the end of the nth text part of the index entry, or match the end of the key word, if the index entry is to be outputted to the index. [VIIIb]

The parameters ZF-, TA- and TE- can be used to specify conditions under which an index entry is not to be included in the output. If one or more of these parameters are used, an index entry will be omitted from the output if it fulfulls one of the specified conditions.

ZF- n Character strings, none of which may occur in the nth text part of the index entry, or in the key word, for the index entry to to be written to the index. [IX]

TA- n Character strings, none of which may match the beginning of the nth text part of the index entry, or the beginning of the key word, for the index entry to be written to the index. [VIIIa]

TE- n Character strings, none of which may match the end of the nth text part of the index entry, or the end of the key word, for the index entry to be written to the index. [VIIIb]

If one or more of the parameters ZF+, TA+ and TE+ as well as one or more of the parameters ZF-, TA- and TE- have been specified, an index entry must fulfill at least one of the conditions of parameters ZF+, TA+ or TE+ and none of the conditions of parameters ZF-, TA- or TE- to be included in the output.

Treatment of identical text parts

GKU Specifies whether a distinction should be made between uppercase and lowercase letters (1) or not (0) when comparing entries for identical text parts in the grouping of index entries. [I] <0>

TTE n Character strings, which (depending on the type of entry) are to replace text part n (which is omitted due to being identical to the same text part of the preceding entry) [II] < — > < — — > ...

TTW n Allow repetition starting with the specified text part in case identical text parts of the preceding levels would be omitted up to text part n [I] <n+1>

Replacing text parts or characters within text parts

TTT n Replacing text part n [IV]

XTT n Replacing character strings in text part n [X]

LTT n Character strings, which (depending on the type of entry) are to be inserted as a replacement for an empty text part n [II]

Adding characters to individual text parts

VTT n Character strings, which (depending on the type of entry) are to be inserted in front of text part n. [II]

NTT n Character strings, which (depending on the type of entry) are to be inserted after the text part n. [II]

Positioning text parts

The following two parameters affect only the LISTING file.

TTP n Horizontal position in the LISTING file which is to be reached after text part n (if necessary, to be filled out with blanks) [I] <0>

TTZ n Additional specification to TTP: in the field whose right border has been defined by TTP, the text part n is to be printed left-aligned (0) or right-aligned (1) [I] <0>

Inserting a running number before text parts

A running number will only be inserted if the value specified for parameter LN is not equal to zero.

LN n Length of the field to be reserved for the running number in front of text part n [I] <0>

LNB n Basic value which is to be added to the running number for text part n. [I] <0>

LNN n The running number for text part n is to be reset to 0 if the text part n changes with a specified (or higher) number. [I] <0>

LNW n If text part n is repeated (see parameter TTW), the running number is either to be repeated (1), or not to be repeated (0) [I] <1>

Inserting characters around the running number

VLN n Character strings, which (depending on the type of entry) are to be inserted before the running number located in front of text part n. [II] < >

NLN n Character strings, which (depending on the type of entry) are to be inserted after the running number located in front of text part n. [II] < >

Positioning the running number

The two following parameters affect only the LISTING file.

LNP n Horizontal position in the LISTING file to be reached after the running number (if necessary, to be filled out with blanks) [I] <0>

LNZ n Additional specification to LNP: in the field whose right border has been defined by LNP, the running number is to be printed left-aligned (0) or right-aligned (1) [I] <0>

Inserting the absolute frequency after text parts

The absolute frequency will inserted only if the value specified for parameter AH is not equal to zero.

AH n Length of the field to be reserved for the absolute frequency after text part n. [I] <0>

AHW n If text part n is repeated, the absolute frequency is also to be repeated (1), or is not to be repeated (0). [I] <0>

Inserting strings around the absolute frequency

VAH n Character strings, which (depending on the type of entry) are to be inserted before the absolute frequency following text part n. [II] < (>

NAH n Character strings, which (depending on the type of entry) are to be inserted after the absolute frequency following text part n. [II] <)>

Positioning the absolute frequency

The following two parameters affect only the LISTING file.

AHP n Horizontal position in the LISTING file which is to be reached after the absolute frequency following text part n (if necessary, to be filled out with blanks) [I] <0>

AHZ n Additional specification to AHP: in the field whose right border has been defined by AHP, the absolute

frequency located after text part n is to be printed left-aligned (0) or right-aligned (1) [I] <0>

Inserting the relative frequency after text parts

The relative frequency is inserted only when the value specified in parameter RH is not equal to zero.

RH n Length of the field which is to be reserved for the relative frequency after text part n. [I] <0>

RHD n Decimal places for relative frequency after text part n [I] <2>

RHB n The relative frequency after text part n is not based on the total number of records that are read, but on the frequency of the higher-level text part whose number is specified here. [I] <0>

RHW n If text part n is repeated, the relative frequency is also to be repeated (1), or not to be repeated (0). [I] <0>

Inserting characters around the relative frequency

VRH n Character strings, which (depending on the type of entry) are to be inserted in front of the relative frequency following text part n [II] < >

NRH n Character strings, which (depending on the type of entry) are to be inserted after the relative frequency following text part n. [II] <^ˇ

Positioning the relative frequency

The following two parameters affect only the LISTING file.

RHP n Horizontal position in the LISTING file which is to be reached after the relative frequency following text part n (if necessary, to be filled out with blanks). [I] <0>

RHZ n Additional specification to RHP: in the field whose right border has been defined by RHP, the relative frequency located after text part n is to be printed left-aligned (0) or right-aligned (1). [I] <0>

Inserting characters around the references

VRF Character strings which (depending on the type of the last text part) are to be inserted before the first reference [II] < >

NRF Character strings which (depending on the type of the last text part) are to be inserted after the last reference. [II] <>

ZRF n Character strings which (depending on the type of the last reference part) are to be inserted between the individual references if the second reference separated by ZRF begins with reference part n [II] < >

Defining reference parts

RFL Numerical values which specify the length of the individual reference parts (see programm PINDEX) [I] <6>

The number of reference parts is determined by the number of numerical values specified (up to 9 values possible).

Selecting references

MRF The output should also include references originally excluded in PINDEX (parameter ORF) (1) or not (0). [I] <0>

RFF An unbroken sequence of ascending references numbers is to be grouped together (1) or the references are to be listed individually (0). [I] <0>

RFU Suppress output of references if more than the specified number of references would be outputed. [I] <9999>

Note: References of index entries which are assigned to a specific type can be suppressed with parameter REF (see page 508).

RTU n Suppress reference part n (1); do not suppress (0). [I] <0>

Treatment of identical reference parts

RTE n Character strings which (depending on the type of reference) are to replace reference part n, which has been omitted due to its identity with the same part of the preceding reference). [II] <>

RTW n Allow reference repetition, starting with reference part whose number is specified here if the preceding reference would have caused the omission of identical higher-level reference parts up to part n.[I] <N+1>

Replacing reference parts or characters in reference parts

TRT n Replacing reference part n [IV]

XRT n Replacing character strings in reference part n [X]

LRT n Character strings, which (depending on the type of reference) are to be used as a replacement for an empty reference part-n [II] <>

Adding characters to reference parts

VRT n Character strings which (depending on the type of reference) are to be inserted before reference part n. [II] <>

NRT n Character strings which (depending on the type of reference) are to be inserted after reference part n. [II] <>

Adding characters to grouped references

References will be grouped into pairs if so requested with parameter RFF, and if the references involved have not been assigned a different treatment as specified in parameters F1, F2, and F3. In addition, references marked by the parameters VON and BIS in the program PREPARE INDEX (PINDEX) will be grouped into reference pairs; any references located between such pairs will be ignored.

F Character strings which (depending on the type of reference) are to be inserted after a reference marked by the parameter MF in the program PINDEX. [II] <f>

FF Character strings which (depending on the type of reference) are to be inserted after a reference marked by the parameter MFF in the program PINDEX. [II] <ff>

F1 Character strings which (depending on the type of reference) are to be inserted after a reference which would be followed by exactly a reference having the next-higher number (cp. parameter RFF). [II]

F2 Character strings which (depending on the type of reference) are to be inserted after a reference which would be followed by exactly two references each having the next-higher number than the reference preceding them. (cp. parameter RFF) [ii]

F3 Character strings which (depending on the type of reference) are to be inserted after a reference followed by three or more references, each having the next-higher number than the reference preceding them (cp. parameter RFF). [II]

ZRP n Character strings (depending on the type of reference) for reference grouping are to be inserted between a reference pair if the second reference of the pair begins with reference part n. [II] <->

Positioning reference parts

The two following parameters affect only the LISTING file.

RTP n Horizontal position in the LISTING file which is to be reached after reference part n (if necessary, to be filled out with blanks). [I] <0>

RTZ n Additional specification to RTP: in the field whose right border has been defined by RTP, reference part n is to be printed left-aligned (0) or right-aligned (1). [I] <0>

Inserting reference frequency specifications

The reference frequency will only be inserted if the value specified for parameter HF does not equal zero.

HF Field length of the frequency specification located after a reference in case the index heading occurs more than once with the same reference. [I] <0>

HFE Character strings which (depending on the type of reference) replace a missing reference frequency specification (because this is 1). [II] <>

Adding characters to a reference frequency specification

VHF Character strings which (depending on the type of reference) are to be inserted before the reference frequency specification. [II] < (>

NHF Character strings which (depending on the type of reference) are to inserted after the reference frequency specification. [II] <)>

Positioning the reference frequency specification

The following two parameters affect only the LISTING file.

HFP Horizontal position in the LISTING file which is to be reached after the reference frequency specification (if necessary, to be filled out with blanks). [I]<0>

HFZ Additional specification to HFP: in the field whose right border has been defined by HFP, the reference frequency specification is to be printed left-aligned (0) or right-aligned. (1) [I] <0>

Adding characters to the context (only when MODE=KWIC)

VK Character strings which (depending on the type of entry) are to be inserted before the context. [II] <>

NK Character strings which (depending on the type of entry) are to be inserted after the context. [II] <>

Adding characters to the key word (only when MODE=KWIC)

The following two parameters refer to the key word located in the context. Whenever a new key word is added, it will be written in an additional line of its own. The program will treat this key word as a (single) text part of the index entry. For this reason, specifications regarding this key word must be made with the corresponding parameters for text part 1.

VSW Character strings which (depending on the type of entry) are to be inserted before the key word in the context . [II] <>

NSW Character strings which (depending on the type of entry) are to be inserted after the key word in the context. [ii] <>

Positioning the key word (only when MODE=KWIC)

The following two parameters affect only the LISTING file. They refer the key word located in the context (see note in the two preceding parameters).

SWP Horizontal position to be reached either before or after the key word. [I] <0>

SWZ Additional specification to SWP: The position specified in parameter SWP is to be reached before (0) or after (1) the key word. [I] <0>

Inserting a running header

At present, the parameters for running heads are only evaluated for output to the DESTINATION file (but not to the LISTING) file.

LK t n Text part n of the index entry is to be used as a running head (1) or is not to be used as a running head (0) [I] <0>

Supplementing running heads

VLK n Character strings which (depending on the type of entry) are to be inserted before text part n in the running head. [II]

NLK n Character strings which (depending on the type of entry) are to be inserted after text part n in the running head. [II]

Replacing character strings in running heads

XLK n Replacing character strings in text part n for the running head. [x]

LLK n Character strings which (depending on the type of entry) are to be used as a replacement for an empty text part n in the running head. [II] <>

Headings when the initial letter changes

AB n If there is a change in the initial letter of text part n, it is to be written to the output as a single uppercase letter (1) or is not to be written to the output (0). [I] <0>

SAB n Character strings located at the beginning of text part n which are to be skipped when a search is carried out for the initial letter. [xI]

If format control characters are present at the beginning of a text part, for example "#k+" (= switches to small caps), these must be specified with parameter SAB. Otherwise, the "k" in "#k+" would be interpreted as an initial letter.

ABD Character strings which serve (or are to be defined) as initial letters. [xI]

If parameter ABD is specified, the following parameter ABE must also be specified.

ABE Text parts (parallel to ABD) which are to be included in the output instead of initial letters. [II]

VAB n Character strings which (depending on the type of entry) are to appear before a new initial letter. [II]

NAB n Character strings which (depending on the type of entry) are to appear after a new initial letter. [II]

Limiting record length The following parameters only affect the output to the DESTINATION file (but not to the LISTING file).

SL Output record length [I]

In case this parameter is not specified, a single line (index entry with references) will not be split (however, see parameters NZB and NZ).

Two numerical values can be specified:

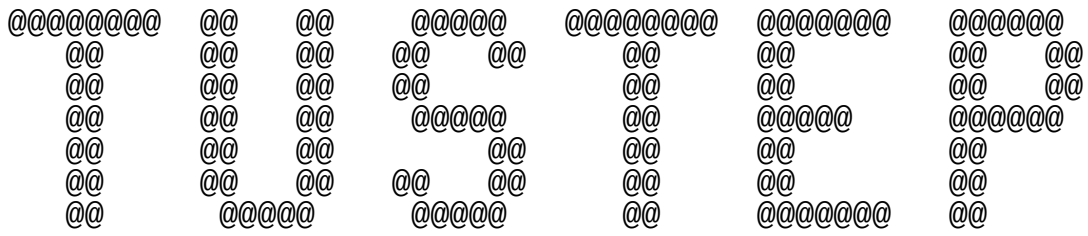
1st number: Maximum length of records to be split because they are longer than the second numerical value of this parameter. <99999>

2nd number: Maximum length for records not to be split. werden sollen. <100 or value of 1st number if this is greater than 100>

Before output, a line having more characters than specified with the second number will be subdivided into output records. The line will be subdivided into

records having a maximum length specified by this parameter's first number. The line is divided at any blank which is not preceded by a "-" (exception: a blank preceded by " -", since this "-" represents a dash and can thus not be confused with a hyphen). If such a division point cannot be found within the number of characters specified by this parameter's first number, the next possible division point will be selected.

Alphabetical list of parameters



Tübingen

System of Text Processing Programs

Program

G L I S T I N G

1993

TÜBINGEN UNIVERSITY - CENTER FOR DATA PROCESSING

Department of Literary and Documentary Data Processing

Brunnenstrasse 27, D-72074 Tübingen

Survey:

Command:

#GLISTING

Specifications:

SOURCE	= file		Name of the file containing the data to be listed (i.e. of which a protocol is to be generated) or (for MODE=A and MODE=U) name of the file containing the protocol of which certain parts are to be copied.
	= -STD-		The standard TEXT file contains the data of which a protocol is to be prepared, or contains the protocol of which parts are to be copied.
MODE	= T		Insert page-line number in front of each record (for data numbered in text mode).
	= -STD-		Begin a new page when the page number of the record changes; insert the line number in front of each record.
	= S		Begin a new page when the page number of the record changes; no page-line number before records.
	= O		No record numbers
	= P		Insert line number in front of each record (for data numbered in program mode, and for segment files).
	= A		The SOURCE file is a listing file; selection by the page number part of the record number.
	= U		The SOURCE file is a listing file; select pages by the page number in the header line.
	= ...		Type of printer for which the data are to be prepared. The type of available printers depends on the computer being used. For a list of printers, use the command #LIST,PRINTERS. The printer can also be specified by parameters.
ERASE	= -	*	If the LISTING file already contains data, they are to be retained.
	= +		If the LISTING file already contains data, they are to be erased beforehand.
PARAMETER	= file		Name of the file containing parameters

	= *		The parameters follow the command and are ended by *EOF.
LISTING	= -STD-	*	The generated listing is to be written into the standard LISTING file.
	= +		The generated listing is to be written to the journal (i.e. the screen, when in interactive mode).
	= file		Name of the file into which the generated listing is to be written.

Features:

With this command, a listing of a file can be generated, i.e. a protocol of its data can be prepared. Control characters included in the data are not intepreted but treated as printable characters. To determine the form of the listing, parameters are employed to specify, for example, the number of columns, headers for pages and columns, type of numbering and line spacing.

Furthermore, specific pages of a listing file given in the specification SOURCE can be selected and copied to the file given for the specification LISTING.

Note

Data in the SOURCE file are never altered with this program. The program's results (the data prepared for printing) are written to the LISTING file, which can subsequently be sent to a printer and printed out with the command #PRINT.

Modes

Selecting the mode selection of the appropriate mode is determined according to the data in the SOURCE file and the type of record numbering employed:

Data	Numbering	Modes
Texts	Text mode	T, O, -STD-, S
Texts	Program mode	P, O
Programs	Program mode	P, O
Makros	Program mode	P, O
Protocols	Text mode	A, U

If a printer is given for the specification MODE instead of the desired mode, the mode will be selected automatically according to the following criteria: Mode P will be selected if the file's records are numbered in program mode (this will be assumed if the record number of the last record is less than 1000000) or if the file is a segment file. Mode T will be used in all other cases.

MODE = T

Records must be numbered in text mode. The page-line number and the distinction number (if this is not zero) will appear in front of every new record (but not in front of continuation lines). The first 15 positions at the left margin are used for this purpose. A running number of the pages prepared for printing will be inserted in the page header. The record numbers do not influence actual page breaks and the page number inserted in the header.

This mode is used for files numbered in text mode which have not yet been prepared for printing, and when none of the following modes are more appropriate.

MODE = -STD-

Records must be numbered in text mode. The line number and the distinction number (if this is not zero) will be printed before each new record (but not before continuation lines). The first 12 positions of the left margin are reserved for this. The page number of the record will be inserted in the page header. A new page will begin each time the page number (part of the record number) changes.

This mode should not be used when the records are numbered in such a way that only few consecutive records in the file have the same page number.

MODE = S

The only difference between this mode and mode= -STD- is that here no line numbers will be printed. The text starts directly at the left margin.

This mode can be used when each page (i.e. records with the same page number) consists of a text which is to be printed on its own separate page, and when record line numbering is of no significance.

MODE = O

In this mode, the record number of the records are not taken into account and will not be printed. Text starts directly at the left margin. A running number of the pages prepared for printing will be inserted in the page header. The record numbers do not influence page breaks and the number inserted in the header.

MODE = P

This mode interprets record numbers in program mode (in all other modes the records numbers are interpreted in text mode). The line number and distinction number (if not zero) will be printed in front of each record. The first 15 positions of the left margin are reserved for this. A running page number of the pages prepared for printing will be inserted in the page header. A change in the page number (part of the record number) results in the start of a new page; (for a segment file, the start of a new segment).

This mode is used to print program and segment files. It is not used for text files.

MODE = A

In this mode, the SOURCE file should be a file that contains data already prepared for printing. This is generally a file which has been written as a LISTING file in a previously-run TUSTEP program. The essential characteristic of such a file is that the first character of every record is a line-feed control character (e.g. "-" for a new page, "1", "2" to "7" for a line feed of 1, 2 to 7 lines) or another control character for the command #PRINT. Such a file can be copied to another LISTING file, where certain parts of the SOURCE file can be selected in the process using parameters. If selection is carried out by page number, please note that the page number of the record number is used for selection, not that of the page number which may appear in the printed header. To select pages corresponding to the printed header page number, MODE U must be used instead.

MODE = U

As in mode A, the SOURCE file should contain data that have already beem prepared for printing. In contrast to mode A, however, selection of page numbers with the parameter SKN is not based on the page number of the record number but on the *first* page number contained in the header line (first line of a page, line-feed control character "-"). To determine which record number is to be used for the page number in the header line, see parameter SNR below. Parameter SKN is obligatory in mode U.

Parameters

Values in [ ] refer to the type of parameter employed. The various types of parameters are described in the "Parameters" chapter of "TUSTEP Basics".

Values in < > refer to default settings.

Selecting data

If the entire file is to be processed, none of the following parameters are necessary.

BER Definition of a single area ("page.line-page.line") or a starting point ("page.line"). This parameter is only used when not processing the entire input file. [xI]

If a segment of a segment file is to be processed, the name of the segment can be substituted for the area.

This parameter can only be used when the record numbers of the file are in ascending order; the parameters BER, SKN and DAE are mutually exclusive. When MODE=P, this parameter can only be used to specify a segment name. It cannot be used when MODE=A and MODE=U (Tip: select data with SKN or DAE instead).

SKN Specifies the page numbers (when MODUS=P line numbers) of the records to be printed. When a page number (line number) is specified, all records having this page number (line number) will be printed. If a number of successive pages (lines) are to be printed, an area definition ("page-page" or "line-line") can be made. In addition, more than one number (or pair of numbers), each separated by an apostrophe, is possible here. However, the page numbers (line numbers) must be given in ascending order.

This parameter can only be used when the record numbers of the file are in ascending order; the parameters BER, SKN and DAE are mutually exclusive. This parameter is obligatory for MODE=U. [xI]

DAE Number of records to be printed if only the beginning and/or end of a file is to be printed. [I]

Two numerical values may be specified here:

1st number: beginning of file <0>

Number of records starting from the beginning of the file which are to be printed

2nd number: end of file <0>

Number of records starting at the end of the file which are to be printed

The parameters BER, SKN and DAE are mutually exclusive. This parameter cannot be used when MODE=U (Tip: use MODUS=A instead).

SNR Character strings used to denote the page number. [IX]

This parameter can only be used when MODE=U

In MODE=U, the selection of pages to be copied is carried out using the page number. This must be located in the first line of every page (the line with the line-feed control character "-"). The number which first occurs after the character string specified in this parameter will be interpreted as the page number. If this parameter is not present, the number which follows the character string "page" or, if the character string "page" does not occur, the first number in the line will be interpreted as the page number.

Parameters for LISTING file settings

For MODE=A and MODE=U, the data must already be prepared for printing. Therefore, for these two modes only the parameters listed under "Selecting data" are allowed.

DR Printer output control [I]

Five numerical values can be specified:

1st number: columns <1>

Number of columns to be printed side-by-side on every page.

2nd number: Left margin <0>

Number of blanks to the left of the first column

3rd number: Width <132>

Number of characters per column

4th number: Space between columns <0>

Number of blanks between columns

5th number: Indentation for continuation lines <0>

Number of blanks at the start of continuation lines (for Modes T, P and -STD-: in addition to the 15 or 12 positions reserved for numbers in the left margin)

DRZ Additional specifications for printer output control [I]

Seven numerical values can be specified:

1st number: Header <3>

Number of lines for the header (including blank lines)

2nd number: Column height <60>

Number of lines per column (excluding lines for the header and footer)

3rd number: Footer <0>

Number of lines for the footer (including blank lines)

4th number: Columns <1> Number of columns per page

5th number: Repetition of footer text

Number of lines of the page's footer text (counting from the first line down) which are to be repeated after every column (except for the last column).

6th number: Blanks between columns <0>

Number of blanks between each column

7th number: Repetition of the header text <0>

Number of lines of the page's header text (counting from the bottom line to the top) which are to be repeated before every column (except for the first column).

KT Text parts to be printed at the top of every page as a header [II]
<"file name" xx. xxx. xxxx xx.xx xxxxxx>

To insert the current date, enter "xx. xxx. xxxx" or "xx.xx.xx" at the appropriate position. Positions for the current time may be indicated by "xx.xx" and for

the page number by "xxxxxx" (2 to 6 "x"s, but at least as many positions as necessary for the page number). If "- xxxxxx -" is entered for the page number, it will appeared centered on the page between two minus signs, with each minus sign separated from the page number by a space having the width of up to one whole blank. However, the date, time and page number can be inserted only one time each.

If a character string begins with a "*:", the rest of the character string serves as a header for every text column. If a numeral n is entered in place of the asterisk, the rest of the character string is used as a header for the nth column. If the numeral given equals 0, the rest of the character string is used for the entire line. If a character string does not begin as just described, "0:" is assumed (standard value).

The following rules determine which line of the header is used by the character string: A character string which is designated for an entire line will be printed at the start of a new line (starting with the first line). A character string which is designated for a particular column will be printed in the same line as the preceding character string, unless this line contains text meant for an entire line, for the same column, or for a column further to the right. In this case, the character string will be printed in the next line.

Each of the specified character strings can be arranged in three parts using the formating instructions "@z" and "@/":

left-aligned @z centered @/ right-aligned

The individual parts will be inserted left-aligned, centered and right-aligned. An individual part may be omitted; in this case the formatting instructions in front of the second and third parts may also be omitted.

FT Text parts (analogous to parameter KT) to be printed as a footer at the bottom of every page. [II]

The date, time and page number may not be inserted in the footer if they have already been included in the header.

LZ Number of blank lines between printed lines. [I]

Two numerical values can be specified here:

1st number: Number of blank lines before every line in which a new record begins. <0>

2nd number: Number of blank lines before every continuation line. <0>

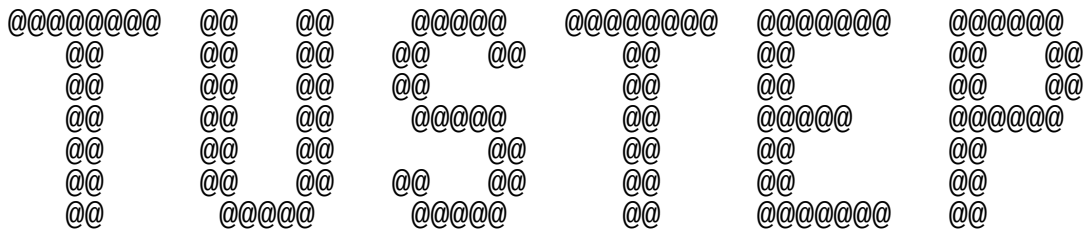
DRT Printing device for which the data are to be prepared. [xi]

This parameter cannot be specified for MODE=A and MODE=U. Otherwise, this parameter is obligatory when no printer has been given for the specification MODE (and may only be specified in this case).

The types of available printers depends on the actual computer being used. To obtain a list of these, use the command
#LIST, PRINTERS.

Alphabetical list of parameters

* * * * *



Tübingen

System of Text Processing Programs

Program

I N S E R T

1993

TÜBINGEN UNIVERSITY - CENTER FOR DATA PROCESSING

Department of Literary and Documentary Data Processing

Brunnenstrasse 27, D-72074 Tübingen

Survey:

Command:

#INSERT

Specifications:

SOURCE	= file		Name of the file containing the data into which text parts are to be inserted
	= -STD-		The data into which text parts are to be inserted are located in the standard TEXT file.
DESTINATION	= file		Name of the file to which the data with the inserted text parts are to be written
	= -STD-		The data with the inserted text parts are to be written to the standard TEXT file.
MODE	= -STD-	*	Unique short forms, normal case
	= SHORT		Unique short forms, short text parts to be inserted
	= LONG		Unique short forms, long text parts to be inserted
	= PARALLEL		Short forms occur in the SHORTFORMS file and in the SOURCE file in parallel fashion.
	= SORTED		Short forms are sorted alphabetically.
	= MORE		The SHORTFORMS file contains more short forms than the SOURCE file.
	= LESS		The SHORTFORMS file contains less short forms than the SOURCE file.
	= GROUPS		The SHORTFORMS file contain groups, each having the same short forms. A new copy of the SOURCE file is to be made for each group.
ERASE	= -	*	If the DESTINATION file already contains data, they are to be retained.
	= +		If the DESTINATION file already contains data, they are to be erased beforehand.
PARAMETERS	= file		Name of the file containing parameters
	= *		The parameters follow the command and are ended by *EOF.
SHORTFORMS	= file		Name of the file containing the text parts to be inserted

Features:

With this program, text parts which are located in a file and are indentifiable by a short form (e.g. a running number) can be inserted into the data of another file. Each text part is inserted at the position in the data where the corresponding short form is located.

Examples of possible applications (the type of data located in the SHORTFORMS file is given in parentheses):

- Merge footnotes into the text (footnotes)
- Replace short forms by full text (full text)
- Compile letters from text blocks (standard text)
- Generate form letters (one address per group)

Data requirements

The data into which the text parts are to be inserted must be contained in the SOURCE file; the text parts which are to be inserted must be contained in the SHORTFORMS file. The results will be written to the DESTINATION file.

SHORTFORMS file: Each text part to be inserted must be preceded by a short form by which the text part can be identified. These short forms must be unambigiously marked by a beginning marker before them and a end marker after them. The beginning marker must be specified with parameter AKK, the end marker with parameter EKK. If no data follow the short forms in the same line, the end marker may be omitted, and parameter EKK need not be specified. Markers for the text parts themselves are not necessary. Each text part begins after the end marker for the short form and ends before the beginning marker of the short form for the next text part. A text part may take up a number of lines. Line breaks within a text part will be kept when the text part is inserted. If the short forms are arranged in groups, where each group has the same short forms but different text parts for identical short forms, the start of each group must be unambigiously marked by a beginning marker as specified in parameter NKG.

SOURCE file: A short form must be located in the data at every position where the corresponding text part is to be inserted. This short form must also be unambigiously marked by a beginning marker in front of it and an end marker after it. The beginning marker must be specified with parameter AKD, the end marker with parameter EKD. If no data follow the short forms in the same line, the end marker may be omitted, and parameter EKD need not be specified. Parameter KKZ can be used to determine whether the markers for the short form and the short form itself should be kept or eliminated. If necessary, an additional character string can be added both before and after an inserted text part. These character strings can be specified with parameter NAE.

Both the SOURCE file as well as the SHORTFORMS file may contain more than one short form (each with a beginning and end marker) per line. However, related beginning and end markers must be kept together in the same line.

Selecting the mode

If the short forms in the SHORTFORMS file are arranged in groups, and a new copy of the SOURCE file is to be written to the DESTINATION file for each group, select the following mode:

- MODE = GROUPS

  In this mode, a short form may occur in the SOURCE file any number of times. However, in the SHORTFORMS file a short form may appear only once in each group. Short forms may be given in any order. However, the text parts which correspond to the short forms given in the SOURCE file must be small enough in size to be copied to the computer's memory.

If the short forms in the SOURCE and SHORTFORMS files have been given in the same order, one of the following modes can be selected:

- MODE = PARALLEL

  To be used when each short form in the SOURCE file has a corresponding form in the SHORTFORMS file, and vice versa. (for exceptions, see: "Program operation modes").

- MODE = SORTED

  To be used when the short forms in the SOURCE and SHORTFORMS files are listed in alphabetical order.

- MODE = MORE

  To be used when the SHORTFORMS file has more short forms than the SOURCE file, provided that all short forms occurring in the SOURCE file also occur in the SHORTFORMS file.

- MODE = LESS

  To be used when the SHORTFORMS file has less short forms than the SOURCE file, provided that all short forms occurring in the SHORTFORMS file also occur in the SOURCE file.

When the order of short forms in the SOURCE file differs from that in the SHORTFORMS file, only one of the following modes may be selected. In this case, each short form may occur any number of times in the SOURCE file, but only once in the SHORTFORMS file. Short forms may appear in any order.

- MODE = -STD-

  To be used when the SHORTFORMS file is small enough to fit into memory.

- MODE = SHORT

  To be used when there is not enough memory for MODE = -STD- and when the text parts in the SHORTFORMS file which correspond to the short forms in the SOURCE file are short enough to be copied to memory.

- MODE = LONG

  when there is not enough memory for MODE=SHORT.

  If working in MODE=LONG also proves to exceed memory capacity, either sort both the SOURCE and SHORTFORMS files so that the short forms are arranged in alphabetical order and switch to MODE=SORTED, or run the program a number of times in MODE=LONG, while selecting with each run a different area of the SOURCE file using parameter BER.

Program operation mode

In all modes, short forms which occur in the SOURCE file but which cannot be found in the SHORTFORMS file will be transferred to the DESTINATION file unaltered. If in this case an error message is desired, this can be specified in parameter UND.

When short forms from the SOURCE and SHORTFORMS files are compared, no distinction is made between uppercase and lowercase letters.

For modes PARALLEL, SORTED, MORE and LESS, the following applies:

The program reads the data from the SOURCE file and writes them to the DESTINATION file while searching the data for short forms. When a short form is found, it is then compared with the next short form in the SHORTFORMS file. If the two short forms match, the corresponding text part will be retrieved from the SHORTFORMS file and inserted in the data. Otherwise the short form found in the SOURCE file will be compared with the short form in the SHORTFORMS file whose text part was last inserted. If these two short forms match, the corresponding text part will be retrieved from the SHORTFORMS file and inserted in the data. If the two do not match, further processing will be determined by the current mode in effect:

MODE = PARALLEL

If no matching short form can be found during the procedure outlined above, the program will abort.

MODE = SORTED

If no matching short form can be found during the procedure outlined above, the search in the SHORTFORMS file will continue until the short form is found or until a short form is found which would follow the searched short form in alphabetical order. In the latter case, the short form is regarded as not found; the search will then continue for the next short form in the SOURCE file.

MODE = MORE

If no matching short form can be found during the procedure outlined above, the search in the SHORTFORMS file will be continued until the short form has been found. The short forms in the SHORTFORMS file that have been passed over in the process will no longer be taken into account. Thus, when a short form has not been found when the end of the SHORTFORMS file has been reached, the subsequent short forms in the SOURCE file will also not be found.

MODE = LESS

If no matching short form can be found during the procedure outlined above, the SOURCE file will be searched until the short form has been found. Any short forms thus passed over in the SOURCE file will remain unaltered.

MODE = -STD-

Upon reading the short forms and their corresponding text parts from the SHORTFORMS file, the program creates an internal table.

After the table has been created, data are read from the SOURCE file and are written to the DESTINATION file. In the process, data are searched for short forms. When a short form is found, the corresponding text part is retrieved from the table and inserted into the data.

MODE = SHORT

The program first reads the SOURCE file and compiles an internal table consisting of the short forms found. The SHORTFORMS file is then searched for these short forms and the corresponding text parts are added to the table.

After the table has been compiled, the program reads the SOURCE file data and writes them to the DESTINATION file, while searching the data for short forms. If a short form is found, the corresponding text part is retrieved from the table and inserted into the data.

This mode has one disadvantage compared to Mode = -STD- in that the SOURCE file must be read twice instead of just once, a process which consumes the corresponding amount of computer time.

MODE = LONG

The program first reads the SOURCE file and compiles a table consisting of the short forms found. The SHORTFORMS file is then searched for these short forms and a pointer is added to the table. This pointer refers to the position in the SHORTFORMS file where the short form is located.

After the table has been compiled, the program reads data from the SOURCE file and writes them to the DESTINATION file, while searching the data for short forms. If a short form is found, the pointer is used to retrieve the corresponding text part from the SHORTFORMS file and insert it into the data.

This mode has one disadvantage compared to MODE = SHORT in that the SHORTFORMS file must be read in random access fashion instead of sequentially, thus increasing the amount of time required for the process to a considerable degree.

MODE = GROUPS

The program first reads the SOURCE file and compiles a table consisting of the short forms found. Then the first group in the SHORTFORMS file is searched for these short forms and the corresponding text parts are added to the table.

After the table has been compiled, the program reads data from the SOURCE file and writes them to the DESTINATION file, while searching the data for short forms. If a short form is found, the corresponding text part is retrieved from the table and inserted into the data.

After all SOURCE file data have been processed, the text parts in the table are erased and the next group in the SHORTFORMS file is searched for the short forms listed in the table, with the corresponding text parts being added to the table. The SOURCE file is then reprocessed and the entire procedure repeated for each group in the SHORTFORMS file.

Parameters

Values in [ ] refer to the type of parameter employed. The various types of parameters are described in the "Parameters" chapter of "TUSTEP Basics".

Values in < > refer to default settings.

In addition to the following parameters, other parameters can be used to define character groups and character strings. [v]

Selecting data in the SOURCE file

If the entire file is to be processed, none of the following parameters are necessary.

BER Definition of a *single* area ("page.line-page.line") or a starting point ("page.line"). This parameter is only used when not processing the entire file. [xi]

If a segment of a segment file is to be processed, the name of the segment can be substituted for the area.

This parameter can only be used when the record numbers of the file are in ascending order.

MAX For test runs, this specifies the maximum number of input records that are to be processed. [i] <999999>

Length of short forms

MKL Maximum length of a short form (excluding beginning and end markers). If a short form is longer than specified here, it will be ignored. [i] <8>

Short form markers in the SOURCE file

Of the two following parameters, parameter AKD must always be specified. Parameter EKD is only necessary when other data follow the short form in the same line.

AKD Character strings which mark the beginning of a short form. [ix]

EKD Character strings which mark the end of a short form. [IX]

Each short form starts after the beginning marker and ends before the end marker, or at the end of the line. If a character string thus marked as a short form contains any leading or trailing blanks, these blanks will be ignored.

Each search for the beginning marker of the next short form starts after the end marker of a short form. If the end marker of a short form cannot be found in the same line, the search for the next beginning marker will start at the beginning of the next line.

Short form markers in the SHORTFORMS file

Of the following two parameters, parameter AKK must alway be specified. Parameter EKK is only necessary when other data follow the short form in the same line.

AKK Character string which marks the beginning of a short form. [IX]

EKK Character string which marks the end of a short form. [IX]

Each short form starts after the beginning marker and ends before the end marker, or at the end of the line. If a character string thus marked as a short form contains any leading or trailing blanks, these blanks will be ignored.

Each search for the beginning marker of the next short form starts after the end marker of a short form. If the end marker of a short form cannot be found in the same line, the search for the next beginning marker will start at the beginning of the next line.

Group markers in the SHORTFORMS file

This parameter is obligatory when MODE=GROUPS and may only be used in this mode. It is used to specify how the beginning of a group of short forms has been marked. The marker must be located in the SHORTFORMS file at the start of each record with which a new group begins. This marker can at the same time also be the character string which marks the beginning of a short form (and thus specified with parameter AKK).

NKG Character string placed at the beginning of a record which marks the start of a new group of short forms. [IX]

Text part markers in the DESTINATION file

KKZ Additional specifications as to whether the short forms and their markers in the SOURCE file should be retained in the DESTINATION file. [I] <0,0,0>

Three numerical values can be given here:

1st number: beginning marker of the short form

0 = is not to be retained
1 = is to be retained

2nd number: short form

0 = is not to be retained
1 = is to be retained

3rd number: the end marker of the short form

0 = is not to be retained
1 = is to be retained; the text part is to be inserted after it.
2 = is to be retained; the text part is to be inserted before it.

NAE Text parts (new beginning and end markers) which are to be added to the beginning or end of the inserted text part. [II]

Two text parts can be specified: the first will be added directly in front of the inserted text part, the second directly after it.

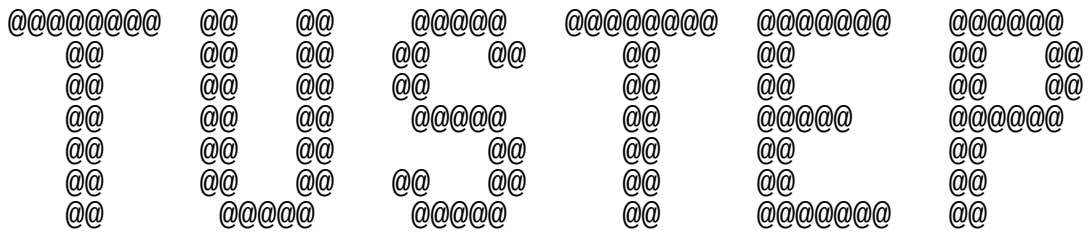
Error log

UND Specifies whether SOURCE file short forms not found in the SHORTFORMS file should be recorded in the journal. [I] <0>

0 = do not record
1 = record

Alphabetical list of parameters

* * * * *



Tübingen

System of Text Processing Programs

Program

N U M B E R

1993

TÜBINGEN UNIVERSITY - CENTER FOR DATA PROCESSING

Department of Literary and Documentary Data Processing

Brunnenstrasse 27, D-72074 Tübingen

Survey:

Command:

#NUMBER

Specifications:

SOURCE	= file		Name of the file containing the data in which numbers and/or references are to be updated
	= -STD-		The standard TEXT file contains the data in which numbers and/or references are to be updated
DESTINATION	= file		Name of the file to which the data with the updated nubmers and/or references are to be written
	= -STD-		The data with the updated numbers and/or references are to be written to the standard TEXT file.
MODE	= -	*	Normal case
	= PUT		The concordance of the old and the new numbering is to be written to the CONCORDANCE file. Any data contained in this file will be erased regardless of what has been given for the specification ERASE.
	= GET		The concordance of the old and new numbering is to be read from the CONCORDANCE file.
	= ADD		The concordance of the old and the new numbering in the CONCORDANCE file is to be supplemented.
ERASE	= -	*	If the DESTINATION file or the LISTING file already contains data, they are to be retained.
	= +		If the DESTINATION file or the LISTING file already contains data, they are to be erased beforehand.
PARAMETER	= file		Name of the file containing parameters
	= *		The parameters follow the command and are ended by *EOF.
CONCORDANCE	= -	*	Normal case
	= file		Name of the file to which the concordance of the old and the new numbering is to be written or which contains the concordance.

	= -STD-		The concordance of the old and the new numbering is to be written to the standard DATA file or is to be read from the standard DATA file.
LISTING	= -		No protocol of the concordance of the old and the new numbering
	= +		A protocol of the concordance of the old and the new numbering is to be written to the journal.
	= -STD-	*	A protocol of the concordance of the old and the new numbering is to be written to the standard LISTING file.
	= file		Name of the file to which a protocol of the concordance of the old and the new numbering is to be written.

Features:

This command is used to update (running) numbers and the respective references contained in a text. For this purpose, running numbers can be inserted at positions marked appropriately. (Old) numbers already present at these positions will be replaced. Appropiately marked references which refer to old numbers can also be be updated.

In addition, this command can be used to update references to a page-line-number after the page-line-division has been changed.

Modes

Running numbers and references are usually located in the same file. In this case, the specification MODE does not have to be given (i.e. equals MODE=-).

If the running numbers and the references are located in different files, the file containing the running numbers must be processed first. This step provides the information necessary for updating the references, i.e. which old reference number is to be replaced by which new reference number, or on which page and line the reference points (= running numbers) are to be located. This information (= concordance) must be saved in order to be used when the program is called up a second time for processing the file containing the references. For this purpose, MODE=PUT is to be used when calling up the program for the first time. The file given in the specification CONCORDANCE will be used for saving the concordance information.
When the program is called up a second time for the purpose of updating the references, MODE=GET is to be specified in order to instruct the program to evaluate the saved concordance information. The file given in the specification CONCORDANCE must be the same given when the progam was called up for the first time (with MODE=PUT).

If the data with the running numbers are located in more than one file, each file must first be processed individually (using the parameter LNR, without the parameter VNR; parameters SK and ZK are necessary for handling any page-line numbers). The first file must be processed in MODE=PUT and all other files in MODE=ADD. In MODE=ADD, the concordance is first read (as in MODE=GET) and then written back in its supplemented form (as in MODE=PUT). Finally, the files containing the references are processed in MODE=ADD (using the parameter VNR, without the parameter LNR; parameters SK and ZK are necessary for inserting any page-line numbers). For this procedure, the files may contain running numbers and references.

Parameters

Values in [ ] refer to the type of parameter employed. The various types of parameters are described in the "Parameters" chapter of "TUSTEP Basics".

Values in < > refer to initial settings.

In addition to the following parameters, other parameters can be used to define character groups and character strings. [v]

Selecting data

If the entire file is to be processed, none of the following parameters are necessary.

BER Definition of a single area ("page.line-page.line") or a starting point ("page.line"). This parameter is only used when not processing the entire file. [xi]

If a segment of a segment file is to be processed, the name of the segment can be substituted for the area.

This parameter can only be used when the record numbers of the file are in ascending order.

MAX For test runs, this specifies how many imput records are to be processed. [i] <999999>

Inserting a running number

LNR Character string after which the running number is to be inserted. If a number is already located directly after such a character string, it will be replaced. Unless specfied otherwise in the parameter ART, the running number will be increased by 1 each time a character string specified in parameter LNR is encountered. [ix]

For the DOS version, the greatest number allowed is 15 999, in all other versions: 99 999.

LNB Numerical value to be added to the running number. [i] <0>

If the new running number is to begin with a number other than 1, this parameter can be used to specify the desired number. The first new running number will

thus be one number larger than the value specified here.

ART Specifies how the running number is to be incremented. [I] <0>

0 = Each time a character string specified in the parameter LNR is encountered, the running number is to be increased by 1.

1 = The running number is to be increased by 1 only if there is still no number in the input data located directly after the character string behind which it is to be inserted, or if the number at this position in the input data occurs for the first time. If this number has already been encountered in the input data, it will be replaced by the same number used at its first occurrence.

Updating the reference numbers

VNR Character string after which the reference number is to be replaced. This reference number may not be omitted. In addition, it must be used exactly once as a running number located after a character string specified in parameter LNR (Exception: when parameter ART = 1). The reference number will be replaced by the same number as its corresponding running number. [IX]

If the corresponding running number occurs more than once, the reference will be altered according to the first occurrence of the running number. Should the corresponding running number be absent, the reference number will be replaced by 0.

References to page and line numbers

By using the parameter SK, the program is instructed to insert references to page numbers. If the parameter ZK is also specified, line numbers will also be inserted.

In this case, running numbers merely serve to mark a reference position and are not altered (the parameter ART is not allowed here). The reference numbers also remain unaltered and merely serve to specify to which position (i.e., the position of the "running number" having the same number) the reference is to be made.

The page number of the page to which the reference refers is inserted in the text directly after the character string specified in parameter SK. This character string must occur in the text after the reference number and at the very latest in the following line. If this character string is directly

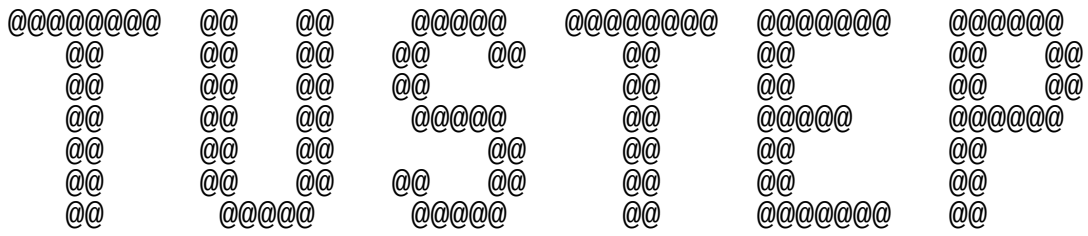
followed by a number, the number will be replaced. If parameter ZK has also been specified, the line number of the line to which the reference refers will be inserted in the text after the character string specified in this parameter. This character string must occur in the text after the character string for the page number. Any number already located there will be replaced.

SK Character string which marks the position in the text where the page number is to be inserted. [IX]

ZK Character string which marks the position in the text where the line number is to be inserted. [IX]

Alphabetical list of parameters

* * * * *



Tübingen

System of Text Processing Programs

Program

P R E P A R E I N D E X

1993

TÜBINGEN UNIVERSITY - CENTER FOR DATA PROCESSING

Department of Literary and Documentary Data Processing

Brunnenstrasse 27, D-72074 Tübingen

Survey:

Command:

#PINDEX

Specifications:

SOURCE	= file		Name of the file containing the data from which index entries are to be extracted or from which a KWIC index is to be prepared
	= -STD-		The standard TEXT file contains the data from which index entries are to be extracted or from which a KWIC index is is to be prepared
DESTINATION	= file		Name of the file to which the index entries or the entries for the KWIC index are to be written; more than one file name is allowed
	= -STD-		The index entries or the entries for the KWIC index are to be written to the standard TEXT file.
MODE	= +	*	Prepare index entries with references
	= -		Prepare index entries without references
	= KWIC		Generate entries for KWIC index
ERASE	=	*	If the DESTINATION file, the DATA file or the LISTING file already contains data, their data are to be retained.
	= +		If the DESTINATION file, the DATA file or the LISTING file already contains data, their data are to be erased beforehand
PARAMETER	= file		Name of the file containing parameters.
	= *		Parameters follow the command and are ended by *EOF.
DATA	= -	*	The data are to be written to the DESTINATION file in their entirety.
	= file		Name of the file to which the text part (i.e. data not needed for sorting) of each record is to be written
	= -STD-		The text part (i.e. data not needed for sorting) of each record is to be written to the standard DATA file.
LISTING	= -	*	No trace listing
	= +		Trace listing is to be written into the journal.

= -STD-	Trace listing is to be written into the standard LISTING file.
= file	Name of the file into which the trace listing is to be written.

Features:

This program can be used to: appropriate composed into entities

- break down text into sortable entries appropriate for an index (e.g. for an index of word-forms) or a KWIC index ;
- extract appropriately marked text parts from texts as index entries (e.g. for an index of authors or a subject index).

General remarks

Index entries are created by decomposing the input text by means of separator character strings and/or by selecting text parts located between beginning and end markers.

More than one text part can be used to compose an index entry. To accomplish this, additional character strings taken from the input text may be included in an index entry.

One to three sort keys can be created from the text of an index entry. This is necessary because the order of characters in the the computer's internal code does not correspond to the usual alphabetical order. The sort key is used only for sorting purposes and is eliminated thereafter. Only one sort key is necessary for texts containing no umlauts or diacritical marks. If the text contains umlauts, the usual alphabetical order (as defined by DIN 5007) is preserved by encoding ä, ö, ü and ß as ae, oe, ue and ss for the first sort key. A second key is necessary, however, if words such as "Maße" and "Masse" or names such as "Jäger" and "Jaeger" are to be kept separate from each other. For this sorting key, ä, ö, ü and ß can be encoded as az, oz, uz und sz. If a text contains diacritical marks, a second sort key is also necessary. Normally, all diacritical marks are eliminated for the first sort key in order that sorting can be first carried out in alphabetical order; accents and other diacritical marks only affect the order of two identical characters. For the second sort key, each accent may be encoded as a "z" followed by a decimal number, for example, so that words which differ only by a diacritical mark may be arranged in the desired order. In addition, if uppercase and lowercase letters are decisive in the sorting process, a third key is required. If, however, umlauts and diacritical marks do not occur in the text, sorting of uppercase and lowercase letters can be carried out in the second sort key.

If the index entries are already organized by group in the input file, it is possible to sort the index entries within each of the respective groups only. This requires that the beginning of each group is marked in a unique way, such as using a marker before each section title. Each group is numbered internally and the corresponding number placed directly in front of the sort key of the individual entries. This keeps the groups in their original, unchanged order.

Index entries can be assigned to various types which are later used to differtiate entries in a variety of ways (e.g. typographically) when the index is actually being prepared. To accomplish this, each entry is assigned a type identification number whose value is based on the type identification character taken from the input text. This type identification character can be either placed before each index entry in the text, or it can be specified as a general type identification. In the latter case, it affects all subsequent index entries.

In case the index entries listed in the index are also to be supplied with a reference to their original position in the basic text, each index entry must be provided with such a reference. If reference is to be made to the page number (as

well as line and word number) in the text from which the index is to be prepared, the reference can be taken from the record number of the input text. Another possibility for supplying a reference is to use text parts marked as such in the input text. In this way, a reference may consist of a number of text parts.

The sort program (command: #SORT) requires a certain amount of memory for temporary storage of the data to be sorted. Large amounts of data may exceed available memory. This can usually be avoided by outputting the sort key and the actual index entries to separate files (DESTINATION file and DATA file, respectively). Of these two files, only the DESTINATION file containing the sort keys needs to be sorted. If the amount of available memory is still too small for sorting, the sort key can be distributed among a number of files. These must be sorted individually and then merged into one file. To distribute the sort key to more than one file, merely specifying a number of DESTINATION files is sufficient; no additional parameters are necessary for this purpose.

Parameters

Values in [ ] refer to the type of parameter employed. The various types of parameters are described in the "Parameters" chapter of "TUSTEP Basics".

In certain parameters, text parts can be selected for further processing by means of beginning and/or end markers as well as markers that serve as left and right parentheses for selecting the desired part. The manner in which these parameters function is also described at the end of the section entitled "Parameters" in the chapter "TUSTEP Basics".

In addition to the parameters described below, parameters for defining string groups and character groups may also be employed. [v]

In addition to the parameter SSL, at least the parameter SWT must be specified when MODE=KWIC; in all other modes, one of the parameters EA, EE or TR must be specified.

Selecting data

If the entire file is to be processed, none of the following parameters are necessary.

BER Definition of an area ("page.line-page.line") or a starting point ("page.line"). This parameter is only used when not processing the entire file. [xI]

If a segment of a segment file is to be processed, the name of the segment can be substituted for the area.

This parameter can only be used when the record numbers of the file are in ascending order.

MAX For test runs, this specifies the maximum number of text units (= input records, if none of the parameters ANR, AA or AE has been given) that should be used for creating index entries and/or the maximum number of index entries to be created. [I]

Two numerical values can be specified:

1st number Max. number of text units to be read <999999>
2nd number Max. number of index entries to be generated <999999>

Organizing records into text units

In case each index entry and, if applicable, each reference to the pertinent text is contained entirely in one input record, none of the following parameters should be used. In this case, each input record comprises a text unit. If this is not the case, the parameters AA and/or AE can be used to organize more than one record into a single text unit.

If one of the following four parameters has been specified, any blanks located at the beginning or end of the record will be eliminated before the parameter is evaluated.

When input records are being reorganized into a single unit, a blank will be inserted between each record, but not at positions were hyphenation has been canceled (i.e. where words have been rejoined (see parameter STR).

ANR Specifies whether successive records, whose records numbers match either in part or in their entirety, are to be organized into a text unit (organizing by numbering). [I] <0>

One numerical value can be specified:

0 = No organization of records by record number
1 = All successive records having the same page number are to be organized as a single text unit.
2 = All successive records having the same page-line number (regardless of distinction number) are to be organized into a single text unit.
3 = All successive records having the same record number are to be organized into a single text unit.

If 0 (default value) has been specified, organization will be based only on the following two parameters; if one of the numbers 1 to 3 has been specified, a further breakdown of the created text units can be carried out using the following two parameters.

AA Character strings placed at the beginning of a record (after any leading blanks have been eliminated) which mark the start of a text unit. [VIIIa]

AE Character strings placed at the end of a record (after any trailing blanks have been eliminated) which mark the end of a text unit. [VIIIb]

STR Hyphenation [I] <0>

0 = Input data is not hyphenated
1 = Rejoin hyphenated words

Here a hyphen is considered to be a "-" which (after trailing blanks have been eliminated) is the last character in an input record if the second-to-last character is also a "-" or a letter and the third-to-last character is not a control character ($, &, @, \, _, #, %).

When hyphenation is turned off, a hyphenated "ck", which according to German hyphenation is written as "k-" and "k", will not be joined back to its "ck" form.

Replacing character strings during input

X Pairs of character strings (and exception strings). On input, the first character string of a pair will be replaced in the text by the pair's second character string. [x]

Replacement is carried out separately in each input record, even if a text unit consists of more than one record (see parameters ANR, AA and AE).

Before the character string is replaced, a blank is added to both the beginning and end of the record (after any previous blanks have been removed); both blanks are removed after replacement.

Before replacement takes place, the program first checks to see whether the record begins with the character string specified in parameter AA, or whether it ends with the character string specified in the parameter AE.

If hyphenated words are to be rejoined (parameter STR), the program checks whether the record ends with a hyphen after the character strings have been replaced.

Selecting text parts which contain index entries

If the entire text unit contains index entries to be generated, none of the following parameters are required.

A Character strings which mark the beginning of text parts containing index entries to be generated. [IX]

E Character strings which mark the end of the text part which containing index entries to be generated. [IX];

Each selected text part starts after the beginning marker and ends before the end marker, or at the end

of the text unit if no end marker is found. However, if parameter EA has been specified, the beginning marker is counted as part of the selected text part.

If both parameters A and E have been specified, all text parts marked with A/E will be selected in succession. The second text part begins after/at the beginning marker which follows the end of the first selected text. In this way, a beginning marker can coincide with the end marker of the preceding text part.

If only parameter A has been specified, the selected text part ends at the end of the text unit; if only the parameter E has been specified, the selected text part starts at the beginning of the text unit.

Defining index entries

Index entries may be defined in two ways: by using the parameters EA and EE to specify the beginning and/or end markers by which they are identified as entries, or by using the parameter TR to specify separator character strings located between index entries.

If the parameter TR is specified along with the parameters EA and/or EE, the text parts selected by EE/EA will be divided into separate index entries according to the separator character strings given in parameter TR.

After the index entries have been defined using parameters EA, EE and/or TR, leading and trailing blanks will be removed from the index entries.

EA Character strings which mark the beginning of an index entry. [IX]

EE Character strings which mark the end of an index entry. [IX]

Each selected index entry starts after the beginning marker and ends before the end marker, or at the end of the text unit (or of the text part selected by the parameter A and E) if no end marker is found.

If both parameters EA and EE have been specified, all text parts marked with EA/EE will be defined as index entries. The second entry begins after the beginning marker which follows the end of the first selected entry. In this way, a beginning marker can coincide with the end marker of the preceding text part.

If only parameter EA has been specified, the selected text part, which is defined as an index entry, ends at the end of the text unit; if only parameter EE has been specified, the selected text part, which is

defined as an index entry, starts at the beginning of the text unit.

TR Character strings used to separate individual entries. [IX]

Modifying index entries (not used when MODE=KWIC)

The following parameters should only be used in case the index entries cannot be processed in an unaltered form.

((Characters strings which serve as a left parenthesis when eliminating parts of index entries. [IX]

)) Character strings which serve as a right parenthesis when eliminating parts of index entries. [IX]

The text parts in parentheses (including the parentheses themselves) will be eliminated. Missing parentheses are added logically at either the beginning or end of the index entries.

XX Pairs of character strings (and exception strings). The first character string of a pair will be replaced by the pair's second character string [X]

Replacement takes place in every index entry after any text parts of an entry have been eliminated by using the parameters described above.

Assigning a type to each entry

If all index entries are to be assigned the same type, the following parameter is the only parameter required for this.

TYP Type assigned to index entries [I <0>

In case the index entries are to be assigned different types, there are two ways to assign each entry its specific type: individually or by area. When MODE=KWIC, assignment is possible by area only.

When assigning a type individually, the first character of each index entry (other than a blank) is interpreted as the type marker and is eliminated afterwards. It is not necessary to provide this type marker with its own identification. Any blank directly following it will be ignored.

When assigning a type to a particular area, parameter TK must be used to specify the identification, which is directly followed (with no intervening blanks) by the type marker valid for the respective index entries that follow it.

In both cases, the characters used as type markers must be given in parameter TKZ.

TK Character strings used to identify type markers. [Ix]

TKZ Characters used as type markers [vI]

Parameter TKN is used to assign type markers (a single character) to a specific type (a number). The types given in this parameter must be given in the same order as the corresponding type markers in parameter TKN. If parameter TKN has not specified, the 1st, 2nd, 3rd, ... type marker given in parameter TKZ will be assigned to type 1, 2, 3, ... etc.

TKN Specifies the type (number) for each type marker. [I]

Numbers are assigned to type markers according to the order of the characters given in parameter TKZ.

If less numbers are specified than the number of type markers in parameter TKZ, the surplus type markers will be assigned to type 0.

Determining key words (only when MODE=KWIC) For the KWIC index, each "index entry", as defined with the help of the parameters described above, makes up the text part which contains the key words. At the same time, it is also the context for these key words.

Should key words occur only in certain parts of the "index entry" (= text part serving as the context), these parts can be marked off with the following parameters.

ASW Character strings marking the beginning of text parts which contain the key words. [Ix]

ESW Character strings marking the end of text parts which contain the key words. [Ix]

If parameter ASW and/or ESW has been specified, only those key words will be processed that begin within the selected text parts.

Each selected text part begins with the beginning marker and ends before the end marker or at the end of the index entry if no end marker is found.

If both parameters ASW and ESW have been specified, the key words will be taken in succession from all text parts as specified with ASW/ESW. The second text part starts after the beginning marker which follows the end of the first text part. In this way a beginning marker may coincide with the end marker of the preceding text part.

If only parameter ASW has been specified, the selected text part, from which the key words are taken, ends at the end of the index entry; if only parameter ESW has been specified, the selected text part, from which the key words are taken, starts at the beginning of the index entry.

(SW Character strings serving as left parenthesis for marking off text parts from which key words are to be taken (if necessary, after being defined by ASW and/or ESW). [IX]

)SW Character strings serving as right parenthesis for marking off text parts from which key words are to be taken (if necessary, after being defined by ASW and/or ESW). [IX]

If parameters (SW and/or)SW have been specified, key words located entirely within the text parts in parentheses will not be processed.

If these parameters have been specified along with parameters ASW and/or ESW, each text part in parentheses will be checked within the text parts selected with said parameters.

Each marked-off text part begins with the character string serving as the left parenthesis, and ends after the character string serving as the right parenthesis.

If a missing parenthesis character is detected when checking for demarcated text parts, the program will logically insert a parenthesis at either the beginning or the end of the text part. This function is also in effect when only one of the parameters (SW or)SW have been specified.

Note: The context of the key words are not altered by these parameters; text parts in parentheses thus remain in their context.

Key words are separated by delimiter characters as specified in parameter SWT.

SWT Character strings to be used as a delimiter character between individual key words. [IX]

Defining the reference

Index entries can be provided with either a "standard reference", or with a reference using text parts taken from the input data. If the "standard reference" is to be used, none of the following parameters (except IRL) need to be specified.

The standard reference specifies the beginning of each index entry in terms of its page, line and word position. Page and line numbers (with distinction numbers) are based on the record numbers of the input file; word numbers are determined by counting the words within the line. 6 digits are reserved for the page number, 3 each for the line number and distinction number, and 2 for the word number.

For counting words, a single word is defined as a character string which is bordered by the start and/or end of the line and/or one or more blanks. Punctuation marks and special characters therefore either belong to a word or, when they are separated from a word by blanks, count as a word of their own. A hard space ("_") does not count as a blank.

If the reference is to be taken from the text, the text parts which are to serve as a reference must be indicated in the parameter RFA and/or RFE. If not specified otherwise in the parameter RFL, the reference can be up to 6 characters long. Each reference taken from the text is valid for all index entries that follow (up to the next reference). (For exceptions, see parameter RFG).

RFA Character strings which mark the beginning of the text part to be used as a reference. [IX]

RFE Character strings which mark the end of the text part to be used as a reference. [IX]

Each selected text part starts after the beginning marker and ends before the end marker, or at the end of the text unit if no end marker is found. If the selected text part contains any leading or trailing blanks, these will be eliminated automatically.

If both parameters have been specified, all text parts marked with RFA/RFE will be used as a reference. The second reference starts after the beginning marker that follows the end of the first selected reference. In this way, a beginning marker may coincide with the end marker of the preceding text part which is to serve as a reference.

If only parameter RFA has been specified, the text part selected as a reference ends at the end of the text unit. If only parameter RFE has been specified, the text part selected as a reference starts at the beginning of the text unit.

The following parameter is used to replace character strings in the text part selected with RFA/RFE before it is used as a reference (and before the parameter RFT, if specified, is evaluated).

RFX Pairs of character strings (and exception strings) in a text part to be used as a reference, where the pair's first character string is replaced by the second character string. [x]

A reference may also consist of more than one part. The individual parts of the reference are numbered in consecutive order. The beginning marker of a text part determines which text part selected by the parameter RFA/RFE will constitute a particular part of the reference. For this purpose, parameter RFZ can be used to specify the reference part number in the same order as the corresponding character strings are given in the parameter RFA/RFE. This reference part will be replaced in the reference; subordinate reference parts (ie., those with a higher number) will be replaced by blanks if not specified otherwise in parameter RFB.

RFZ Specifies the number of the reference part designated by each beginning marker. [I] <1,1,1,...>

Numbers are assigned to the character strings according to the order of the character strings given in parameter RFA.

RFB Specifies whether subordinate reference parts should be retained or replaced by blanks (i.e. deleted) [I] <0>

0 = replace with blanks (delete)
1 = retain other reference parts unchanged

In case the reference does not consist of a single part, the parameter RFL must be used to specify the maximum length of each individual reference part. The number of these length specifications establishes in turn the number of reference parts. If the reference consists of a single part, this parameter should only be specified in case the default maximum length (6 characters) is to be replaced by another value. If the entire length of the reference (ie. the sum of all values given in parameter RFL) is larger than 14, parameter IRL must be used to specify the corresponding value.

RFL Length (number of characters) of each reference part. [I] <6>

A text part selected with RFA/RFE can contain more than one reference part. In this case, parameter RFT must specify the character string which separates reference parts from one another.

RFT Character strings which separate individual reference parts from one another. [Ix]

The reference part with the number one (or the number specified in the parameter RFZ) is located between the beginning marker and the first separator character string. Each separator character is followed by the reference part with the next higher number. If a different order is desired, parameter RFN can be used to specify which reference parts are to be preceded by which separator character string parallel to the order of character strings given in the parameter RFT. However, the individual reference parts must always be given in ascending order. Thus, parameter RFN can be used to keep track of missing reference parts; in their place blanks will be inserted in the reference.

RFN Specifies the numbers of the reference parts which are defined by each (preceding) separator character string. [I]

Numbers are assigned to the separator character strings according to their order as given in parameter RFT.

If less numbers are specified than the number of separator character strings in parameter RFT, a reference part which is preceded by one of the remaining separator character strings is assigned the next higher number than that of the preceding reference part.

If, based on the value given in parameter RFN, an attempt is made to introduce a reference part whose number is equal to or less than that of the preceding reference part, the corresponding specification made in parameter RFN will be ignored. In this case, the reference part will be assigned the next higher number than that of the preceding reference part.

Fields of fixed length are provided for the individual reference parts in the output records. Parameter RFL can be used to define the length of each field. Parameter RFS is used to specify whether each reference part should be aligned to the right or to the left in the corresponding field. This is usually important for sorting purposes only (i.e. in case the reference has not been provided with its own sort key).

RFS Specifies whether each reference part should be aligned to the right or to the left in its corresponding field. [I] <0,0,0,...>

0 = align to the right
1 = align to the left

It is advisable to align a reference part to the left when it consists of letters, and to the right when it consists of digits.

A reference taken from the text is valid for each of the following index entries. If a text unit contains an index entry located before the first text part to be used as a reference, this index entry is assigned to the last reference of the previous text unit. If in this case the first reference of the current text unit is to be used instead, this can be specified in parameter RFG. But the preceding reference remains valid for index entries located in a text unit which follows the text part to be used as a reference.

RFG Specifies whether the first text part to be used as a reference should take effect from the beginning of the text unit. [I] <0>

0 = Takes effect from the point where it is located.
1 = Takes effect from the beginning of the text unit.

14 character positions are reserved for the reference in the output records (when MODE=+). This value can be modified by using parameter IRL. Please note that this modification must be taken into account in any subsequent programs used to process this data. A modification of the default value is necessary if the total reference length (= sum of the values specified in the parameter RFL) is greater than 14. This is also recommended in order to save available disk space when working with large amounts of data if the total reference length is less than 14.

IRL Specifies the internal reference length. This is the number of character positions that will be reserved for the reference in the output records. [I] <14>

If the standard reference has been chosen, only one of the values 6, 9, 12 and 14 may be given; in this case only those reference parts will be inserted for which there is enough space.

Marking the reference (when not in MODE=KWIC)

When preparing an index with the TUSTEP program PREPARE INDEX, the references for individual index entries can be treated in a variety of ways. For example, the reference can be omitted entirely from entries which serve as cross-references; in other cases an "f" or "ff" can be added to the reference, or two successive references can be generated in such a way as to identify the beginning and end of a certain portion of text. In order to accomplish this, the references of such index entries must be provided with the corresponding marker when preparing the index.

If none of the index entries are to contain a reference, MODE=- can be specified when using the programs PREPARE INDEX and GENERATE INDEX. If index entries without a reference are to be merged with entries having a reference, entries which are to be supplied with no reference when the index is generated (with the TUSTEP program GENERATE INDEX) must nevertheless be provided with a reference when the index is prepared (with the TUSTEP program PREPARE INDEX). These must be marked accordingly. There are two ways to do this: If all index entries generated by PREPARE INDEX are to be treated in a uniform fashion, parameter RFM can be used to specify whether or not the index entries are to be supplied with a reference later on. If only certain index entries are not to be supplied with a reference later on, these must be given a unique marker character (this cannot be a blank). This marker character must be specified in parameter ORF.

RFM Specifies whether the index entries are to be provided with a reference when the index is generated with the TUSTEP program GENERATE INDEX. [I] <1>

0 = Index entries are not to be provided with a reference
1 = Index entries are to be provided with a reference

ORF Character used to mark index entries which are to be included in the index without a reference. [VI]

The marker character must be located at the beginning of each entry (and after any type identification character). After it has been recognized as such, it will be eliminated. Any blank which directly follows it will be ignored.

If references are to be provided with additional characters, the corresponding entry must be given a unique marker (other than a blank). This marker must be specified in one of the two following parameters.

MF Character string used to mark the index entries after whose reference the character "f" (or another

character to be specified in the parameter F of the TUSTEP program GENERATE INDEX) is to be added. [VI]

MFF Character string used to mark index entries after whose reference the characters "ff" (or another character to be specified in the parameter FF of the TUSTEP programm GENERATE INDEX) are to be added.[VI]

The marker character must be located at the beginning of each entry (and after any type identification character). After it has been recognized as such, it will be eliminated. Any blank which directly follows it will be ignored.

If the references are used to specify the beginning and/or end of a text area, the corresponding entries must be given a unique marker (other than a blank). This marker must be specified in the parameter VON (beginning) or the parameter BIS (end).

VON Character used to mark index entries whose reference represents the beginning of an area. [VI]

BIS Character used to mark index entries whose reference represents the end of an area. [VI]

The marker character must be located at the beginning of each entry (and after any type identification character). After it has been recognized as such, it will be eliminated. Any blank which directly follows it will be ignored.

Frequency specifications (when not in MODE=KWIC)

Besides generating the index, the TUSTEP program GENERATE INDEX can also generate a frequency count. This means that identical index entries are counted when they are condensed, in other words, each index entry generated by the TUSTEP program GENERATE INDEX has an implicit frequency of 1. However, it is possible to provide index entries with an explicit frequency. This is done by specifying the corresponding numerical value after a unique character string at the end of the index entry. This character string and the frequency specification are eliminated after they are evaluated. Index entries without such frequency specification are assigned a frequency of 1. The character string used to specify the frequency must be given in parameter HFA. In addition, the parameter HFL is used to specify the maximum number of decimal places needed for the frequency field in the output record.

HFA Character strings which define the explicit frequency specification in the index entries. [IX]

HFL Number of decimal places be reserved for the frequency count in every generated index entry. [I] <0>

Inverting index entries (when not in MODE=KWIC)

The order of words in an index entry can be inverted (e.g. "Friedrich von Schiller" can be changed to "Schiller, Friedrich von"). This is accomplished by splitting the entry either at the character string specified in the parameter UMP, or at the last space between words, and then exchanging the two parts. In addition, the character string ", " is inserted between the two entry parts unless otherwise specified in parameter UME.

UMD Specifies whether the index entries are to be inverted. [I] <0>

0 = Do not invert entries
1 = Entries are to appear in the index in inverted form if they contain a character string specified in the parameter UMP, or a blank; otherwise they should be written to the index in their normal form.
2 = Entries are to appear in the index in their normal form, and also in their inverted form if they contain a character string specified in the parameter UMP, or a blank.
3 = Entries are to appear in the index in their inverted form if they contain a character string given in the parameter UMP; otherwise they are to appear in the index in their normal form.
4 = Entries are to appear in the index in their normal form and in their inverted form if they contain a character string specified in the parameter UMP.

The specifications 1 and 2 differ from 3 and 4 in that in the former, the last blank of the entry serves as the point of inversion in case the index entry cannot be inverted at a character string specified in the parameter UMP.

If the entries contain type markers which have been given in the parameter TKZ, a value can be given for each type marker in parameter UMD. In order to make the proper match to the type markers, these values are to be given in the same order as the characters are specified in parameter TKZ. If less numerical values are specified than the number of type markers in parameter TKZ, the surplus type markers will be assigned the value 0.

UMP Character strings marking the point where index entries are to be inverted. [IX]

If an index entry is to be inverted, it will be scanned from left to right for the character strings

specified in parameter UMP. When such a string is found, the index entry will be inverted at this position, unless this character string is at the very beginning or end of the index entry. If none of the specified character strings can be found, and 1 or 2 has been specified in parameter UMD, the index entry will be inverted at the last space between two words (one or more blanks) if present. After inversion, the found character string or the blank space will be eliminated. A blank located directly before or after the found character string will also be eliminated.

UME Character strings to be inserted between the two inverted parts of an index entry should inversion take place. [II] <, >

If the entries contain type markers specified in parameter TKZ, character strings can be assigned to each type marker in parameter UME. In order to make the proper match to the type markers, these character strings are to be given in the same order as the characters are specified in parameter TKZ. If less character strings are specified that the number of type markers, the surplus type markers will be assigned the character string ", ".

Selecting index entries and key words

If not all key words (when MODE=KWIC), or not all index entries (in all other modes) are to be included in the output, those which are to be included can be selected with the following parameters.

If an index entry, or key word, matches an entry specified in parameter T+ or T+U, it will be included in the index. In this case, the entry will no longer be checked against any other parameters described in this section that may have been specified. If no other parameters from this section have been specified, only these index entries will be included in the index.

T+ Index entries (or key words) to be included in the index. [III]

No distinction is made between uppercase and lowercase letters.

T+U Index entries (or key words) to be included in the index. [III]

A distinction is made between uppercase and lowercase letters.

If an index entry matches an entry specified in parameter T- or T-U, it will not be included in the index. In this case, the entry will no longer be checked against any other parameters from this section that may have been specified.

T- Index entries (or key words) to be omitted. [III]

No distinction is made between uppercase and lowercase letters.

T-U Index entries (or key words) to be omitted. [III]

A distinction is made between uppercase and lowercase letters

Parameters ZF+, TA+ and TE+ can be used to specify conditions under which an index entry (or key word) is to be included in the index. If one or more of these parameters are used, an index entry (or key word) must fulfill at least one of the given conditions if it is not to be omitted from the index.

ZF+ Character strings, of which at least one must occur in the index entry (or key word) if it is to be included in the index. [IX]

TA+ Character strings, of which at least one must match the beginning of the index entry (or key word) if it is to be included in the index. [VIIIa]

TE+ Character strings, of which at least one must match the end of the index entry (or key word) if it is to be included in the index. [VIIIb]

Parameters ZF-, TA- and TE- can be used to specify conditions under which an index entry (or key word) is not to be included in the index. If one or more of these parameters are used, an index entry (or key word) will be omitted from the index if it fulfills at least one of the given conditions.

ZF- Character strings, none of which should occur in the index entry (or key word) if it is to be included in the index. [IX]

TA- Character strings, none of which should occur at the beginning of the index entry (or key word) if it is to be included in the index. [VIIIa]

TE- Character strings, none of which should occur at the end of the index entry (or key word) if it is to be included in the index. [VIIIb]

If one or more of the parameters ZF+, TA+ and TE+ as well as one or more of the parameters ZF-, TA- and TE- have been specified, an index entry (or key word) must fulfill at least one of the conditions of the parameters ZF+, TA+ und TE+ and none of the conditions of the parameters ZF-, TA- and TE- if it is to be included in the index.

Checking the length of index entries

MTL Specifies the maximum length of an index entry. If an index entry is longer than specified here, it will be printed with the corresponding error message. [I] <999999>

Supplementing index entries (not applicable when MODE=KWIC)

Supplementary text can be added before and/or after the index entry (e.g. a general term under which the index entry is to appear). The supplementary text consists of parts of the regular text which have been marked as such; each supplementary text so marked is valid for all subsequent index entries. In the following description of the relevant parameters involved, those parameters beginning with "EV" refer to a supplementary text which is to be placed before each index entry, those parameters beginning with "EN" refer to one which is to be placed after each index entry.

EVK / ENK Character string to be inserted between the supplementary text and the index entry. [II]

The text part specified in this parameter can be regarded as a constant supplementary text in case the supplementary text is not to be taken from the input data by using one of the following parameters.

EVA / ENA Character strings which mark the beginning of the text part which will serve as a supplementary text. [IX]

EVE / ENE Character strings which mark the end of the text part, which will serve as a supplementary text. [IX]

Each selected text part begins after the beginning marker and ends before the end marker, or at the end of the text unit if no end marker is found.

If both the parameters EVA and EVE or ENA and ENE have been specified, all text parts marked with EVA/EVE or with ENA/ENE will be used in succession as supplementary text. The second text part begins after the beginning marker which follows the end of the first text part. In this way, the beginning marker can

coincide with the end marker of the preceding text part which is to be used as a supplementary text.

If only the parameter EVA or ENA has been specified, the text part selected as a supplementary text will end at the end of the text unit; if only the parameter EVE or ENE has been specified, the text part selected as a supplementary text will begin at the beginning of the text unit.

The following parameter is used to replace character strings in text selected with the parameters EVA/EVE or ENA/ENE before the text is taken as a supplementary text (and, if applicable, before the parameters EVT or ENT are evaluated).

EVX / ENX Pairs of character strings (and exception strings); the first character string of a pair will be replaced by its second character string in the text part to be used as a supplementary text. [x]

A supplementary text can also consist of a number of text parts (e.g. when using a hierarchy of general terms). The individual parts of the supplementary text are numbered in ascending order. The beginning marker of these text parts is used to specify which part of the supplementary text is actually selected with EVA/EVE or ENA/ENE. For this purpose, the parameter EVZ or ENZ can be used to specify (parallel to the order of the character strings given in the parameter EVA or EVE) which part of the supplementary text each character string corresponds to. This part of the supplementary text will be replaced; subordinate parts of the supplementary text (i.e. those with a higher number) will be deleted, unless otherwise specified in parameter EVB/ENB.

EVZ / ENZ Specifies the number of the supplementary text part which has been marked by separate beginning markers. [ı] <1,1,1,...>

Numbers are assigned to the character strings in the same order as the character strings are given in parameter EVA or ENA.

EVB / ENB Specifies whether subordinate supplementary text are to be retained or deleted [ı] <0>

0 = delete subordinate supplementary text parts
1 = retain other supplementary text parts unchanged

A text part selected with EVA/EVE or ENA/ENE can also contain more than one part of the supplementary text. In this case, the parameter EVT or ENT must be used to specify the character strings which separate the individual text parts.

EVT / ENT Character strings which separate each supplementary text part from one other. [IX]

The supplementary text part with the number 1 (or with the number specified in parameter EVZ or ENZ) is located between the beginning marker and the first separator character string. Each separator character string is followed by the supplementary text part having the next higher number. This arrangement can be altered by using parameter EVN or ENN to specify (in the same order as the separator character strings are given in parameter EVT or ENT) which part of the supplementary text is introduced by which separator character string. However, the individual supplementary text parts must always be given in ascending order.

EVN / ENN Specifies the numbers of the supplementary text parts which are introduced by the individual separator character strings. [I]

Numbers are assigned to separator character strings in the same order as the character strings are given in parameter EVT or ENT.

If less numbers than separator character strings are specified in the parameter EVT or ENT, a supplementary text part which is introduced by one of the surplus separator character strings is allocated 1 number greater than the preceding supplementary text part.

If an attempt is made to introduce a supplementary text part whose number (based on the specifications given in the parameter EVN or ENN) is equal to or less than that of the preceding supplementary text part, the corresponding specification in parameter EVN or ENN will be ignored. In this case, the supplementary text part will be allocated 1 number greater than that of the preceding supplementary text part.

Sorting by group

NSN Character strings used to mark the beginning of a new group of index entries; i.e. the point from which index entries are given a new sort number. The number of digits needed for the sort number must be specified in parameter SNL.[VIII]

SNL Specifies the maximum number of digits needed for the sort number. [I] <0>

Determining the DESTINATION file

The program PREPARE INDEX can also be used to compile index entries for different indexes as long as each index entry is provided with a type marker as specified in parameter TKZ. This is done by specifying a DESTINATION file for each index in the command PREPARE INDEX. The following parameters are needed to specify which index entries are to be written to which DESTINATION file.

ZD Specifies the identification number of the DESTINATION file to which the index entries, depending on their type markers, are to be written. [I]

The DESTINATION files are assigned numbers in ascending order, i.e. the first file is number 1, the second is number 2, etc.

Numbers are assigned to the type markers according to the order of type markers given in the parameter TKZ.

If the value 0 is given instead of a number of a DESTINATION file, those index entries with the corresponding type markers will not be included in the output.

If less numbers are given than the number of markers in the parameter TKZ, the index entries with the remaining type markers will not be included in the output.

The parameter ZDU is used to specify the DESTINATION file for index entries having undefined type markers (i.e. a type marker not specified in parameter TKZ).

ZDU Specifies the number of the DESTINATION file to which index entries having undefined type markers are to be written. [I] <0>

The DESTINATION files are assigned numbers in ascending order, i.e. the first file is number 1, the second is number 2, etc.

If the value 0 is given instead of a number of a DESTINATION file, index entries with an undefined type marker will not be included in the output.

Generating the sort keys

A total of 9 sort keys may be defined. They are numbered consecutively from 1 to 9. In the following parameters that require a numerical value, a numerical value can be specified for each sort key in their respective order: the first number in each specification applies to the first sort key, the second to the second sort key, etc. For all other paramters, the last character of the parameter label is a digit (where n stands for this digit), which specifies the number of the sort key for which the parameter is valid.

The sort key can be used as follows:

- for MODE=-

  Sort keys 1 to 3 for the index entry

- for MODE=+

  Sort keys 1 to 3 for the index entry
  Sort keys 7 to 9 for the reference

- for MODE=KWIC

  Sort keys 1 to 3 for the key word
  Sort keys 4 to 6 for the context
  Sort keys 7 to 9 for the reference

In the following, the term "sort text" refers to the text created from the respective sort key according to the specifications given in the following parameters, i.e. the text for the index entry, key word, context, or reference, depending on which sort key is to be generated.

Should the sequence of references in the input data correspond to the desired sequence of references in the index entry, the reference generally does not have to be taken into account when making the sort.

The reference therefore requires a sort key only when it cannot be used as a sort field in its original form (for the commands #SORT or #MERGE), i.e. when the desired sequence cannot be realized with these commands.

If the reference consists of 2 or more parts, these will be directly appended to each other. Should it be necessary to separate the individual parts in order to generate the sort key, an identification marker can be inserted in front of each reference part with the following parameter.

RSK Text parts to be inserted in the sort text for marking individual reference parts. The first text part specified will be inserted in front of the first reference part, the second text part in front of the second, etc. [II]

This parameter is only relevant for generating the sort text from the reference for the sort keys 7 to 9. The reference itself is not altered in the process.

The following parameters (ASn to KLS) are only necessary whenever the entire sort text does not match the respective sort key.

ASn Character strings which mark the beginning of the sort text part which is to taken into account in the nth sort key. [IX]

ESn Character strings which mark the end of the sort text part which is to be taken into account in the nth sort key. [IX]

AES Index for ASn and ESn [I] <1,1,1,1,1,1,1,1,1>

1 = Selects the first text part marked with A/E (starting with the beginning marker and ending before the following end marker or at the end of the sort text).
If only A has been specified, the selected text part will end at the end of the sort text; if only E has been specified, the selected text part will begin at the beginning of the sort text.

0 = Selects that part of the sort text which would be excluded by choosing 1.

3 = As in 1. However, this selects not only the first text part marked with A/E but all text parts marked in this way (with the second text part starting with the beginning marker that follows the end of the first text part marked with A/E).
If only A has been specified, the selected text part will start at the last beginning marker occuring in the text unit and will end at the end of the sort text. If only E has been specified, the selected text part will start at the beginning of the sort text and end before the last end marker occuring in the sort text.

2 = Selects that part of the sort text which would be excluded by choosing 3.

If one of the values 0 to 3 has been specified, each beginning marker is counted as part of the text following it, while the end marker is not counted as part of the text. This treatment of beginning and end markers can be reversed by adding either 10 or 20 to the number chosen. If the value 10 is added (i.e. by entering a number from 10 to 13), each beginning marker will not be counted as part of the following text; if 20 is added, each end marker will be counted as part of the preceding text; if 30 is added, the beginning marker is not counted as part of the following text and the end marker is counted as part of the preceding text.
If 2 or 3 is chosen, the program is instructed to search the text for the next beginning marker starting at the first position that follows the end of the preceding text part. Therefore, for the values 2 and 3, the search for a new beginning marker starts at the first character of the preceding text's end marker,

since the end marker is not part of the preceding text. If 20 or 30 is added to these values, the search for the next beginning starts at the character which follows the last position of the most recently found end marker, since this marker counts as part of the preceding text.

(Sn Left parenthesis for selecting a sort text part (in case ASn and ESn have not been specified) or for eliminating text parts from the sort text part already selected by ASn and/or ESn which is to be taken into account in the nth sort key. [IX]

)Sn Right parenthesis which closes (Sn [IX]

KLS Index for (Sn and)Sn [I] <0,0,0,0,0,0,0,0,0>

0 = Eliminates the parts of the text in parentheses (including the parentheses themselves). Missing parentheses are added logically at either the beginning or end of the sort text or the text part which has already been selected.
1 = Selects all text parts which would be eliminated by option 0.
2 = As in 0, but unpaired parentheses are ignored instead of being logically provided with a complementary parenthesis.
3 = Selects those text parts which would be eliminated by option 2.

If the values 0 to 3 are specified, the parentheses themselves are considered part of the text in parentheses are thus either eliminated along with the text or are kept with it. This treatment of left and right parentheses can be reversed by adding either 10 or 20 to the value chosen. If 10 is added, (i.e. by entering a value from 10 to 13), left parentheses will not be counted as part of the text in parentheses. If 20 is added, right parentheses will not be counted as part of the text in parentheses. If 30 is added, no parenthesis is counted as part of the text in parentheses.

Nn Character strings used to mark words which are to be omitted from the nth sort key. This marking and the following characters, up to and including the following blank or apostrophe, will be eliminated in the respective sort key. [IX]

DEZ Number of positions to which a number in sort key is to be filled out with leading zeros. Numbers which have at least the specified number of digits remain unaltered. [I] <1,1,1,1,1,1,1,1,1>

Rn Character string used to mark Roman numerals which are to be converted into a string of 4 Arabic numerals for the nth sort key. The character strings must be placed directly before the Roman numeral. [IX]

XSn Pairs of character strings (and exception strings); the first character string of a pair will be replaced by the pair's second character string in the nth sort key. [X]

A1 - A3 Specifies a sorting alphabet for the nth sort key. [VII]

SSL Specifies the length of each sort key. [I] <0,0,0,0,0,0,0,0,0>

This parameter is obligatory.

RLF Specifies whether the sort key is to be written in normal or reverse order. [I] <0,0,0,0,0,0,0,0,0>
0 = Arrange sort key in normal order
1 = Arrange sort key in reverse order (i.e. starting with last character)

The sort keys 4 to 6 for the context can be arranged in either normal or reverse order. In normal order, the context part to the right of the key word will be used as the sort text. In reverse order, the context part to the left of the key word will be used.

Alphabetical list of parameters

The character "n" in the parameter labels stands for the numerals 1 to 9(e.g. Rn stands for R1, ..., R9).

Further processing of data

After the data have been prepared for sorting with this program, they must then be sorted with the command #SORT. If two or more DESTINATION files have been specified, each of these must sorted separately and subsequently merged back into one file with the command #MERGE.

After sorting (and any merging) the index entries (or key words with their respective context) can be processed for output by using the TUSTEP program GENERATE INDEX.

If the sort key and the actual index entries have been written to different files, these files must be merged into a single file. To accomplish this, the file which was given in the DATA specification (or -STD- for the standard DATA file) in the PREPARE INDEX program must also be given in the DATA specification of the TUSTEP program GENERATE INDEX.

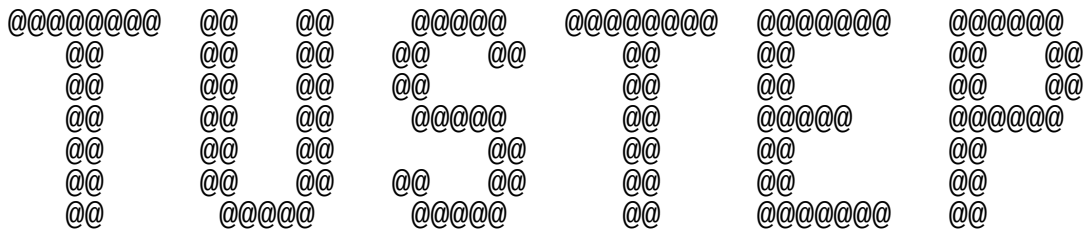
Structure of a data record


```
MODE = -                          ++++ ++++++++ ++++ ============
MODE = +    ******* ***** ***** ++++ ++++++++ ++++ ============
```

REF Reference (normally 14 characters)
TYP Type (1 character)
STB Control bits (1 character)
SN Sort number
SS Sort key
HF Frequency

The length of the sort number and sort key depends on the length specified in the parameters SNL and SSL; the standard reference length can be altered by the parameter IRL.

=== Input data
*** Data supplied automatically
+++ Data supplied upon specification of the corresponding parameters



Tübingen

System of Text Processing Programs

Program

P R E P A R E S O R T I N G

1993

TÜBINGEN UNIVERSITY - CENTER FOR DATA PROCESSING

Department of Literary and Documentary Data Processing

Brunnenstrasse 27, D-72074 Tübingen

Survey:

Command:

#PRESORT

Specifications:

SOURCE	= file		Name of the file containing the data to be prepared for sorting
	= -STD-		The standard TEXT file contains the data to be prepared for sorting
DESTINATION	= file		Name of the file to which the data prepared for sorting are to be written. More than one file may be specified.
	= -STD-		The data prepared for sorting are to be written to the standard TEXT file.
MODE	= -		Generate sort key only
	= +	*	Generate sort key and add REF/TYPE/STB
	= R		Input data contain REF/TYP/STB; add sort key
	= K		Input data are correcting instructions; add corrrection key and, if indicated by parameters, sort key
	= S		Input data are correcting instructions with correction key; add sort key
ERASE	= -	*	If the DESTINATION, DATA, or LISTING file already contains data, they are to be retained.
	= +		If the DESTINATION, DATA, or LISTING file already contains data, they are to be erased beforehand.
PARAMETER	= file		Name of the file containing parameters
	= *		Parameters follow the command and are ended by *EOF
DATA	= -	*	The data are to be written to the DESTINATION file in their entirety.
	= file		Name of the file to which the text part of each record is to be written.
	= -STD-		The text part of each record is to be written into the standard DATA file.
LISTING	= -	*	No trace
	= +		Trace listing is to be written into the journal.

= -STD-	Trace listing is to be written to the standard LISTING file.
= file	Name of the file to which the trace listing is to be written.

Features:

With this command, text units (consisting of one or more input records which may also be correcting instructrions) can be prepared for sorting. It is possible to define the criteria for sorting.

General information:

Input data are generally arranged into text units (see: "Organizing records into text units "), so that each text unit represents a sorting unit.

If sorting is not to be carried out according to the wording of the text but according to certain parts of the text unit, selective parameters (see: "Generating the sorting text") are used to specify which parts of the text are to be sorted into which order. These will be organized into a sorting text, with each part separated by two additional blanks. If no selective parameters are specified, the sorting text is identical to the text of the text unit.

One to three sort keys may be constructed from the sorting text. This is necessary because the order of characters in the the computer's internal code does not correspond to the usual alphabetical order. The sort key is used for sorting purposes only and is subsequently eliminated. Only one sort key is necessary for texts containing no umlauts or diacritical marks. If the text contains umlauts, the usual alphabetical order (as defined by DIN 5007) is preserved by encoding ä, ö, ü and ß as ae, oe, ue and ss for the first sort key. A second key is necessary, however, if words such as "Maße" and "Masse" or names such as "Jäger" und "Jaeger" are to be kept separate from each other. For this sorting key, ä, ö, ü and ß can be encoded as az, oz, uz und sz. If a text contains diacritical marks, a second sort key is also necessary. Normally, all diacricritical marks are eliminated for the first sort key in order that sorting can be first carried out in alphabetical order; accents and other diacritical marks only affect the order of two identical characters. For the second sort key, each accent may be encoded as a decimal number, for example, so that words which differ only by diacritical marks may be arranged in the desired order. In addition, if uppercase and lowercase letters are decisive in the sorting process, a third key is required. If, however, umlauts and diacritical marks do not occur in the text, sorting of uppercase and lowercase letters can be handled by the second sort key.

If the text units of an imput file are organized by group, they can be sorted within each of their respective groups only. This requires that the beginning of each group is marked in a unique way, such as using a marker before each section title. Each group is numbered internally and the corresponding number placed directly in front of the sort key of the individual text units. This keeps groups in their original, unchanged order.

The program's memory limits the length of each text unit. If memory is not large enough to handle extremely large text units, they can be broken down into smaller text units which are then organized into a single sorting unit whose size is unlimited. Text units which belong to the same sorting unit as the previous text unit must be marked as such. However, all sorting criteria must be contained in the first text unit of each sorting unit.

The sort program (#SORT comand) requires a certain amount of memory for temporary storage of the data to be sorted. Large

amounts of data may exceed available memory. This can usually be avoided by outputting the sort key and the actual text data to separate files (DESTINATION file and DATA file, respectively). Of these two files, only the DESTINATION file containing the sort keys needs to be sorted. If the amount of available memory is still too small for sorting, the sort key can be distributed among a number of files by specifying the appropriate DESTINATION files (no additional parameters are necessary). These must be sorted individually and then merged back into a single file.

Parameters

Values in < > refer to initial settings.

Values in [ ] refer to the type of parameter employed. The various types of parameters are described in the chapter "TUSTEP Basics".

In certain parameters, text parts can be selected for further processing by means of beginning and/or end markers as well as markers that serve as left and right parentheses for selecting the desired part. The manner in which these parameters function is also described at the end of the "Parameters" chapter of "TUSTEP Basics".

In addition to the parameters described below, parameters for defining string groups and character groups may also be employed. [v]

Parameter SSL must be specified at the very least.

Selecting data

If the entire file is to be processed, none of the following parameters are necessary.

BER Definition of an area ("page.line-page.line") or a starting point ("page.line"). This parameter is only used when not processing the entire file. [xI]

If a segment of a segment file is to be processed, the name of the segment can be substituted for the area.

This parameter can only be used when the record numbers of the input files are in ascending order.

MAX For test runs, this specifies how many text units (= input records, if neither the parameters AA or AE are specified) to be prepared for sorting. [I] <999999>

Organizing records into text units

In case each input record contains a complete text unit (sorting unit), the following parameters should not be used. If this is not the case, the parameters AA and/or AE can be used to organize more than one record into a single text unit.

If one of the following four parameters has been specified, any blanks located at the beginning or end of the record will be eliminated before the parameter is evaluated.

When input records are being reorganized into a single unit, a blank will be inserted between each record, but not at positions were hyphenation has been canceled (i.e. where words have been rejoined (see parameter STR).

ANR Specifies whether successive records, whose records numbers match either in part or in their entirety, are to be organized into a text unit (organizing by numbering). [I] <0>

One numerical value can be specified:

0 = No organization of records by record number
1 = All successive records having the same page number are to be organized as a single text unit.
2 = All successive records having the same page-line number (regardless of distinction number) are to be organized into a single text unit.
3 = All successive records having the same record number are to be organized into a single text unit.

If 0 (default value) has been specified, organization will be based only on the following two parameters; if one of the numbers 1 to 3 has been specified, a further breakdown of the created text units can be carried out using the following two parameters.

AA Character strings placed at the beginning of a record (after any leading blanks have been eliminated) which mark the start of a text unit. [VIIIa]

AE Character strings placed at the end of a record (after any trailing blanks have been eliminated) which mark the end of a text unit. [VIIIb]

STR Hyphenation [I] <0>

0 = Input data is not hyphenated
1 = Rejoin hyphenated words

Here a hyphen is considered to be a "-" which (after trailing blanks have been eliminated) is the last character in an input record if the second-to-last

character is also a "-" or a letter and the third-to-last character is not a control character ($, &, @, \, _, #, %).

When hyphenation is turned off, a hyphenated "ck", which according to German hyphenation is written as "k-" and "k", will not be joined back to its "ck" form.

Checking the length of text units

MTL Specifies the maximum length of a text unit [I]

If a text unit is longer than specified, it will be printed out along with a corresponding error message.

Organizing text units into a sorting unit

FS Character string used to designate (continuation) text units. If the specified character is placed at the beginning of a text unit, this text unit will not be treated as a sorting unit but rather considered as a continuation of the preceding text unit. [VIIIa]

This parameter can only be used when MODE=-. Furthermore, a file name (or -STD- for the standard DATA file) must be given for the DATA specification.

Sorting by group:

NSN Character string used to mark the beginning of a new group of text units; i.e. the point from which text units are given a new sort number. The number of digits needed for the sort number must be given in parameter SNL. [VIIIa]

SNL Specifies the maximum number of digits available for the sort number. [I] <0>

Generating the sorting text:

AK1/AK9 Character strings used to mark the beginning of the 1st,2nd,...,9th part of the sorting text. [Ix]

EK1/EK9 Character strings used to mark the end of the 1st,2nd,...,9th part of the sorting text. [Ix]

AEI Index for AK1,EK1... AK9,EK9 [I] <1,1,1,1,1,1,1,1,1>

1 = Selects the first text part marked with A/E (starting with a beginning marker and ending before the following end marker or at the end of the text unit itself).
If only A has been specified, the selected text part will end at the end of the text unit; if only E has been specified, the selected part will begin at the beginning of text unit.
0 = Selects that part of the text unit which would be excluded by choosing 1.
3 = As in 1. However, this selects not only the first text part marked with A/E but all text parts marked in this way (with the second text part starting with the beginning marker which follows the end of the first text part marked with A/E).
If only A has been specified, the selected text part will begin at the last beginning marker occuring in the text unit, and will end at the end of the text unit. If only E has been specified, the selected text part will start at the beginning of the text unit and end before the last end marker of the text unit.
2 = Selects that part of the text unit which would be excluded by choosing 3.

If one of the values 0 to 3 is specified, each beginning marker is counted as part of the text following it, while the end marker is not counted as part of the text. This treatment of beginning and end markers can be reversed by adding either 10 or 20 to the number chosen. If the value of 10 is added (i.e. by entering a number from 10 to 13), each beginning marker will not be counted as part of the following text; if 20 is added, each end marker will be counted as part of the preceding text. If 30 is added, the beginning marker is not counted as part of the following text and the end marker is counted as part of the preceding text.
When 2 or 3 is chosen, the program is instructed to search the text for more than one beginning marker, and will look for a new beginning marker starting at the first position after the end of the preceding text part. Therefore, for the values 2 and 3, the search for a new beginning marker starts at the first character of the preceding text's end marker, since the end marker is not part of the preceding text. Beginning and end markers may thus overlap in the

text. If 20 or 30 is added to these values, the next beginning marker is searched from the character which follows the last position of the most recently found end marker, since this marker is counted as part of the preceding text.

(K1/(K9 Left parenthesis for selecting the 1st,2nd,...,9th part of the sorting text (in case AK1|EK1... AK9|EK9 has not been specified) or for eliminating text from the 1st,2nd,...,9th part of the sorting text already selected with AK1|EK1... AK9|EK9 [Ix]

)K1/)K9 Right parenthesis which closes (K1 ... (K9 [Ix]

KLI Index for (K1,)K1 ... (K9,)K9 [I] <0,0,0,0,0,0,0,0,0>

0 = Eliminates the parts of the text in parentheses (including the parentheses themselves). Missing parentheses are added logically at either the beginning or end of the text unit or text part which has already been selected.
1 = Selects all text parts which would be eliminated by option 0.
2 = As in 0, but unpaired parentheses are ignored instead of being logically provided with a complementary parenthesis.
3 = Selects those text parts which would be eliminated by option 2.

If the values 0 to 3 are specified, the parentheses themselves are considered part of the text in parentheses and are thus either eliminated along with the text or are kept with it. This treatment of left and right parentheses can be reversed by adding either 10 or 20 to the value chosen. If 10 is added (i.e. by entering a value from 10 to 13), each left parenthesis will not be counted as part of the text in parentheses. If 20 is added, each right parenthesis will not be counted as part of the text in parentheses. If 30 is added, neither parenthesis will be counted as part of the text in parentheses.

((1/((9 Left parenthesis for selecting the 1st,2nd,...,9th part of the sorting text (in case AK1|EK1... AK9|EK9 and/or (K1|)K1 ... (K9|)K9 have not been specified), or for eliminating text from the 1st,2nd,...,9th part of the sorting text already selected with AK1|EK1... AK9|EK9 and/or (K1|)K9 [Ix]

))1/))9 Right parenthesis for ((1 ... ((9 [Ix]

DKI Index for ((1,))1 ... ((9,))9
[I] <0,0,0,0,0,0,0,0,0>

Specifications match those used for parameter KLI

XX1/XX9 Replacing character strings in the 1st,2nd,...,9th part of the sorting text. [x]

ERG Text parts to be added before the first part of the sorting text, between each individual part of the sorting text, and after the last part of the sorting text. [II] <no additions made before the first and after the last part, two blanks are added between each individual part of the sorting text>

MLS Specifies the maximum length for each individual part of the sorting text. It is especially advisable to limit the length for a part of text when, on one hand, the corresponding parts of the sorting text differ after a certain number of characters but, on the other hand, are so long that the sorting text thereby becomes unnecessarily long, or even so long that the following parts of the sorting text can no longer be taken into account in the sort key because the sort key will be filled up by the extremely long text part. [I] <999999,...,999999>

Generating the sort keys from the sorting text:

The following parameters (AS1 - AS3 to KLS) are only necessary in case the entire sorting text is not to be taken into account in the respective sort key.

AS1 - AS3 Character string which marks the beginning of the part of the sorting text which is to be taken into account in the 1st,2nd,3rd sort key. [Ix]

ES1 - ES3 Character string which marks the end of the part of the sorting text which is to be taken into account in the 1st,2nd,3rd sort key. [Ix]

AES Additional specifications for AS1,ES1, ... AS3,ES3 [I] <1,1,1>

Values here are analogous to those used in the parameter AEI

(S1 - (S3 Left parenthesis for selecting the part of the sorting text (in case AS1|ES1 ... AS3|ES3 is not specified) or for eliminating parts of text from the part of the sorting text already selected by AS1|ES1 ... AS3|ES3 which is to be taken into account in the 1st,2nd,3rd sort key. [Ix]

)S1 -)S3 Right parenthesis which closes (S1 ... (S3.

KLS Index for (S1,)S1, ... (S3,)S3 [I] <0,0,0>

Values here are analogous to those used in the parameter KLI.

N1 - N3 Character strings used to mark words which are to be omitted from the 1st,2nd,3rd sort key. This marking and the following characters, up to and including the following blank or apostrophe, will be eliminated in the respective sort key. [IX]

DEZ Number of positions to which a number in a sort key is to be filled out with leading zeros. Numbers which consist of the specified number of digits (or more) remain unaltered. [I] <1,1,1>

R1 - R3 Character string used to mark Roman numerals which are to be converted into a string of 4 Arabic numerals for the 1st,2nd,3rd sort key. The character strings must be placed directly before Roman numerals. [IX]

XS1 - XS3 Replacing character strings for the 1st,2nd,3rd sort key. [X]

A1 - A3 Specifies a sorting alphabet for the 1st,2nd,3rd sort key. [VII]

SSL Specifies the length of 1st,2nd,3rd sort key. [I] <0,0,0>

This parameter is obligatory.

The individual modes and their specific parameters:

MODE = -

Only sort keys are generated for the text units (each of which may consist of more than one input record).

MODE = +

As in MODE=- . In addition, for each text unit a reference (REF), a type (TYP) und control bits (STB) are inserted before the sort key or sort number. The reference indicates where the text unit is located in the source file. It usually consists of 14 characters: 6 characters for the page number, 3 characters for the line number, 3 characters for the distinction number, 2 blanks (reserved for the word number used by other programs). If a text unit consists of more than one line, its first line determines the reference. The type (1 character) is determined by the parameter TYP. It can be used for output control with the TUSTEP program GENERATE INDEX. Of the STB control bits, only one (i.e. one character) can be set or erased by parameter RFM. This bit controls whether the reference of the sorting units will be included (bit set) in, or excluded (bit erased) from the output generated by the TUSTEP programs GENERATE INDEX.

TYP Specifies the type for all text units. [I] <0 = undefined>

RFM Specifies whether the sorting units should be supplied with a reference when processed by the TUSTEP program program GINDEX. [I] <1>

0 = sorting units to be supplied with a reference
1 = sorting units are not to be supplied with a reference

IRL Specifies the number of characters for the reference (REF). [I] <14>

MODE = R

As in MODE=+, but each input record contains a (complete) text unit (parameters AA and AE prohibited in this case), and REF/TYP/STB are already present. The type and reference bits can be altered by using the parameters TYP and RFM, respectively.

MODE = K

Each input record contains a correcting instruction (e.g. as generated by the TUSTEP program COMPARE), for which correction keys and, depending on the parameter SSL, sorting keys are to be generated. The correction key is the program's internal encoding of part of the correcting instruction and is 44 characters long. Its structure is identical to that of the correction key used in the TUSTEP program COMPARE (specified there by parameter SW). Should the correcting instructions contain a version identification (e.g. as inserted by parameter VKZ in the TUSTEP

program COMPARE), this can be converted into a reference by using the parameter RFL. In this case, REF/TYP/STB is added as in MODE=+.

RFL Maximum length of the version identification of the correcting instruction which is to be converted into a reference. [I] <0>

SW Specifies the sorting value to be inserted into the correcting instruction. [I] <0>

MODE = S

As in MODE=K, but here the correction key is already present (generated by using the SW parameter in the TUSTEP program COMPARE. The sorting value can be altered by specifying this in parameter SW.

Alphabetical list of parameters

The character "n" in the parameter labels stands for the numerals 1, 2, 3 (e.g. Rn stands for R1, R2 and R3), the character "m" for the numerals 1 to 9.

Further processing of data after sorting:

After the data have been prepared for sorting with this program, they must then be sorted with the command #SORT. If two or more DESTINATION files have been specified, each of these must sorted separately and subsequently merged back into one file with the command #MERGE.

After sorting (and any merging) has been completed, the text units can be restored to the normal TUSTEP text format by using the TUSTEP program COPY, or can be processed for output with the TUSTEP program GENERATE INDEX.

If the sort key and the actual text files have been written to different files, these files must be united in a single file. To accomplish this, the file which was given for the DATA specification (or -STD- for the standard DATA file) in the PREPARE SORTING program must also be given in the DATA specification of the TUSTEP programs COPY or GENERATE INDEX.

Unless the file given for the DATA specification (or -STD- for the standard DATA file) has already been specified in the SORT command, the data assembled from more than one text unit into a single sorting unit can be further processed only by the TUSTEP program COPY in MODE=S.

Structure of a data record:

	REF	TYP	STB	KS	SN	SS	Text ...
MODE = -					++++	++++++++	==========
MODE = +	*****	*****	*****		++++	++++++++	==========
MODE = R	=====	=====	=====		++++	++++++++	==========
MODE = K	+++++	+++++	+++++	******	++++	++++++++	==========
MODE = S	+++++	+++++	+++++	======	++++	++++++++	==========

REF Reference (normally 14 characters)
TYP Type (1 character)
STB Control bits (1 character)
KS Correction key (44 characters)
SN Sort number
SS Sort key

The length of the sort number and sort key depends on the length specified in the parameters SNL and SSL; the standard reference length can be altered by the parameter IRL.

=== Input data
*** Data supplied automatically
+++ Data supplied upon specification of the corresponding parameters

Structure of a correction key:

The correction key consists of a total of 44 characters and is structured as follows:

APO	EPO	KA	SW	FNR	POS

APO	1	17	Starting position of the corrective text
	1	6	page number
	7	3	line number
	10	3	distinction number
	13	2	word number
	15	3	character number
EPO	18	17	End position of the corrective text
	18	6	page number
	24	3	line number
	27	3	distinction number
	30	2	word number
	32	3	character number
KA	35	2	Type of correction:
	35	1	0 = error 1 = page-line-word-character 2 = page-line-word 3 = page-line
	36	1	0 = error 1 = comment (*) 2 = delete (-) 3 = replace (=) 4 = insert (+)
SW	37	2	Sorting value (version number)
FNR	39	3	Continuation number (for continuation lines)
POS	42	3	Position of the correction code (*, -, =, +) in the correcting instruction

The first number specifies the position of each character within the correction key, the second number specifies the maximum number of characters allowed.