TMaker™

QUICK REFERENCE BOOKLET

DATABASE

COMMANDS

SPELLING

CHECKER

DISCLAIMER

No representations or warranties are made with respect to the contents hereof. Further, T/MAKER Company reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation to notify any person of such revision.

transfer (Data Transfer Command)

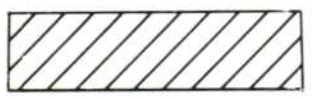
The Transfer command transfers values from a file on disk into the working area.

WHAT NEXT? ***transfer*** *disk-file-name or* ***blank***

Mask Lines

To use the Transfer command, mask lines are placed in the working area and in the file on disk immediately ahead of the line to which they refer. The shaded area below shows the zone defined as "total" according to the mask line.

{ total }

Total 

Aside from the name of the value (i.e. "total"), the braces must contain a special character. For zones in the working area which are to be replaced by values, the following characters may be used.

Character	Interpretation
—	value centered in zone
>	value right-justified with right-most limit of zone
<	value left-justified with left-most limit of zone

If none of the above three characters is used, the zone is assumed to be "protected" and no data will be placed into it.

For zones in the disk file which are to be sources of data, the symbol "!" must be placed within the braces.

A zone may be both a source of data when in a disk file and a destination for data when in the working area (see below).

{ > ! ratio }

If a zone in the working area is too small to hold a value, the zone is filled with asterisks.

See: Load (Data Files) for the rules governing the name of a value for a set of braces containing no name.

Unload (Long Names in Masks) for the means to specify names which are longer than the zones they define.

Note: Mask lines may be made Print Comment Lines so that they do not appear in printed copy.

The Blank Option

When the word "blank" is given in place of a disk-file-name, all unprotected zones in the working area are filled with spaces.

Example

Disk File "EXP"

```
     Expenses
ex                  999,999.99
+    Rent             1,240.00
+    Food               350.00
                   { extotal! }
=    Total            1,590.00
```

Working Area

```
     Summary
ex                  999,999.00
+    Income           4,000.00
                   {extotal>}
–    Expenses
=    Net
```

Result of: ***transfer exp***

```
     Summary
ex                  999,999.99
+    Income           4,000.00
                   {extotal>}
–    Expenses         1,590.00
=    Net
```

records

T/Maker's Data-Base commands deal with files of records. Such files contain a "record definition" which describes the manner in which a record is organized.

A record is a group of contiguous lines. On each line data values may be stored in fixed locations called "fields."

Record Definition

Any file of records must contain a record definition prior to the first record. The record definition shows:

- the identifying label for each line in the record
- the names and locations of fields in the record
- the manner in which the data in the field is justified

Record definitions are preceded by a line containing ***< record >*** and ended by a line containing ***< end >*** .

Record definitions look like masks used by Load and Unload, however, the mask is assumed to apply repetitively to each successive record in the file.

Example

```
< RECORD >
A { name                } { age        }
B { city                }
< END >
```

Line Labels

Normally each line in a record begins with a unique label. The label extends from the first position of the line to the first brace. A close-brace may be used to de-liniate the label without beginning a field.

```
< RECORD >
+ }         { category   } { amount    }
< END >
```

In some instances it is useful to have the same label for a number of lines in the record.

```
< RECORD >
1 { head                                }
2 { dependent1                          }
2 { dependent2                          }
2 { dependent3                          }
< END >
```

This should only be done in situations like the one above where it remains nonetheless possible to identify the start of the new record because the first line does begin with a unique label.

Field Justification

Special symbols within the braces are used to denote the manner in which the value for the field should be justified.

Symbol	Interpretation
—	value centered in field
>	value right-justified with right-most limit of field
<	value left-justified with left-most limit of field
!	value left-justified and unused space after data eliminated ("squeezed")

The last symbol may destroy the validity of the record definition by shifting the positions occupied by following fields on the line. It should only be used to create temporary files intended for printing.

If a field contains no justification symbol, the same justification as the previous field is used.

For most record processing commands, right-justified values are assumed to be numeric. Fields justified in another fashion are assumed to be alphabetic. One should have a valid reason for violating this convention.

Optional Lines

Lines in a record may be made optional (i.e., they will not be generated if all fields are blank) by ending the line with a pair of close-braces.

```
<RECORD>
1 { head                    }
2 { dependent1              } }
2 { dependent2              } }
2 { dependent3              } }
<END>
```

The first line in a record should never be made optional unless it is the only line in the record.

Non-Record Lines

Files of records may contain lines which are not parts of a record (e.g., headings). By adding a record definition to an existing list or table, data base commands may be used on the list or table.

Multiple Record Definitions

A file may contain more than one record definition. In such cases, the first record definition encountered in the working area (not before it) or the first record definition in a disk file will be used. The Update command, however, always uses the first definition in the file.

Field Names

The names of fields are placed within the braces that define the location of the field. Names should not contain any of the following special symbols:

— < > = ! # + — * / !

In general, it is best to begin names with a letter (alphabet character) and to pick names which are not themselves data values or keywords used with other commands such as "when" or "contains" (used by Select). Consider that short names conserve space.

If a set of braces contains no name, its name is assumed to be "one higher than the previous name" (see Load – "Data Files").

If a name is too long to fit within the corresponding braces, the conventions described in "Long Names in Masks" may be used (see Unload). For fields which are only one character long, double number signs are used immediately after the open brace.

{##gender}

Note: Data Transfer commands do not acknowledge this convention.

Each field should have a unique name.

Note: This rule is sometimes profitably violated when creating temporary files with the Select command.

update

The Update command permits files of records to be altered.
WHAT NEXT? ***update***

General

The working file to be updated must contain a record definition and, optionally, form definitions. When no form definition is present, the record definition itself is used as a form.

Form Definition

A form is a screen image used for the purpose of entering, updating, or displaying a record (see Example below).

A form definition begins with a line containing ***< form >*** and ends with a line containing ***< end >***.

Form definitions are placed in the working area ahead of the correct record definition.

A form must fit entirely on one screen. Up to ten different forms may be used.

Forms may consist of lines of text or decoration and fields. Fields are enclosed in braces and contain the name of the field.

The space allotted for a field in a form should be larger than or equal to the space alloted in the record. Data can be entered up to the smaller of the two allotments.

Ideally, the first line of a form is left blank to permit space for generated messages.

New Files

When creating a new file, records are normally placed immediately after the record definition. A line containing ***< here >*** can be put in the file to mark where the first record should be placed.

Records can only be added next to other records. If records are to be entered in different groups separated by non-record lines, dummy records must be put in using the Editor. These serve as seeds from which the groups can grow.

The CHANGED Field

If one of the fields in the record is named ***changed*** or ***CHANGED,*** it will be set to the value ***1*** for any record which is changed via the Update command.

Example

Working Area

```
< FORM >

Name: {name                                  }

Room: {room                                  }

< END >
< RECORD >
1 {name              } {room}      {changed}
< END >
1 John Doe              X99
```

Initial Screen Produced by the Update Command

```
Name: John Doe

Room: X99
```

update

Revise Records

		MINE
Insert character	CTRL W	
Delete character	CTRL E	
Erase line	CTRL T	
Delete spaces to next character	CTRL U	
Destructive backspace	DEL	
Erase to end of field	CTRL Y	
Delete record	ESC <	
Insert record	ESC >	

Move a Record

		MINE
Clear the buffer	CTRL C	
Copy to the buffer	CTRL F	
Move to the buffer	CTRL G	
Insert the buffer	CTRL D	

Search

		MINE
Search for string	ESC ,	
Continue search	ESC &	

Change Modes

		MINE
Insert mode	ESC I	
Change form	ESC F	

Note: When a keystroke for the Editor is changed with T/Modify, the corresponding keystroke for the Update command will also be changed.

Move the Cursor

		MINE
Up	CTRL K	
Down	CTRL J	
Left	CTRL H	
Right	CTRL L	
Home	CTRL O	
Next word	CTRL P	
Next field	TAB	
Previous field	ESC TAB	
Next line	RET	

Move the Screen

		MINE
Home	ESC CTRL O	
First record	ESC A	
Previous record	ESC CTRL K	
Next record/ First field	ESC RET	
Next record	ESC CTRL J	
Last record	ESC Z	

Others

		MINE
Quit the update	CTRL Q	
Repeat last keystrokes	CTRL A	
Information	ESC ?	
Redraw the screen	ESC #	

See Addenda for IBM Users keystrokes

update

PC Version

Revise Records

MINE

Insert character ..... Ins ____

Delete character ..... Del ____

Erase line ..... F10 ____

Delete spaces to next character .. Esc Ins ____

Destructive backspace ..... ← ____

Erase to end of field ... Esc Del ____

Delete record ..... Esc F10 ____

Insert record ..... Esc F9 ____

Move a Record

Clear the buffer ..... Esc F8 ____

Copy to the buffer ..... F7 ____

Move to the buffer ..... F8 ____

Insert the buffer .... Esc F7 ____

Search

Search for string .... F5 ____

Continue search ..... F6 ____

Change Modes

Insert mode ..... Esc I ____

Change form ..... Esc F ____

Move the Cursor

MINE

Up ..... ↑ ____

Down ..... ↓ ____

Left ..... ← ____

Right ..... → ____

Home ..... Home ____

Next word ..... End ____

Next field ..... ←→ ____

Previous field ..... Esc ←→ ____

Next line ..... ↵ ____

Move the Screen

Home ..... Esc Home ____

First record ..... Esc ↑ ____

Previous record ..... PgUp ____

Next record/ First field ..... Esc ↵ ____

Next record ..... PgDn ____

Last record ..... Esc ↓ ____

Others

Quit update ..... F1 ____

Repeat last keystrokes ..... Ctrl K ____

Information ..... F2 ____

Redraw the screen .. Esc F2 ____

Note: When a keystroke for the Editor is changed with T/Modify, the corresponding keystroke for the Update command will also be changed.

order

The Order command organizes the records in the working area alphabetically or numerically in either ascending or descending order. Records are ordered on the basis of a field in the record.

WHAT NEXT? ***order*** *[options, if any] field-name*

General

Any lines in the file which are not a part of records form barriers. Records are never moved across such barriers (see Example below).

Options

Select one or none from each of the two following categories.

ascending	ascending order
descending	descending order
characters	orders characters alphabetically
numbers	orders numerically

Options may be abbreviated to one or more letters provided the given spelling is not also the name of the field (e.g., if a field is named "d", "descending" can be abbreviated to "de").

Defaults

If ***ascending*** or ***descending*** is not specified, ***ascending*** is assumed.

If ***numbers*** or ***characters*** is not specified, ***numbers*** is assumed if the field used as the basis of the ordering is right-justified. Otherwise, ***characters*** is assumed.

Note: To order on multiple fields, order first on the least important field, then the next most important, etc.

Example

Working Area (records are unshaded)

```
< RECORD >
+{name< }        {age > }
< END >

Males
+ Bob           27
+ Tom           32

Females
+ Alice         23
+ Susan         27
+ Mary          44
```

Result of: ***order d age***

```
< RECORD >
+{name< }        {age > }
< END >

Males
+ Tom           32
+ Bob           27

Females
+ Mary          44
+ Susan         27
+ Alice         23
```

set

The Set command sets fields in a record to values which may be alphabetic or calculated from other fields in the record.

WHAT NEXT? ***set***

Rules Definition

The working area must contain a definition of the rules to be applied.

The definition begins with a line containing ***< rules >*** and ends with a line containing ***< end >*** . Rules are placed between the two lines. The definition is located ahead of the corresponding record definition.

A file may contain multiple sets of rules, however, the Set command will impose only the first set of rules encountered.

General Form of a Rule

A rule is written on one line and has the following general form.

field-name [model-number [zero-value]] = expression [***when*** condition] ***end***

The model number is optional and is formed as for the Compute command to control the accuracy and presentation of a resulting figure.

The model number may optionally be followed by a character string used to represent exact zeros.

The condition is also optional. If no condition is present, ***when*** should not be placed on the line.

If the rule is followed by a comment, ***end*** must be placed prior to the start of the comment.

For each record in the file for which the condition is true, the field name specified is set equal to the expression.

A number of rules can be made conditional on the basis of a single condition by the following form.

when condition
rule1
rule2
etc.

Each rule in the above may also have its own specific condition. The condition for the group of rules continues to apply until a similar form is encountered.

To escape from this form, specify a condition which is always true.

when 1 < 2

Expressions and Conditions

Expressions and conditions are formed using normal algebra and logic with the following elements:

a. **Field Names**
Must be spelled exactly as in the record without any imbedded spaces.

b. **Constants**
Must be enclosed in single or double quotes if the constant contains an embedded space, a single or double quote, a special character, or if the constant is the spelling of a field name or operator.

c. **Operators**
Assuming no parentheses, operators are evaluated in the precedence order shown below. Operators in the same group have equal precedence.

!	concatenation
*	multiplication
/	division
+	addition
–	subtraction
<	less than
<=	less than or equal to
=	equal to
< >	not equal to
> =	greater than or equal to
>	greater than

set

contains	e.g., "the" contains "h"
within	e.g., "h" within "the"
among	e.g., "Single" among "Married Single"
not	logical not
and	logical and
or	logical or

d. **Parentheses**
May be used to override normal precedence and should be used in any confusing case.

Evaluation of:

< <= = <> >= >

When any of the above operators are used, the evaluation may be made as a character comparison or a numeric comparison. A numeric comparison is made only if both fields are right-justified or numeric constants (i.e., not enclosed in quotes and beginning like a number).

Otherwise a character comparison is made.

This convention can be overridden by preceding the operator with the keyword ***numeric*** or ***char*** to denote the desired type of test.

Note that the following conditions are both true.

0 char **< >** 0.0 0 numeric = 0.0

Truth and Falsehood

A value by itself is true if it converts to a non-zero number, otherwise it is false (e.g., "1 and 3" is true).

A logical comparison yields a value of "1" if it is true otherwise "0".

Relative Record Number

The word ***number*** may be used in an expression in place of a field or constant to refer to the relative number of the record in the file (1, 2, 3, etc.).

id = ***number***

Examples

total 99,999.99 " " = quantity * price end
Mathematical assignment using a model number and a zero value of spaces.

category = A when age < = 20 and tickets > 5 end
Character assignment with a condition.

state = "New York" when zip numeric > = 10001 and zip numeric < 10200 end
Character assignment with a condition. Presumably the field zip is shown as left-justified thus "numeric" is needed to ensure the correct comparison.

positives = (answer1 = Y) + (answer2 = Y) + (answer3 = Y) end
A condition returns "1" if true, thus "positives" will be set to the number of answers which are "Y".

status = "ok" when type among "dealer distributor oem" end
"Among" tests for one of a series of possibilities separated by spaces.

price = cost + 1.67 when category not < 3 end
Example use of "not" to negate a particular comparison.

control = ERROR when not (rate > = 2.50 and rate < = 35.00) end
Example use of "not" to negate a condition.

group = 1 when name > E and name < K end
Sets "group" to 1 for names beginning with "F" through "J".

condition = RED when expenses > income − 1000 end
Mathematical operators may also be used in the formation of conditions.

select

The Select command selects particular records and fields from an existing file of records.

WHAT NEXT? ***select*** *disk-file-name or* ***it*** *[options, if any]* ***end***

Selecting from a Disk File

If the working area has no record definition, the definition from the disk file is inserted.

The working area may contain a record definition different from the disk file. Records selected from the disk file will be reorganized according to the new definition.

Selected records are placed:

- after a line containing ***< here >*** ;
- after the record definition.

Selecting IT

When ***it*** is specified, records are selected from those in the working area itself. Lines which are not parts of records are left in place.

When the ***it*** option is used, a ***<continue>*** specification in the working area is ignored.

Options

list	Lists the names of the fields in the record.
from 5	Selects records only from the 5th record forward.
to 25	Selects records up to and including the 25th record.
when condition	Selects records only if the logical condition is true. If used, this option must be the last one. (See Set command for details of logical conditions.)

<CONTINUE> file-name

When selecting from a disk file, the last line of the disk file may contain ***<continue>*** followed by a file name. The named file will be accepted as a continuation of the original. This file may itself be continued.

The Record Definitions of all such linked files must be identical.

Example

Disk File "PERSONS"

```
< RECORD >
1 { name       }
2 { age >} { phone  }
< END >
1 John Doe
2          47 335-6521
1 Mary Ellen
2          43 532-9911
1 Robert Smith
2          55 777-8080
```

Working Area

```
< RECORD >
{ name          }   { phone > }
< END >

Name                Phone number
< HERE >
```

Result of: ***select persons when age > 45 end***

```
< RECORD >
{ name          }   { phone  > }
< END >

Name                Phone number
< HERE >
John Doe                335-6521
Robert Smith            777-8080
```

complete

The Complete command completes records in the working area with information from records in a disk file. A specified field is used to determine which records on disk match those in the working area.

WHAT NEXT? ***complete*** *disk-file-name matching-field-name*

Matching Records

For a record in the working area to match a record on disk:

- the record in the working area must have some non-blank value for the matching field.
- the value for the field in the working area must be identical with the value in the disk record. (**Note:** 05 is not identical with 5)

One record on disk may match none, one, or more than one record in the working area.

Completing Fields

When a match is made, any field in the working area record, which is also in the disk record, will take the value of the disk record.

Fields defined in the working area record, which are not defined in the disk record, will keep their initial values.

If no match is made for a record, the record will be unchanged.

<CONTINUE> file-name

The last line in the disk file may contain ***< continue >*** followed by a file name. The named file will be accepted as a continuation of the original. This file itself may be continued.

Example

Disk File "TITLES"

```
<RECORD>
+ {project    }    {title          }
<END>
+ ROAD          Road Constr.
+ HOUSE         House Constr.
+ BRIDGE        Bridge Constr.
+ PAVE          Street Paving

<CONTINUE> titles.2
```

Working Area

```
<RECORD>
1{name    }{project }{title    }
<END>
1 Al        PAVE
1 Mary      HOUSE
1 Peter     STAND
1 Tom       PAVE
```

Result of:
complete titles project

```
<RECORD>
1{name    }{project }{title          }
<END>
1 Al        PAVE      Street Paving
1 Mary      HOUSE     House Constr.
1 Peter     STAND
1 Tom       PAVE      Street Paving
```

group

The Group command groups together records in the working area. A group is formed of records which have identical values for the fields specified.

WHAT NEXT? ***group [list]*** *[field-name 1, field-name 2, etc.]* ***end***

General

The command ***group end*** would group all records into one record.

Lines which are not a part of records are left in place.

The List Option

The List option lists the names of the fields in the record.

The Grouped Records

For each group, one record results.

For any fields which are left-justified or centered, the resulting group record will contain the values of the first record encountered for the group.

For any fields which are right-justified (and not used as a basis for defining groups), the resulting group record will contain the sum of the values for all records in the group.

The COUNT Field

If a field in the record has the name ***count*** or ***COUNT,*** it will be set to one for any record which has a blank value for the field.

When records are grouped, the ***COUNT*** field will be summed like any other right-justified field. Thus, this field can be used to count the number of records in each group.

Example

Working Area

```
< RECORD >
+ {name  } {dept } {age > } {count }
  END
+ Bob      PERS      27
+ Tom      FIN       32
+ Alice    PERS      23
+ Susan    FIN       27
+ Mary     FIN       44
```

Result of: ***group dept end***

```
< RECORD >
+ {name  } { dept } { age > } {count }
  END
+ Bob      PERS      50        2
+ Tom      FIN       103       3
```

error messages

General Errors

Most error messages generated by the Data-Base commands are self-explanatory. Care should be taken to spell field names correctly, since an incorrectly spelled name may be treated as a constant and not diagnosed. A few non-trivial problems are described below.

(ALMOST) OUT OF SPACE

Data-Base commands will complain about running out of space if there is insufficient space for one more complete record (in addition to the minimum required for other commands).

PROBLEM IS TOO COMPLEX

The combination of the definitions in use by the command (i.e. record, form, and rules) are too complex. To reduce the complexity of the problem:

- keep identifying labels and field names short
- reduce the length of fields to a minimum
- eliminate any unused fields entirely from the record

If the Set command is the problem, consider breaking the rules into two groups and using Set once for each group.

*INVALID EXPRESSION**

An expression or condition is improperly formed. Check the spellings of field names and operators as well as their placement within the expression.

*UNMATCHED PARENTHESES**

Left and right parentheses do not match in an expression or condition.

*TOO MANY EX- OR IMPLICIT PARENTHESES**

An expression or condition generates too many nested parentheses. For example:

A + B + C + D + E + F

is implicity equivalent to

((((A + B) + C) + D) + E) + F

If there were 20 pluses instead of just 5, this would be a problem. An explicit and more balanced use of parentheses would be:

((A + B) + (C + D)) + (E + F)

. . . VALUES TRUNCATED

Some values were larger than the fields intended to hold them. Such values are shortened on the right in order to fit.

NEED FEWER RECORDS OR SHORTER FIELDS

There are too many unique groups for the Group command. Grouping on fewer or shorter fields or grouping a fewer number of records may solve the problem.

INADEQUATE SPACE FOR SUMMED VALUES

While using the Group command, the summations of some field produced numbers which did not fit in the space provided. The problematic fields will be filled with asterisks.

RECORDS TOO BIG TO ORDER

The Order command may not work with unusually large records.

MINOR ERRORS

The Set command may produce the same types of minor mathematical errors as the Compute command.

*When using the Set command, the top line on the screen is the offending rule.

proof

The Proof command checks the working area for any words which are not in the T/Maker Dictionary or other specified dictionaries.

WHAT NEXT? ***proof [with** dictionary-1 **with** dictionary-2 etc.]*

The full command should be specified initially since prompts will not be made for optional dictionaries. ***Proof with,*** however, will result in a prompt for the dictionary name.

Dictionaries

Dictionaries are special compressed files prepared by the Amend command. They may not be edited.

Proof will always use the standard T/Maker dictionary (i.e., ***T MAKER.DIC),*** if present. For additional dictionaries, ***with*** should precede each name. The file-type ***.DIC*** will be appended to dictionary names specified without a file-type.

All dictionaries are expected to reside on the same drive as the command file ***PROOF.TMK.*** For optional dictionaries, a drive specification may be given.

proof with b:acronyms

Word Lists

A simple list of words may be used in place of a dictionary. However, the list must be in alphabetical order. To prepare such a list, all characters should be typed in the same case (all upper or all lower case). Then, the Sort command can be used to put the list in proper alphabetical order.

Definition of a Word

A word is a series of alphabetic characters ("a" through "z") that may include an apostrophe within it. However, if a word ends with an apostrophe followed by an "s," both the apostrophe and the "s" are not considered part of the word.

Capitalization

No distinction is made between upper and lower case when searching for words in the dictionary.

The Results of *Proof*

Proof prepares a "correction list" of words which were not found in the dictionaries used. Each line in this list begins with the characters *=!* followed by two occurrences of the word.

The first occurrence may be thought of as the current spelling of the word in the file. The second occurrence may be thought of as the spelling to be corrected and later substituted.

Proof also marks unidentified words in the text by preceding them with the characters **?=.**

Example

Working Area

```
Itt is now tme for all
good men to come to the aid
of their prty.
```

Result of: *proof*

```
=!  Itt          Itt
=!  prty         prty
=!  tme          tme

?=Itt is now ?=tme for all
good men to come to the aid
of their ?=prty.
```

spell

The Spell command lists words from a dictionary on the screen or places them on correction lines.

WHAT NEXT? ***spell*** *[**it** or ambiguous-word-reference]*
*[**with** dictionary-name]*

An optional dictionary may be used; however, only the specified dictionary will be examined. The default dictionary is TMAKER.DIC.

Ambiguous Word References

An ambiguous word reference is composed of letters, question marks and asterisks. A question mark is used to represent one unknown letter. An asterisk is used to represent zero or more unknown letters. Thus, the ambiguous reference below would refer to all words that start with "b", have any second letter, have "m" as the third letter, and end with "ing".

b?m*ing

Ambiguous word references should be given as completely as possible to limit the range of the dictionary that needs to be searched.

When the Spell command is followed by an ambiguous word reference, words corresponding to the reference are listed on the screen.

The *it* Option

When ***it*** is specified instead of a word reference, the Spell command searches the working area for correction lines beginning with "=!" that also contain ambiguous word references.

For these correction lines, corresponding words in the dictionary are put on the line after the ambiguous reference until the line is full.

You can construct such correction lines from scratch or alter the second word of a correction line generated by ***Proof*** to look up words in a dictionary.

Example

Working Area

```
=! mi*pell
=! compatable      compat*
```

Result of: *spell it*

```
=! mi*pell    misspell
=! compatable compat* compatibility compatible
```

correct

The Correct command corrects the working area based on the "correction list."

WHAT NEXT? ***correct***

Preparing for *Correct*

After using the Proof command. the working area will contain a correction list of any unidentified words (see leftmost screen below).

You may react in one of four ways to each word in this list.

1. The word is an obvious misspelling. (See case of "compny.")

   You correct the spelling of the second occurrence of the word on the correction line.

   The Correct command subseqently substitutes the second spelling of the word for the first spelling. The correction line is deleted from the file and any corresponding occurrences of **?=** in the text are removed.

   Note: If you are unsure of the correct spelling, the second occurrence of the word may be replaced by an ambiguous word reference. (See Spell command.) The Spell command may then be used to suggest alternatives.

2. The word is correct but should not be added to the dictionary. (See case of "Isuzu".)

   You leave the correction line unchanged or delete it entirely.

   The Correct command deletes the correction line (if you did not) and removes corresponding occurrences of **?=** in the text.

3. The word is correct and should be added to the dictionary. (See case of "quadraphonic.")

   You change the first position of the correction line from = to +.

   The Correct command will remove corresponding occurrences of **?=** from the text; however, the correction line will be left in the file. A subsequent use of the Amend command will add the word to the dictionary and remove the correction line.

4. The correctness of the word is impossible to determine without looking at the word in context. (See case of "rd.")

   You can search for "**?=**" in the file to find the context and correct the problem with the Editor. To correct words which fell into the above cases before attacking those in this case, the second occurrence of the word in the correction list should be erased. This ensures that ***Correct*** will take no action for this word.

correct

Capitalization

Correct attempts to preserve capitalization in cases where one spelling is substituted for another. ***Correct*** differentiates four types of capitalization:

1. all in capitals
2. all in lower case
3. all in lower case except the first letter
4. none of the above

For the first three cases, ***Correct*** will mimic the capitalization of the word in the text when replacing it with another spelling. For the fourth case, ***Correct*** will use the exact spelling, including capitalization, as it is given on the correction line. Thus, when substituting for words with unusual capitalization, you should specify the exact spelling for case four.

= ! MicroMan MicroPerson

If only a single form of capitalization is to be used as the new spelling, then the second word in the correction line should be enclosed in single or double quotes.

= ! company "IBM"

If substitutions are to be made for only a single form of capitalization, then the first word in the correction line should be enclosed in single or double quotes. The second word in the correction line will be substituted exactly as it is entered (including capitalization).

= ! "JONES" MCBRIDE
= ! "Jones" McBride

Substituting Phrases

A phrase may be substituted for a word by placing the phrase appropriately on the correction line.

=! us the company

Note: ***Correct*** may be used to replace words in a text file independently of other spelling commands by manually preparing the correction lines.

Note: ***Correct*** will not substitute spellings when a word on a correction line appears to be an ambiguous word reference. (See Spell command.)

Example

Result of: *proof*

```
=! compny          compny
=! lsuzu           lsuzu
=! quadraphonic    quadraphonic
=! rd              rd
```

Our ?=compny is considering the purchase of an ?=lsuzu with ?=quadraphonic sound in the 3?=rd quarter.

Result of: *editing*

```
=! compny          company
=! lsuzu           lsuzu
+! quadraphonic    quadraphonic
=! rd
```

Our ?=compny is considering the purchase of an ?=lsuzu with ?=quadraphonic sound in the 3?=rd quarter.

Result of: *correct*

```
+! quadraphonic  quadraphonic
=! rd
```

Our company is considering the purchase of an lsuzu with quadraphonic sound in the 3?=rd quarter.

amend

The Amend command creates or updates a dictionary from a list of corrections in the working area and other dictionaries.

WHAT NEXT? ***amend*** *[**from** dictionary-name* + or − *dictionary-name] [**to** dictionary-name]*

All options should be specified in advance.

The Correction List

Correction lines in the working area beginning with "*+!*" denote words to be added to the dictionary.

Correction lines in the working area beginning with "*−!*" denote words to be deleted from the dictionary.

Lines of the above form will be removed from the working area after the dictionary has been amended.

Forms of *Amend*

Amend may be used in any of the following forms to achieve the described result. Dictionary names may be preceded by a drive designation (e.g., ***b:master***). They are otherwise expected to reside on the same drive as the file ***AMEND.TMK.***

amend
amends the dictionary TMAKER.DIC based on the correction list.

amend from temp
amends the dictionary TEMP.DIC based on the correction list.

amend to temp
creates the dictionary TEMP.DIC based on the correction list.

amend from x to y
creates the dictionary Y.DIC based on the correction list and X.DIC.

amend from x + y to z
creates the dictionary Z.DIC based on the correction list (if any) and the combination of the dictionaries X.DIC and Y.DIC.

amend from x − y to z
creates the dictionary Z.DIC by applying the correction list (if any) to the words in X.DIC which are not in Y.DIC.

When a dictionary is modified or created, the previous version is kept as a file with the file-type of ".BAK".

Saving Time

Amending a lengthy dictionary may take a number of minutes. The following procedure will save a considerable amount of time.

For the first amendments of the day use the command: ***amend to temp***

For later amendments use the command: ***amend from temp***

At the end of the day, use the command: ***amend from temp + tmaker to tmaker***

During the day use: ***proof with temp***

T/MAKER Company
2115 Landings Drive
Mountain View, CA 94043