Introduction

The ALTER procedure is intended to be used as a replacement for the EDIT/3000 MODIFY command. The principal features are an expanded set of flexible editing commands and the capability to perform multiple editing operations from the same command line. The ALTER procedure has been implemented through the PROCEDURE command of EDIT/3000.

Definitions used in this document

Original line: the text line as passed to the ALTER procedure from EDIT/3000.

ALTER line: the copy of the original line to which editing operations are applied.

Final line: the text line, with alterations, as passed back to EDIT/3000.

Command line: a line of ALTER commands.

HOLD string: a space that can hold a string of characters and is used in a manner similar to the HOLD file of EDIT/3000.

Z string: a space that can hold a sequence of ALTER commands and is used in a manner similar to Z:: of EDIT/3000.

Pointer: a pointer that moves through the ALTER line as the result of the ALTER commands. The action of most commands commences at the current pointer position.

Break character: a character used to separate operands and commands in the command line.

General operating procedures

When altering a line, as many commands may be used in a single line as desired. A break character (typically a control G) is used to separate the operand of one command from the subsequent command. Alteration of a line is terminated by entering a null command line.

Any numbers associated with a command are optional, if a number is not specified, the default value (typically one) is used.

If a command is incorrectly specified or its use would cause some limit to be exceeded, all subsequent commands in the same command line are ignored and an error message is printed that indicates the point in the command line at which the error was detected. If the user is not accessing the ALTER procedure interactively, all subsequent commands pertaining to the current line are ignored.

When accessing the ALTER procedure in a non-interactive mode (typically from a JOB) all trailing blanks are stripped from the command line. In this mode of operation a break character should be inserted after any desired trailing blanks to ensure their inclusion in the command line.

Limitations of the ALTER procedure

Maximum line length is 256 characters.

Maximum command line length is 132 characters.

Maximum HOLD string length is 132 characters.

Maximum Z string length is 72 characters.

No number in a command may exceed 256.

Syntax notation

The elements of the ALTER syntax are described below. Each element is enclosed in "<>", followed by its definition and a list of commands in which the element appears.

<brk-chr> - Any ASCII character; the initial value is control G(BELL). <brk-chr> is used to separate data from subsequent commands.

Commands: INSERT, JOIN, LOCATE, MUNCH, REPLACE, SUBSTITUTE, EXTEND, Z=

<character> - Any ASCII character except <brk-chr>.

Commands: ADD, BREAK, CHANGE, FIND, GRAB, KILL, MUNCH

<string> - A sequence of <character>s to be used for input, comparison, or commands.

Commands: ADD, CHANGE, INSERT, JOIN, MUNCH, REPLACE, SUBSTITUTE, EXTEND, LOCATE, Z=

<c> - An unsigned integer that indicates how many copies of the specified characters are to be manipulated.

Commands: ADD, CHANGE, INSERT, JOIN, MUNCH, REPLACE, SUBSTITUTE, EXTEND, HOLD

<n> - An unsigned integer that indicates the number of characters affected by the command, the number of times to repeat the command, or the particular occurrence of a character or string.

Commands: DELETE, ERASE, HOLD, QUASH, SUBSTITUTE, INVERT, UNDO, Z, SPACE, FIND, GRAB, KILL, LOCATE, MUNCH

<w> - An unsigned integer which represents a field or line width.

Commands: JOIN, OUTLINE, QUASH, WIDTH

<column> - An unsigned integer which represents a column position in the ALTER line.

Commands: OUTLINE, POSITION

- indicates that input for the command will come from the HOLD string rather than from a <string> following the command, or that characters are to be appended to the HOLD string.

Commands: INSERT, JOIN, LOCATE, MUNCH, REPLACE, SUBSTITUTE, EXTEND, HOLD

* - indicates that the string referenced in the last GRAB, HOLD, or LOCATE command is to be affected by the command, or that the length of the string is to be used as a field width.

Commands: DELETE, ERASE, HOLD, SPACE, SUBSTITUTE, INVERT, JOIN, QUASH

Elements enclosed in "[]" are optional and may be omitted if the default value for the element is desired.

Example notation

The examples that follow each command are delimited by "!". A "U" at the beginning of the line indicates a user input line, comments follow the right-hand "!". Blank lines are used for purposes of clarity only, and are not generated by the actual ALTER procedure.

For demonstrative purposes, the break character used in the examples is "/".

Implementation notes

The ALTER procedure is a single SPL procedure named "M" contained in the file ITELEDIT.SOURCE.LIB. The procedure should be compiled and added to an appropriate SL, typically the system SL.

Accessing the procedure from EDIT/3000 is done in the following manner:

```
/P M,{G P S} <rangelist>
```

The first parameter is the procedure name "M", the second parameter (G, P, or S) indicates the sequence of SLs that will be searched for the procedure, and the third parameter is a range list specifying which lines will be affected.

ADD

Purpose: Add characters to the line.

Syntax:

Form 1 - [<c>] A <character>

Description:

Form 1 - Add <c> copies of <character> in front of the current pointer position.

Defaults - <c> defaults to 1.

Pointer - The pointer is positioned after the last character added.

Comments:

ADD is typically used to add a single character into the line.

Examples:

```
U!  /P M,S,1                              !
 !  ALTER     1                           !
 !  THIS IS THE SAMPLE LINE.              !
U!                      AS                ! 1
 !  THIS IS THE SAMPLES LINE.             !
U!  3A$                                   ! 2
 !  $$$THIS IS THE SAMPLES LINE.          !
U!                                        ! 3
 !  /                                     !
```

1 - Add an "S" in front of the space between "SAMPLE" and "LINE".

2 - Add three "$"s in front of the first character of the line.

3 - End of alteration.

BREAK

Purpose: Change the definition of <brk-chr>.

Syntax:

Form 1 - B <character>

Description:

Form 1 - Use <character> as the ALTER <brk-chr>.

Defaults -

Pointer - The pointer is unaffected by this command.

Comments:

<brk-chr> is initially set to a control G.

The break character can be changed to a character that is more convenient to type than control G. It should be set to a character that does not appear in the text since the ALTER procedure can not differentiate between the break character used as text or as a break.

Examples:

```
U!  /P M,S,1                              !
 !  ALTER     1                           !
 !  THIS IS THE SAMPLE LINE.              !
U!  B/                                    ! 1
 !  THIS IS THE SAMPLE LINE.              !
U!  IHERE, /F.I??                         ! 2
 !  HERE, THIS IS THE SAMPLE LINE??.      !
U!                                        ! 3
 !  /                                     !
```

1 - Change the break character to be a "/".

2 - Insert the string "HERE, " in front of the first character of the line, find the next occurrence of "." and insert "??" in front of it. Note that a break character was used to separate the string of the first INSERT command from the FIND command, but that no break character was required after the second INSERT command since there are no subsequent commands on that line.

3 - End of alteration.

CHANGE

Purpose: Change characters in the line.

Syntax:

Form 1 - [<c>] C <character>

Description:

Form 1 - Replace characters of the line starting at the current pointer position with <c> copies of <character>.

Defaults - <c> defaults to 1.

Pointer - The pointer is positioned after the last character changed.

Comments:

CHANGE is typically used to replace a single character in the line.

Examples:

```
U!  /P M,S,1                          !
 !  ALTER    1                        !
 !  THIS IS THE SAMPLE LINE.          !
U!  FACI                              ! 1
 !  THIS IS THE SIMPLE LINE.          !
U!  F.2C??                            ! 2
 !  THIS IS THE SIMPLE LINE??         !
U!                                    ! 3
 !  /                                 !
```

1 - Find the first occurrence of an "A" and change it to an "I".

2 - Find the first occurrence of a "." and change that character and the next to "??".

3 - End of alteration.

DELETE

Purpose: Delete characters from the line.

Syntax:

Form 1 - [<n>] D

Form 2 - * D

Description:

Form 1 - Delete the next <n> characters starting at the current pointer position.

Form 2 - Delete the string recorded by the immediately preceding GRAB, HOLD, or LOCATE command.

Defaults - <n> defaults to 1.

Pointer - The pointer is positioned after the last character deleted.

Comments:

If the original pointer position is less than <n> characters from the end of the line, only the characters through the end of the line are deleted and the pointer is positioned after the last character of the line.

Examples:

```
U!  /P M,S,1                          !
 !  ALTER     1                       !
 !  THIS IS THE SAMPLE LINE.          !
U!  3FS7D                             ! 1
 !  THIS IS THE LINE.                 !
U!  2LIS /*D                          ! 2
 !  THIS THE LINE.                    !
U!                                    ! 3
 !  /                                 !
```

1 - Find the third "S" in the line and delete seven characters starting at that point.

2 - Locate the second occurrence of the string "IS " and delete it.

3 - End of alteration.

ERASE

Purpose: Erase (change to blank) characters in the line.

Syntax:

Form 1 - [<n>] E

Form 2 - * E

Description:

Form 1 - Erase the next <n> characters starting at the current pointer position.

Form 2 - Erase the string recorded in the immediately preceding GRAB, HOLD, or LOCATE command.

Defaults - <n> defaults to 1.

Pointer - The pointer is positioned after the last character erased.

Comments:

If the original pointer position is less than <n> characters from the end of the line, only the characters through the end of the line are erased and the pointer is positioned after the last character erased.

Examples:

```
U!  /P M,S,1                              !
 !  ALTER     1                           !
 !  THIS IS THE SAMPLE LINE.              !
U!  2FI2E                                 ! 1
 !  THIS    THE SAMPLE LINE.              !
U!  2LTH/*E                               ! 2
 !  THIS      E SAMPLE LINE.              !
U!                                        ! 3
 !  /                                     !
```

1 - Find the second occurrence of "I" in the line and erase that character and the next.

2 - Locate the second occurrence of "TH" in the line and erase it.

3 - End of alteration.

FIND

Purpose: Position the pointer at a specific character.

Syntax:

Form 1 - [<n>] F <character>

Description:

Form 1 - Find the <n>th occurrence of <character> starting from the current pointer position.

Defaults - <n> defaults to 1.

Pointer - The pointer is positioned at the <n>th occurrence of <character>.

Failure - The command fails if the <n>th occurence of <character> is not found.

Comments:

Examples:

```
U!  /P M,S,1                              !
 !  ALTER    1                            !
 !  THIS IS THE SAMPLE LINE.              !
U!  3FECT                                 ! 1
 !  THIS IS THE SAMPLE LINT.              !
U!  4FSIXXX                               ! 2
 !     ^                                  !
 !  *** ERROR *** COMMAND FAILED          !
 !  THIS IS THE SAMPLE LINT.              !
U!                                        ! 3
 !  /                                     !
```

1 - Find the third "E" in the line and change it to a "T".

2 - Finding the fourth "S" will fail since there are only three "S"s in the line. The insert command is not executed.

3 - End of alteration.

GRAB

Purpose: Record the position and length of a string.

Syntax:

Form 1 - [<n>] G <character>

Description:

Form 1 - Record the position and length of the string from the current pointer position up to, but not including, the <n>th occurrence of <character>. This command does not change the line, but the information it records can be used by subsequent commands to alter the line.

Defaults - <n> defaults to 1.

Pointer - The pointer is unaffected by this command.

Failure - The command fails if the <n>th occurrence of <character> is not found.

Comments:

The GRAB command is used to specifiy a string for use by the DELETE, ERASE, HOLD, SPACE, SUBSTITUTE, or INVERT commands; or to define a field width for the JOIN and QUASH commands.

Examples:

```
U!  /P M,S,1                          !
 !  ALTER    1                        !
 !  THIS IS THE SAMPLE LINE.          !
U!  3FSG *V                           ! 1
 !  THIS IS THE sample LINE.          !
U!                                    ! 2
 !  /                                 !
```

1 - Find the third "S" in the line, grab all characters up to the next " " in the line and down shift the string recorded.

2 - End of alteration.

HOLD

Purpose: Place characters in a HOLD string from which they may later be retrieved.

Syntax:

Form 1 - [<n>] [, <c>] H

Form 2 - [<n>] # [<c>] H

Form 3 - * [<c>] H

Form 4 - * # [<c>] H

Description:

Form 1 - Place <c> copies of the <n> characters which begin at the current pointer position into the HOLD string.

Form 2 - Append <c> copies of the <n> characters which begin at the current pointer position to the contents of the HOLD string.

Form 3 - Place <c> copies of the string recorded by the preceding GRAB, HOLD, or LOCATE command into the HOLD string.

Form 4 - Append <c> copies of the string recorded by the immediately preceding GRAB, HOLD, or LOCATE command to the contents of the HOLD string.

Defaults - <n> and <c> default to 1.

Pointer - The pointer is unaffected by this command.

Comments:

Note the difference between Form 1(3) and Form 2(4): in Form 1(3) the designated string replaces whatever was previously contained in the HOLD string; whereas in Form 2(4) the designated string is appended to the existing contents of the HOLD string.

If the original pointer position is less than <n> characters from the end of the line, only the characters through the end of the line are held.

The contents of the HOLD string are retained between calls to the ALTER procedure.

The length of the HOLD string is limited to 132 characters.

The HOLD string is used in a manner similar to the hold file of EDIT/3000, but the two areas are not related.

Examples:

```
U!  /P M,S,1                          !
 !  ALTER     1                       !
 !  THIS IS THE SAMPLE LINE.          !
U!  2LIS /*H*D                        ! 1
 !  THIS THE SAMPLE LINE.             !
U!  #I                                ! 2
 !  IS THIS THE SAMPLE LINE.          !
U!  L THIS/*#H                        ! 3
 !  IS THIS THE SAMPLE LINE.          !
U!  8DF.#I                            ! 4
 !  THE SAMPLE LINE IS THIS.          !
U!                                    ! 5
 !  /                                 !
```

1 - Locate the second occurrence of "IS " in the line, place it in the HOLD string and then delete it from the line.

2 - Insert the HOLD string in front of the first character of the line.

3 - Locate the first occurrence of " THIS" and append it to the HOLD string.

4 - Delete the first eight characters of the line, find the "." and insert the HOLD string in front of it.

INSERT

Purpose: Insert characters into the line.

Syntax:

Form 1 - [<c>] I <string> [<brk-chr>]

Form 2 - # [<c>] I

Description:

Form 1 - Insert <c> copies of <string> in front of the current pointer position. <string> must be followed by a <brk-chr> if additional commands follow on the same command line.

Form 2 - Insert <c> copies of the HOLD string in front of the current pointer position.

Defaults - <c> defaults to 1.

Pointer - The pointer is positioned after the last character inserted.

Comments:

INSERT is used to add an indefinite number of characters into the line.

Examples:

```
U!  /P M,S,1                          !
 !  ALTER    1                        !
 !  THIS IS THE SAMPLE LINE.          !
U!  IIS /2FI3DF.I NOW                 ! 1
 !  IS THIS THE SAMPLE LINE NOW.      !
U!                                    ! 2
 !  /                                 !
```

1 - Insert "IS " in front of the first character of the line, find the second occurrence of "I" (starting after the inserted characters), delete three characters, find the "." and insert " NOW". Note the use of the "/" to separate the string of the first INSERT command and the subsequent FIND command.

2 - End of alteration.

JOIN

Purpose: Insert characters into a field without affecting the portion of the line to the right of the field.

Syntax:

Form 1 - [<w>] [, <c>] J <string> [<brk-chr>]

Form 2 - * [<c>] J <string> [<brk-chr>]

Form 3 - * # [<c>] J

Description:

Form 1 - Within a field which begins at the current pointer position and has width <w>, join <c> copies of <string> to the front of the field. <string> must be followed by a <brk-chr> if additional commands follow on the same command line.

Form 2 - Within a field which begins at the current pointer position and has a width equal to the length of the string recorded by the last GRAB, HOLD, or LOCATE command, join <c> copies of <string> to the front of the field. <string> must be followed by a <brk-chr> if additional commands follow on the same command line.

Form 3 - Within a field which begins at the current pointer position and has a width equal to the length of the string recorded by the last GRAB, HOLD, or LOCATE command, join <c> copies of the HOLD string to the front of the field.

Any characters that would be moved past the end of the field because of the insertion will be deleted.

Defaults - <w> and <c> default to 1.

Pointer - The pointer is positioned after the last character inserted.

Comments:

If the width of the field is not known, the GRAB command can be used to establish the field width. Examples:

```
U!  /P M,S,2                                  !
 !  ALTER     2                               !
 !  SMITH           ,ANN        ,F,29         !
U!  FAG,*JMARY                                ! 1
 !  SMITH           ,MARYANN    ,F,29         !
U!                                            ! 2
 !  /                                         !
```

1 - Find the first "A" in the line, grab up to the next "," and insert "MARY" at the front of the field recorded without affecting the portion of the line to the right of the ",".

2 - End of alteration.

KILL

Purpose: Delete characters up to a specific character.

Syntax:

Form 1 - [<n>] K <character>

Description:

Form 1 - Delete all characters from the current pointer position up to but not including the <n>th occurrence of <character>.

Defaults - <n> defaults to 1.

Pointer - The pointer is positioned at the <n>th occurrence of <character>.

Failure - The command fails if the <n>th occurrence of <character> is not found.

Comments:

Examples:

```
U!  /P M,S,1                          !
 !  ALTER     1                       !
 !  THIS IS THE SAMPLE LINE.          !
U!  2KI                               ! 1
 !  IS THE SAMPLE LINE.               !
U!                                    ! 2
 !  /                                 !
```

1 - Delete all characters up to but not including the second occurence of "I" in the line.

2 - End of alteration.

LOCATE

Purpose: Locate a string of characters.

Syntax:

Form 1 - [<n>] L <string> [<brk-chr>]

Form 2 - [<n>] # L

Description:

Form 1 - Locate the <n>th occurrence of <string> starting from the current pointer position. <string> must be followed by a <brk-chr> if additional commands follow on the same command line.

Form 2 - Locate the <n>th occurrence of the HOLD string starting from the current pointer position.

Defaults - <n> defaults to 1.

Pointer - The pointer is positioned at the first character of the <n>th occurrence of <string> or the HOLD string.

Failure - The command fails if the <n>th occurrence of <string> or the HOLD string is not found.

Comments:

Examples:

```
U!  /P M,S,1                          !
 !  ALTER     1                       !
 !  THIS IS THE SAMPLE LINE.          !
U!  LLINE/*E                          ! 1
 !  THIS IS THE SAMPLE     .          !
U!                                    ! 2
 !  /                                 !
```

1 - Locate the first occurrence of "LINE" in the line and erase it.

2 - End of alteration.

MUNCH

Purpose: Replace characters up to a specific character.

Syntax:

Form 1 - [<n>] [, <c>] M <character> <string> [<brk-chr>]

Form 2 - [<n>] # [<c>] M <character>

Description:

Form 1 - Replace all characters from the current pointer position up to but not including the <n>th occurrence of <character> with <c> copies of <string>. <string> must be followed by a <brk-chr> if additional commands follow on the same command line.

Form 2 - Replace all characters from the current pointer position up to but not including the <n>th occurrence of <character> with <c> copies of the HOLD string.

Defaults - <n> and <c> default to 1.

Pointer - The pointer is positioned after the last character replaced.

Failure - The command fails if the <n>th occurrence of <character> is not found.

Comments:

MUNCH is equivalent to a KILL command followed by an INSERT.

Examples:

```
U!  /P M,S,1                                  !
 !  ALTER    1                                !
 !  THIS IS THE SAMPLE LINE.                  !
U!  3FS2MLHEAD                                ! 1
 !  THIS IS THE HEADLINE.                     !
U!  3,3MHEXTRA                                ! 2
 !  EXTRA EXTRA EXTRA HEADLINE.               !
U!  /                                         !
```

1 - Find the third "S" in the line, replace all characters up to but not including the second occurrence of "L" with the string "HEAD".

2 - Replace all characters up to but not including the third occurrence of "H" in the line with three copies of the string "EXTRA ".

3 - End of alteration.

OUTLINE

Purpose: Generate a column position template.

Syntax:

Form 1 - [<w>] [, <column>] O

Description:

Form 1 - Generate a column position template <w> columns wide starting in <column>.

Defaults - <w> and <column> default to 1.

Pointer - The pointer is unaffected by this command.

Comments:

The template generated by the OUTLINE command is useful when column dependent editing is being performed.

Examples:

```
U!  /P M,S,1                                  !
 !  ALTER     1                               !
 !  THIS IS THE SAMPLE LINE.                  !
U!  300                                       ! 1
 !          1         2         3             !
 !  123456789012345678901234567890            !
 !  THIS IS THE SAMPLE LINE.                  !
U!                                            ! 2
 !  /                                         !
```

1 - Print a column template thirty columns wide.

2 - End of alteration.

POSITION

Purpose: Position the pointer at a specified column.

Syntax:

Form 1 - <column> P

Description:

Form 1 - Position the pointer at <column>.

Defaults - <column> defaults to 1.

Pointer - The pointer is positioned at <column>.

Comments:

The leftmost column of the ALTER line is column one.

Examples:

```
U!  /P M,S,1                          !
 !  ALTER    1                        !
 !  THIS IS THE SAMPLE LINE.          !
U!  10PIAB/PCX                        ! 1
 !  XHIS IS TABHE SAMPLE LINE.        !
U!                                    ! 2
 !  /                                 !
```

1 - Position the pointer at column ten, insert the string "AB" in front of that column, return the pointer to column one and change that character to an "X".

2 - End of alteration.

QUASH

Purpose: Delete characters from a field without affecting the portion of the line to the right of the field.

Syntax:

Form 1 - [<w>] [, <n>] Q

Form 2 - * [<n>] Q

Description:

Form 1 - Within a field which begins at the current pointer position and has width <w>, delete the first <n> characters.

Form 2 - Within a field which begins at the current pointer position and has a width equal to the length of the string recorded by the last GRAB, HOLD, or LOCATE command, delete the first <n> characters.

Blanks are inserted at the right end of the field to maintain the original field width.

Defaults - <n> and <w> default to 1.

Pointer - The pointer is positioned after the last character deleted.

Comments:

If the width of the field is not known, the GRAB command can be used to establish the field width.

Examples:

```
U!  /P M,S,2                                  !
  !  ALTER      2                             !
  !  SMITH            ,MARYANN    ,F,29      !
U!  2FMG,*4Q                                  ! 1
  !  SMITH            ,ANN        ,F,29      !
U!                                            ! 2
  !  /                                         !
```

1 - Find the second occurrence of "M" in the line, grab all characters up to but not including the next "," and delete four characters from the field recorded without affecting the portion of the line to the right of the ",".

2 - End of alteration.

REPLACE

Purpose: Replace characters.

Syntax:

Form 1 - [<c>] R <string> [<brk-chr>]

Form 2 - # [<c>] R

Description:

Form 1 - Replace characters of the line starting at the current pointer position with <c> copies of <string>. The number of characters replaced is equal to <c> times the length of <string>. <string> must be followed by a <brk-chr> if additional commands follow on the same command line.

Form 2 - Replace characters of the line starting at the current pointer position with <c> copies of the HOLD string. The number of characters replaced is equal to <c> times the length of the HOLD string.

Defaults - <c> defaults to 1.

Pointer - The pointer is positioned after the last character replaced.

Comments:

REPLACE is used to change an indefinite number of characters in the line.

Examples:

```
U!  /P M,S,1                              !
 !  ALTER    1                            !
 !  THIS IS THE SAMPLE LINE.              !
U!  2FIRPRETTY                            ! 1
 !  THIS PRETTY SAMPLE LINE.              !
U!                                        ! 2
 !  /                                     !
```

1 - Find the second occurrence of "I" in the line and replace characters from that point on with "PRETTY".

2 - End of alteration.

SUBSTITUTE

Purpose: Substitute one string for another.

Syntax:

Form 1 - [<n>] [, <c>] S <string> [<brk-chr>]

Form 2 - [<n>] # [<c>] S

Form 3 - * [<c>] S <string> [<brk-chr>]

Form 4 - * # [<c>] S

Description:

Form 1 - Substitute for the next <n> characters starting at the current pointer position <c> copies of <string>. <string> must be followed by a <brk-chr> if additional commands follow on the same command line.

Form 2 - Substitute for the next <n> characters starting at the current pointer position <c> copies of the HOLD string.

Form 3 - Substitute for the string recorded by the last GRAB, HOLD, or LOCATE command <c> copies of string. <string> must be followed by a <brk-chr> if additional commands follow on the same command line.

Form 4 - Substitute for the string recorded by the last GRAB, HOLD, or LOCATE command <c> copies of the HOLD string.

Defaults - <n> and <c> default to 1.

Pointer - The pointer is positioned after the last character inserted.

Comments:

SUBSTITUTE is most often used when replacing a string of characters with <string> which is of different length.

Examples:

```
U!  /P M,S,1                          !
 !  ALTER     1                       !
 !  THIS IS THE SAMPLE LINE.          !
U!  2FT3SA                            ! 1
 !  THIS IS A SAMPLE LINE.            !
U!  2LIS/*SILLUSTRATES                ! 2
 !  THIS ILLUSTRATES A SAMPLE LINE.   !
U!                                    ! 3
 !  /                                 !
```

1 - Find the second occurrence of "T" in the line and substitute the string "A" for the three characters starting at that point.

2 - Locate the second occurrence of the string "IS" in the line and substitute the string "ILLUSTRATES" for it.

3 - End of alteration.

TRANSPOSE

Purpose: Transpose two adjacent characters.

Syntax:

Form 1 - T

Description:

Form 1 - Transpose (reverse the relative positions) of the next two characters starting at the current pointer position.

Pointer - The pointer is positioned after the two transposed characters.

Comments:

Examples:

```
U!  /P M,S,1                              !
 !  ALTER    1                           !
 !  THIS IS THE SAMPLE LINE.             !
U!  FNT                                   ! 1
 !  THIS IS THE SAMPLE LIEN.             !
U!                                        ! 2
 !  /                                    !
```

1 - Find the first occurrence of "N" in the line and transpose it with the next character in the line.

2 - End of alteration.

Syntax:

Form 1 - [<n>] U

Description:

Form 1 - Restore the line to its <n>th previous state. The only values for <n> currently defined are one and two. A value of one causes the ALTER line to be restored to the immediately previous state, a value of two(or any larger value) causes the ALTER line to be restored to the original line.

Defaults - <n> defaults to 1.

Pointer - The pointer is positioned at the beginning of the line.

Comments:

Note that all previous states back to the original are not saved, only the immediately previous state and the original line. Thus, issuing the UNDO command twice in succession is equivalent to performing a '2U' command, the line is restored to its original state.

Examples:

```
U!  /P M,S,1                          !
 !  ALTER     1                       !
 !  THIS IS THE SAMPLE LINE.          !
U!  4V                                ! 1
 !  this IS THE SAMPLE LINE.          !
U!  5D                                ! 2
 !  IS THE SAMPLE LINE.               !
U!  F.S HERE?                         ! 3
 !  IS THE SAMPLE LINE HERE?          !
U!  U                                 ! 4
 !  IS THE SAMPLE LINE.               !
U!  U                                 ! 5
 !  THIS IS THE SAMPLE LINE.          !
U!                                    ! 6
 !  /                                 !
```

1 - Downshift the first four characters of the line.

2 - Delete the first five characters of the line.

3 - Find the first occurrence of a "." and substitute the string " HERE?" for it.

4 - Restore the line to its immediately previous state.

5 - The second UNDO command in succession will restore the line to its original state.

6 - End of alteration.

INVERT

Purpose: Change the case of alphabetic characters.

Syntax:

Form 1 - [<n>] V

Form 2 - * V

Description:

Form 1 - Invert the case of the next <n> characters starting at the current pointer position. Upper case characters are downshifted, lower case characters are upshifted, and numeric and special characters are unaffected.

Form 2 - Invert the case of all characters in the string found in the last GRAB, HOLD, or LOCATE command.

Defaults - <n> defaults to 1.

Pointer - The pointer is positioned after the last character scanned, this is not necessarily the last character actually inverted.

Comments:

If the original pointer position is less than <n> characters from the end of the line, only the characters through the end of the line are inverted and the pointer is positioned after the last character of the line.

Examples:

```
U! /P M,S,1                                  !
 ! ALTER     1                               !
 ! THIS IS THE SAMPLE LINE.                  !
U! 6V                                        ! 1
 ! this iS THE SAMPLE LINE.                  !
U! 7V                                        ! 2
 ! THIS Is THE SAMPLE LINE.                  !
U!                                           ! 3
 ! /                                         !
```

1 - Invert the case of the first six characters in the line. Note that this is the first six of any characters, not the first six alphabetics.

2 - Invert the case of the first seven characters in the line.

WIDTH

Purpose: Specify maximum final line length.

Syntax:

Form 1 - [<w>] W

Description:

Form 1 - Set the maximum final line length to <w>.

Defaults - <w> defaults to 1.

Pointer - The pointer is unaffected by this command.

Comments:

The initial value for <w> is 72.

After alteration of the line is complete, the final line length is checked against <w>. If the final line length exceeds <w> , the user is given three options:

1) The line may be altered further to bring the line length down to the specified <w>.
2) The line may be truncated at <w>.
3) The line may be left as is (i.e., longer than <w>).

If the user is not accessing the ALTER procedure interactively, option 3 is chosen automatically.

EDIT/3000 does not issue a warning if the user creates a line that extends beyond the right margin.

The value of <w> is retained until a subsequent WIDTH command changes it.

Examples:

```
U!  /P M,S,1                              !
 !  ALTER     1                           !
 !  THIS IS THE SAMPLE LINE.              !
U!  15W                                   ! 1
 !  THIS IS THE SAMPLE LINE.              !
U!                                        ! 2
 !  ***WARNING*** LINE LENGTH( 24) > MAX( 15),
 !                ALTER/TRUNCATE/RETURN
U!  *A/T/CR?A                             ! 3
 !  THIS IS THE SAMPLE LINE.              !
U!  72W                                   ! 4
 !  THIS IS THE SAMPLE LINE.              !
U!                                        ! 5
 !  /                                     !
```

1 - Set the maximum final line length to fifteen.

2 - End of alteration.

3 - Since the final line length exceeds the maximum specified, the user is given the options of further altering the line, truncating the line at the maximum final line length, or returning the line to EDIT/3000 as is.

4 - Set the maximum final line length to seventy-two.

5 - End of alteration.

EXTEND

Purpose: Append characters to the end of the line.

Syntax:

Form 1 - [<c>] X <string> [<brk-chr>]

Form 2 - # [<c>] X

Description:

Form 1 - Append <c> copies of <string> to the end of the line. <string> must be followed by a <brk-chr> if additional commands follow on the same command line.

Form 2 - Append <c> copies of the HOLD string to the end of the line.

Defaults - <c> defaults to 1.

Pointer - The pointer is positioned after the last character appended.

Comments:

Examples:

```
U!  /P M,S,1                                  !
  !  ALTER:    1                              !
  !  THIS IS THE SAMPLE LINE.                 !
U!  X NEXT LINE                               ! 1
  !  THIS IS THE SAMPLE LINE. NEXT LINE.      !
U!  U4X /XFILLER                              ! 2
  !  THIS IS THE SAMPLE LINE.    FILLER       !
U!                                            ! 3
  !  /                                        !
```

1 - Extend the line with the string " NEXT LINE".

2 - Restore the line to its previous state and extend the line with four blanks followed by the string "FILLER".

3 - End of alteration.

Z

Purpose: Save and use ALTER commands with the Z string.

Syntax:

Form 1 - Z= <string> [<brk-chr>]

Form 2 - [<n>] Z

Description:

Form 1 - Save the sequence of ALTER commands contained in <string> in the Z string. <string> must be followed by a <brk-chr> if additional commands follow on the same command line.

Form 2 - Execute the commands saved in the Z string <n> times.

Defaults - <n> defaults to 1.

Pointer - Form 1 leaves the pointer unaffected. Form 2 has the same effect as <n> copies of <string> in the command line.

Comments:

The length of the Z string is limited to 72 characters.

The contents of the Z string are retained between calls to the ALTER procedure.

Examples:

```
U!  /P M,S,1                          !
 !  ALTER    1                        !
 !  THIS IS THE SAMPLE LINE.          !
U!  Z=FSCX                            ! 1
 !  THIS IS THE SAMPLE LINE.          !
U!  2Z                                ! 2
 !  THIX IX THE SAMPLE LINE.          !
U!                                    ! 3
U!  /P M,S,1                          ! 4
 !  ALTER    1                        !
 !  THIX IX THE SAMPLE LINE.          !
U!  Z                                 ! 5
 !  THIX IX THE XAMPLE LINE.          !
U!                                    ! 6
 !  /                                 !
```

1 - Assign the string "FSCX" to the Z string. This represents the commands "find next S, change it to an X".

2 - Perform the commands contained in the Z string twice.

3 - End of alteration.

4 - Invoke the ALTER procedure again.

5 - Perform the commands contained in the Z string once. Note that the previous definition is retained.

6 - End of alteration.

SPACE

Purpose: Move the pointer left or right in a line.

Syntax:

Form 1 - [<n>] -

Form 2 - [<n>] <space bar>

Form 3 - * <space bar>

Description:

Form 1 - Position the pointer <n> spaces to the left of the original pointer position.

Form 2 - Position the pointer <n> spaces to the right of the original pointer position.

Form 3 - Position the pointer immediately to the right of the string recorded by the last GRAB, HOLD or LOCATE command.

Defaults - <n> defaults to 1.

Pointer - The pointer is positioned as described above.

Comments:

In form 1, if positioning the pointer would move it past the beginning of the line, the pointer will be positioned at the first character of the line. In form 2, if positioning the pointer would move it past the end of the line, blank characters are appended to the end line to the point where the pointer is positioned.

Examples:

```
U!  /P M,S,1                          !
 !  ALTER    1                       !
 !  THIS IS THE SAMPLE LINE.         !
U!  F.2-T                            ! 1
 !  THIS IS THE SAMPLE LIEN.         !
U!       CT                          ! 2
 !  THIS IT THE SAMPLE LIEN.         !
U!  LSAM/* 2D                        ! 3
 !  THIS IT THE SAME LIEN.           !
U!                                   ! 4
 !  /                                !
```

1 - Find the first occurrence of a "." in the line, space left two positions and transpose the two characters starting at that point.

2 - Space right to the second "S" in the line and change it to a "T".

3 - Locate the first occurrence of the string "SAM", space right over it and delete the next two characters.

4 - End of alteration.

DEBUG

Purpose: Call the intrinsic DEBUG.

Syntax:

Form 1 - ?

Description:

Form 1 - Call the intrinsic DEBUG.

Pointer - The pointer is unaffected by this command.

Comments:

This allows poking around in ALTER and EDIT/3000.

QUOTE

Purpose: Manipulate ASCII control characters in a line.

Syntax:

Form 1 - '

Description:

Form 1 - The special characters flag is toggled each time ' is encountered as a command.

Defaults -

Pointer - The pointer is unaffected by this command.

Comments:

The special characters flag is set to false before each command line is scanned.

When the special characters flag is true, data tokens of the form "'nnn" are converted to a single ASCII character with value nnn. These characters may be manipulated in the same manner as any other characters. When the special characters flag is false, data tokens of the form "'nnn" are treated as separate ASCII characters. "nnn" should be the decimal representation of the desired ASCII character.

Examples:

```
U! /P M,S,1                                  !
  ! ALTER    1                               !
  ! THIS IS THE SAMPLE LINE.                 !
U! 'F.A'10                                   ! 1
  ! THIS IS THE SAMPLE LINE                  !
  !                                          !
U! 'F'10D                                    ! 2
  ! THIS IS THE SAMPLE LINE.                 !
U! A'10                                      ! 3
  ! '10THIS IS THE SAMPLE LINE.              !
U! 3D                                        ! 4
  ! THIS IS THE SAMPLE LINE.                 !
U!                                           ! 5
  ! /                                        !
```

1 - Turn the special characters flag on, find the first occurrence of "." in the line and add a line feed character ('10) in front of it.

2 - Turn the special characters flag on, find the line feed character and delete it.

3 - Since the special characters flag is off, the string "'10" is inserted in front of the first character of the line.

4 - Delete the first three characters from the front of the line.

5 - End of alteration.