TECO-64 – New & Enhanced Features

General syntax changes

Whitespace may be generally used to make TECO-64 commands more readable, but unlike previous versions of TECO, whitespace may not occur within anything considered a token, which includes digit strings and multi-character commands such as ER or FS. Therefore, commands such as “`1 2 3 =`” are no longer valid, and must instead be written as “`123=`” or “`123 =`”. Whitespace between tokens is always valid.

On the other hand, whitespace may occur between a search or replace command and any subsequent semi-colon command. Other versions of TECO typically required that semi-colons immediately followed such commands, and printed a NAS error if they did not. So the following are now valid (within a loop):

```
@S/foo/ ;

@FN/foo/baz/ :;
```

If a colon or at-sign modifier is used preceding any command that does not allow them, a MOD error will be printed. This will also occur if more than one at-sign, or more than two colons, are used for commands that do allow such modifiers.

Numeric expressions consist of either a single numeric value, called *n*, or a pair of values, called *m* and *n*, separated by a comma. If a comma is seen, then it must be preceded by an *m* argument, and must be followed by an *n* argument. Also, multiple commas are not allowed.

^A Print string

The ^A command was changed to allow the use of a colon modifier, which causes the output of a CR/LF after the specified message, as shown in the following example:

```
:@^A/hello, world!/
```

^E Match control constructs (TECO-10)

^Ennn indicates that the character whose ASCII octal code is nnn is acceptable in this position.

^K Reset window colors

The ^K command resets the foreground and background colors for the command, text, and status line windows. The foreground color for the command and text windows is reset to black, and the background color is reset to white. The foreground and background colors for the status line are reset to white and black, respectively.

^Q Convert line numbers to character values (TECO-10)

`n^QC` is identical to `nL`. This command returns the number of characters between the buffer pointer and the nth line separator (both positive and negative). This command converts line oriented command argument values into character oriented argument values. Used after an expression.

^T Type ASCII character

The ^T command was changed to allow the use of -1, and also to allow a repeat count, as shown in the following examples:

`-1^T`	Equivalent to `13^T 10^T`.
`m,n^T`	Specifies a repeat count of *m* for the character whose ASCII value is *n*. *n* may be -1.

^V Lower case text (TECO-10)

^V puts TECO into lower case conversion mode. In this mode, all alphabetic characters in string arguments are automatically changed to lower case. This mode can be overridden by explicit case control within the search string. This command makes all strings behave as if they began with a ^V^V.

Two successive ^V characters in a string argument indicate to TECO that all following alphabetic characters in this string are to be converted to lower case unless an explicit ^W is given to override this state. This state continues until the end of the string or until a ^W^W construct is encountered.

0^V returns TECO to its original mode. No special case conversion occurs within strings except those case conversions that are explicitly specified by ^V and ^W string build constructs located within the string.

^W Upper case text (TECO-10)

^W puts TECO into upper case conversion mode. In this mode, all alphabetic characters in string arguments are automatically changed to upper case. This mode can be overridden by explicit case control within the search string. This command makes all strings behave as if they began with ^W^W.

Two successive ^W characters indicates to TECO that all following alphabetic characters in this string are to be converted to upper case unless an explicit ^V is encountered to override this state. This state continues until the end of the string or until a ^V^V construct is encountered.

0^W is equivalent to 0^V.

^X Search mode flag

The ^X command was changed to allow the use of 1 as a numeric value. It now accepts the following values for search matches:

`-1^X`	In searches, text arguments must be identical to text in the text buffer.

`0^X`	In searches, upper case letters match lower case letters, and “`”, “{”, “|”, “}”, and “~” match “@”, “[”, “\”, “]”, and “^”, respectively.
`1^X`	In searches, upper case letters match lower case letters.

= Print numeric value

The = command was enhanced to allow the use of a text argument if an at-sign modifier is used, in order to specify a *printf()* format string, as in the following example:

```
42@=/The answer is %5u/
```

Note that any string for *printf()* may be used, as long as it contains at most one numeric conversion specifier. Non-numeric specifiers such as `%s`, or multiple numeric specifiers, are not allowed. However, `%%` may be used to print a percent sign.

@ Command modifier

The @ command was enhanced to allow whitespace between a command and the text argument delimiter. A side effect of this is to disallow the use of whitespace as a delimiter.

`@S /foo/`	Equivalent to `@S/foo/`.

C Move character forward (TECO-10)

The C command was enhanced to allow the use of a colon modifier.

`n:C`	Same as `nC` except that a value is returned. If the command succeeded, -1 is returned. If the command failed, the pointer does not move and a value of 0 is returned.
`:C`	Equivalent to `1:C`.

D Delete characters (TECO-10)

The D command was enhanced to allow the use of a colon modifier.

`n:D`	Same as nD but returns a value (-1 if command succeeds, 0 if command failed because the range of characters to be deleted fell outside the text buffer).

E1 Enable/disable extended features

`E1&1` Allow extended operators within parentheses. If this bit is clear, no extended operators are allowed. When this bit is set, the following operators may be used, as long as they are within parentheses:

==	(equal)
<>	(not equal)
<	(less than)
<=	(less than or equal)

>	(greater than)
>=	(greater than or equal)
//	(division yielding remainder)
~	(exclusive OR)
<<	(arithmetic left shift)
>>	(arithmetic right shift)
!	(logical NOT)

The relational operators (the first six operators in the last above) return a value of -1 if the relation is true, and 0 if the relation is false.

`E1&2`	Modifies the behavior of the ^H command. If this bit is clear, ^H returns the time as seconds since midnight divided by two (the same as TECO on RT-11, RSX-11, and VMS). If this bit is set, ^H returns the time as milliseconds since midnight.

`E1&4`	Allows the use of braced text strings. If this bit is clear, text strings following commands work as with other TECOs. If this bit is set, then commands which are modified by at-signs can use paired braces to enclose the text strings, as in the following examples:
	@S {foo}
	@FN {foo} {baz}
	Note that a consequence of this feature is that whitespace before a left brace or after a right brace is ignored.

`E1&8`	If this bit is clear, dollar signs are not valid symbol constituents for the <CTRL/E>C match control construct or the n“C command. If this bit is set, dollar signs are valid.
`E1&16`	If this bit is clear, underscores are not valid symbol constituents for the <CTRL/E>C match control construct or the n“C command. If this bit is set, underscores are valid.

E2 Enable/disable additional error checking

`E2&1`	If set, print DIV error if attempt is made to divide by zero.
`E2&2`	If set, print IFE error if double operators are used in expressions (for example, 2**3).
`E2&4`	If set, print MOD error if command does not allow an at-sign modifier, or if more than one at-sign modifier is seen.
`E2&8`	If set, print MOD error if command does not allow a colon modifier, or if more than two colon modifiers are seen.
`E2&16`	If set, print MCA error if missing *m* argument before comma command.
`E2&32`	If set, print IMA error if command does not allow an *m* argument.

`E2&64`	If set, print UTQ error if loop is not complete within a conditional.
`E2&128`	If set, print UTL error if conditional is not complete within a loop.
`E2&256`	If set, print MOD error for m,n:P or H:P or :PW.

E3 Enable/disable I/O operations

`E3&1`	Specifies whether the line delimiter for input files is LF or CR/LF. If this bit is clear, the delimiter is LF. If this bit is set, CR/LF is translated to LF on input. The default setting is clear for Linux and MacOS, and set for Windows and VMS.
`E3&2`	Specifies whether the line delimiter for output files is LF or CR/LF. If this bit is clear, the delimiter is LF. If this bit is set, LF is translated to CR/LF on output. The default setting is clear for Linux and MacOS, and set for Windows and VMS.
`E3&4`	If this bit is clear, FF is a page separator, and is not normally included in text buffers. If this bit is set, set, FF is a normal input character and not a page separator.
`E3&8`	This bit affects the behavior of echoed input to log files (opened with EL command). If the bit is clear, all echoed input is written to the log file. If the bit is set, echoed input is not written to the log file.
`E3&16`	This bit affects the behavior of output messages to log files (opened with EL command). If the bit is clear, all output is written to the log file. If the bit is set, output is not written to the log file.

E4 Enable/disable window features

`E4&1`	This bit controls whether the text window is above or below the command window. If this bit is clear, the text window is above the command window. If this bit is set, the text window is below the command window.
`E4&2`	This bit controls whether there should be a line between the text and command windows. If this bit is clear, there is no line. If this bit is set, there is a line separating the text and command windows.
`E4&4`	This bit controls whether status information should be included on the status line. If the bit is clear, there is no information. If the bit is set, then information about the position within the file as well as the date and time are included.

nEC Increase/decrease memory (TECO-10)

The EC command has been enhanced to accept a numeric argument, which changes the memory used for a text buffer instead of closing a file. `nEC` tells TECO to expand or contract until it uses nK words of memory. If this is not possible, then TECO's memory usage does not change.

The `0EC` command tells TECO to shrink back to its original size (use the least amount of memory possible).

EI Indirect command file

The EI command has been enhanced to allow a colon modifier.

`:@EI/`*`filespec`*`/`	Open *filespec* as an indirect command file. -1 is returned if the file open succeeded; 0 is returned if the file could not be found.

EL Write TECO input and output to log file (TECO-10)

Open the specified file for output as a log file. Any currently open log file will be closed. All TECO output, including echoed input, will be copied to the file (E3 may be used to suppress either input or output). EL may also be used without a file specification to close a log file.

`EL`*`filespec`*`` ` ``	Open *filespec* as a log file.
`@EL/`*`filespec`*`/`	Equivalent to `EL`*`filespec`*`` ` ``.
`:@EL/`*`filespec`*`/`	Open *filespec* as a log file. -1 is returned if the file open succeeded; 0 is returned if the file could not be found.
`` EL` ``	Close any open log file.
`@EL//`	Equivalent to `` EL` ``.

EM Write Q-register directly to file (TECO-10)

Write the contents of a Q-register directly to the specified file.

`` EMqfilespec` ``	Write the contents of Q-register *q* to *filespec.*
`@EMq/filespec/`	Equivalent to `EMq`*`filespec`*`` ` ``.
`:@EMq/filespec/`	Write the contents of Q-register *q* to *filespec*. -1 is returned if the file open succeeded; 0 is returned if the file could not be opened.

EO TECO version

The EO command was changed to return 200 for the version number of TECO-64.

EQ Read file directly into Q-register (TECO-10)

Read the specified file directly into a Q-register. However, unlike TECO-10, trailing NUL characters are not deleted.

`` EQqfilespec` ``	Read *filespec* into Q-register *q*.
`@EQq/filespec/`	Equivalent to `@EQq`*`filespec`*`` ` ``.
`:@EQq/filespec/`	Read *filespec* into Q-register *q*. -1 is returned if the file open succeeded; 0 is returned if the file could not be found.

EW Open output file

The EW command has been enhanced to allow a colon modifier.

`:@EW/`*`filespec/`*	Open *filespec* as an output file. -1 is returned if the file open succeeded; 0 is returned if the file could not be opened.

EZ Compress command string in Q-register

The EZ command can be used to compress data in a command string stored in a Q-register, by removing whitespace and comments. This can be used to reduce the storage required for such command strings, as well as to improve the execution speed for macros.

`1EZq`	Remove all whitespace from the command string in Q-register *q*.
`m,1EZq`	Equivalent to 1EZq, but also delete any tags whose first character after the ! delimiter has an ASCII value of *m*. *m* may have the value for any graphic ASCII character other than 33 (!). If *m* is 33, or has an ASCII value for any non-graphic character, it is considered to have a value of 0, which means that no tags will be deleted.

F1 Set command window colors

Set the foreground and background colors for the command window. This command also allows the saturation levels to be set for the specified colors. The default saturation is 100%. Currently, eight colors may be specified in the text arguments below: black, red, yellow, green, blue, cyan, magenta, and white.

`F1`*`fg`*`` ` ``*`bg`*`` ` ``	Set foreground and background colors to *fg* and *bg*, respectively.
`@F1/fg/bg/`	Equivalent to ``F1fg`bg` ``.
`70@F1/fg/bg/`	Equivalent to ``F1fg`bg` ``, but set saturation for foreground color to 70%.
`70,80@F1/fg/bg/`	Equivalent to ``F1fg`bg` ``, but set saturation for foreground color to 70%, and saturation for background color to 80%.

The CTRL/K command may be used to reset colors for all windows to their defaults.

F2 Set text window colors

The F2 command is equivalent to F1, but affects colors for the text window.

F3 Set status line colors

The F3 command is equivalent to F1, but affects colors for the status line window.

FD Search and delete (TECO-10)

Search for the specified text string and deletes it.

nFDtext`	Equivalent to nFStext``.
FDtext`	Equivalent to 1FDtext``.
@FD/text/	Equivalent to FDtext`.

FK **Search and delete (TECO-10)**

Search for the specified text string and delete all characters from original position up through the end of the string.

nFKtext`	Searches for the nth following occurrence of *text* and then deletes all characters in the text buffer between the pointer positions before and after the search.
FKtext`	Equivalent to 1FKtext`.
@FK/text/	Equivalent to FKtext` except that *text* may contain any character including <ESCAPE>, other than the delimiter (shown here as /).

FL **Change characters to lower case**

nFL	Change the following *n* lines to lower case. If *n* is negative, change the preceding *n* lines to lower case.
m,nFL	Change all characters between buffer positions *m* and *n* to lower case.

FU **Change characters to upper case**

nFU	Change the following *n* lines to upper case. If *n* is negative, change the preceding *n* lines to upper case.
m,nFU	Change all characters between buffer positions *m* and *n* to upper case.

J **Jump to position (TECO-10)**

The J command was enhanced to allow the use of a colon modifier.

n:J	Same as the nJ command except that if pointer position *n* is outside of the buffer, the pointer does not move and a value of 0 is returned. If the command succeeded, a value of -1 is returned.
:J	Equivalent to 0:J.

L **Line command**

The L command was enhanced to allow the use of a colon modifier, which returns a numeric value about the number of lines in the text buffer instead of moving the dot forward or backward.

0:L	Total number of lines in buffer.
:L	Equivalent to as 0:L.

`-1:L`	Number of lines preceding dot.
`1:L`	Number of lines following dot.

R **Reverse command (TECO-10)**

The R command was enhanced to allow the use of a colon modifier.

`n:R`	Same as nR except that a value is returned. If the command succeeded, -1 is returned. If the command failed, the pointer does not move and a value of 0 is returned.
`:R`	Equivalent to `1:R`.

{} **Braced text arguments**

If a command allows an at-sign modifier for text arguments, and if the E1&4 is set, then braces may be used to delimit the text arguments. Note that a consequence of doing this is that whitespace before a left brace or after a right brace is ignored. Note the following examples:

@S{foo}	Equivalent to `@S/foo/`.
@FN{foo}{baz}	Equivalent to `@FN/foo/baz/`.
@FS {foo} {baz}	Equivalent to `@FN/foo/baz/`.