MODIFIED EDITOR
EDITMOD

User's Guide

Version 2

November 15, 1976

PREFACE

The Modified EDITOR (EDITMOD) is an enhanced version of the DOS General EDITOR. All available memory is used as a circular buffer which allows rolling backwards through a file. User defined command strings give the user an interpreted language facility. The use of two scratch files protects the input file from modification until the EDIT is completed. Text files now carry their individual configuration (tabs, etc.) in their first sector, eliminating the EDIT/OV1 file. Many command enhancements and additions have been made for consistency and usability for the user. The chapters in this User's Guide have been numbered so as to provide easy updating of the DOS. User's Guide, Version 2 (Model Code 50216).

TABLE OF CONTENTS

CHAPTER 1. INTRODUCTION

The Modified EDITOR (EDITMOD) is an enhanced version of the DOS General EDITOR.

All available memory is used as a circular buffer which allows rolling backwards through a file. This also speeds up FINDs and LOCATEs and reduces disk "thrashing".

User defined command strings (:0 through :9) give the user an interpreted language facility. Additional commands allow conditional or unconditional skips along the command string. These command strings may be pre-defined in an EDITable file (default EDITMOD/DEF) which is automatically loaded when EDITMOD is executed.

The use of two scratch files protects the input file from modification until the EDIT is completed. It also allows multiple passes on a file while in 'ONE-PASS' option.

Text files now carry their individual configuration (tabs, etc.) in their first sector, eliminating the EDIT/OV1 file.

Many command enhancements and additions have been made for consistency and usability for the user. For instance, MODIFY now has a VERIFY option and FINDs and LOCATEs allow a field parameter and display the full screen.

The chapters in this User's Guide have been numbered so as to provide easy updating of the DOS. User's Guide, Version 2 (Model Code 50216).

CHAPTER 2. EDIT COMMAND

2.1 Introduction

The DOS Editor is used to create and to update source data files on the disk. The editor will enable the creation of files in a variety of formats: text files, assembler code files, DATABUS source code files, or many user designed data files.

A GLOSSARY of the many terms and phrases used throughout this chapter is provided in the Glossary at the end of the chapter. A list of commands and brief definitions is provided in the COMMAND List Section.

2.2 Operation

The EDIT program is parameterized as follows:

```
EDIT <f1>[,<f2>][,<f3>][;parameter list]
```

2.2.1 Files

<f1> is the source file, [<f2>] is the scratch file and [<f3>] is the definition file. The source file <f1> is assumed to have an extension of 'TXT' if none is provid provided. If there is no file of the specified name, one will be opened after checking with the operator. If no scratch file [<f2>] is specified, a primary file 'SCRATCH/TXT' and a secondary scratch file 'SCRATCH/XTX' will be used. The definition file [<f3>] is assumed to be EDIT/DEF unless otherwise specified.

2.2.2 Parameter List

A parameter list, indicated by the SEMI-COLON (;) following the file specifications, may be included. That list may include up to seven parameters which are order independent. The possible parameters are:

```
[;[margin][tab key][mode][shift][line][update][keyclick]]
```

If no parameter list is provided, Assembler mode with a margin at 75 and SPACE bar for tabbing is assumed.

2.2.2.1 Margin Bell

A number in the parameter list will be taken to be the margin designator; this causes the margin 'bell' to ring at the designated margin. The default margin is 75.

For Example ";30" will cause the bell to ring in column 30.

2.2.2.2 Tab Key Character

A tab key character encountered in the parameter list, i.e., non-alpha, non-numeric, non-colon, will replace the assumed tab key character. (SPACE in Assembler, DATABUS and Comment mode, SEMI-COLON in Text mode.)

For Example, ";^" will cause the caret key (^) to replace the assumed character as the tab key.

2.2.2.3 Mode

A new set of assumptions will be used if one of the 'mode' parameters is set. If no mode is listed or 'A' is typed, Assembler mode will be used. DATABUS or DATAFORM (D) mode simply changes the tab stops. Comment mode (C) changes the nature of the DELETE and SCRATCH commands to facilitate adding or changing comments on assembly code files.

Text mode (T) sets no tabstops, does no shift inversion and enables the word wrap around feature (see the glossary). To activate line truncation instead of word wrap-around in Text mode, enter 'L' in the parameter list. To enable shift key inversion (see glossary) in Text mode, enter the parameter 'S' in the list. Text mode is especially useful for generating SCRIBE input files.

See the glossary for complete definitions of the various modes.

2.2.2.4 Update

During editing, the source file is transferred into the scratch file as the text is updated. A second scratch file may be used as the scratch file as the edit proceeds. When the edit is terminated, the physical source file is updated.

The 'ONE-PASS' parameter 'O' may be set in the parameter list. Then, at the completion of the edit, the scratch file will contain the updated information and the source file will be unchanged.

2.2.2.5 Key-click

If the 'K' parameter is set, a 'click' will sound each time a key is struck.

2.2.3 Examples

To perform standard Assembler code editing, enter the command:

```
EDIT <source>
```

To edit a file for input to the text processor, SCRIBE, enter the command:

```
EDIT <source>;T
```

To change the margin bell to ring at column 35 (e.g. for labels) enter the command:

```
EDIT <source>;35T
```

The parameters would set the bell and use the Text mode assumptions. Note that the parameters are order independent; therefore, the command:

```
EDIT <source>;T35
```

would achieve the same results.

To generate a second slightly different file (without updating the original file), enter the command:

```
EDIT <source>,<new file>;OT
```

If the file is Assembler code instead of text, simply omit the 'T'; if DATABUS, replace 'T' by 'D'.

A second file, with the same name as <f1> but with a different extension, may be used as the scratch file by entering:

```
EDIT <f1>,/<extension>
```

Once the initial command (and parameter list) has been entered, the DOS Editor signon message will appear on the screen followed with the file's configuration information (e.g. tabs, special characters, margin, keyclick, word wrap-around, and shift inversion) with the cursor left on the 'command line'. From this position data may be entered, lines may be fetched from the source file, or editor commands may be entered.

2.2.4 Data Entry

To enter text, simply type on the bottom line; when the ENTER key is presssed the screen rolls up one line. The command line is once again blank and the cursor is at the beginning of the command line, ready to accept more input.

When a SPACE is typed to the right of the margin bell column and word wrap-around is enabled, the editor will automatically roll up the screen and begin a new line. If a non-space character is typed into the last column and word wrap-around is enabled, the last word on the line is removed and, after the screen is rolled up, that word is placed on the command line, where data entry may proceed.

When typing on a 'screen line' (as the result of a command), the ENTER key causes the cursor to return to the command line. To continue data entry at the same screen area, the Pseudo-ENTER key may be used. This key (shift DEL) causes (in all but command mode), a new blank line to be inserted at that point on the screen so that data entry may proceed.

If word wrap-around is enabled, and data is being entered on a screen line, a new line will automatically be inserted at that point when, as on the bottom line, a space is entered to the right of the margin bell column or a character is typed past column 79.

The BACKSPACE key erases the last character and moves the cursor back one position. The CANCEL key erases the line back to the previous tabstop (this would erase the entire line if no tabs are set).

Typing the tab key character causes the cursor to move to the next tab stop to the right. If there are no tab stops to the right of the cursor, the tab key character is accepted as a normal data character.

2.2.5 Data Retrieval

To fetch data from the source file, hold down the DISPLAY key and press the KEYBOARD key. As long as the two keys are depressed, data will be fetched, displayed on the command line and rolled up the screen. If the end of file is reached, no more data is fetched and the machine beeps.

To fetch data backwards, hold down the KEYBOARD key and press the DISPLAY key. As long as the two keys are depressed, the screen will be rolled down and prior lines inserted on the top line. If the beginning of the file contained in memory is reached, the machine beeps and then data is fetched rolling up the screen.

To fetch a single line, the shifted DEL key may be pressed in the first column of the command line. Using this key insures that only one input line will be fetched.

All of available memory is used as a circular buffer. As the operator proceeds through the file, EDIT writes lines from the end of the buffer out to disk, maintaining a pre-screen buffer so the user can roll backwards. The pre-screen buffer size varies from about 61 lines in a 16K processor to about 189 lines in a 48K processor. Rolling backwards is restricted by the size of the pre-screen buffer. Of course, if the file is small enough to fit completely in memory (500 to 1400 lines in a 48K processor), there is no restriction. Sometimes the processor will appear to hang while it is doing buffer management.

2.3 EDITOR Command Format

The text appearing on the screen lines (i.e. the lines above the command line) may be edited using a set of 'commands'. A 'pointer' (>) in the left hand column of the screen indicates the line which the command will affect.

To move the pointer up, press the KEYBOARD key. To move the pointer down, press the DISPLAY key. The pointer wraps around from the top to the bottom and vice versa.

Commands allow the user to delete a single line (:D) or part of the screen (:SC and :SB), insert (:I) a new line between the current lines on the screen and modify (:M) parts of a line by replacing text or inserting new text. Commands are also available to search the file for specific text (:F and :L), for the end of the file (:EO or :E*), or for a particular line by number (:G).

An editor command is always preceeded by a COLON (:). To enter a command, type a colon in the first column of the command line with the appropriate command character(s) and any necessary parameters. The command characters may be upper or lower case. To force a line to be entered with a colon in the first column, start the line with two colons (the first one will be discarded and the line shifted left).

2.4 Basic EDITOR Commands

The following commands are a few basic editor commands. The user can get started without worrying about complex command forms. Remember that the 'pointer' on the screen indicates the line affected by the command.

2.4.1 Setting Tabs

:T - TAB set - this command enables the user to reset the tab stops during execution. The command causes a line of numbers to be displayed across the bottom of the screen.

The operator should space over to each position where a tabstop is desired and type any non-blank character. These tab stops are meaningful during data entry. A maximum of 20 tab stops may be set.

See the section on 'Changing Tabs' for more information on setting tabs.

2.4.2 Setting TEXT Mode

:X - TEXT - this command enables word wrap-around and disables shift key inversion and space insertion after leading periods, pluses, and asterisks. It automatically enters the tab set command (:T), so that tab stops may be cleared by the operator. The tab key character is not changed; therefore, the ":[tab key]" command must be used to set a new tab key character if one is desired.

See the section on 'Changing Modes and Options' for more information.

2.4.3 DELETEing a Line

:D - DELETE - this command deletes the entire pointed line.

The cursor is left on the now null line where new text may be entered. If no replacement text is needed, pressing the ENTER key in the first column of the pointed line returns the cursor to the command line.

Pseudo-ENTER may be used to generate additional lines at this area of the screen. Word wrap-around, if enabled, will apply to text entered on a deleted line. Pressing the ENTER key will return the cursor to the command line.

For other approaches to deleting lines, see the 'Deleting Lines' section.

2.4.4 INSERTing a Line

:I - INSERT - Perform a line insert at the pointed line. This command causes the lines from the bottom of the screen to the pointed line, inclusive, to be rolled down and a blank line to be inserted. The cursor is left at the beginning of the new blank line where data entry may proceed.

To make complex changes to a line already on the screen, the operator may INSERT a line immediately below the original and then retype the line with the changes. The original line may then be DELETED.

The Pseudo-ENTER (shift DEL) key may be used to generate additional lines at the same point on the screen.

2.4.5 COPYing a line

:A - APPEND - copies the pointed line to the bottom of the screen and rolls the screen up one line.

The cursor returns to the command line.

:C - COPY - copies the pointed line to the bottom of the screen, deletes the pointed line and rolls the screen up one line.

The cursor is left on the now null pointed line. Text may be entered at this point (the Pseudo-ENTER and word wrap around, if enabled, will apply). When the ENTER key is finally pressed, the pointer is automatically moved to the following screen line so that a group of lines may be easily copied to another part of the screen.

2.4.6 MODIFYing a line

:M [old text][command separator][new text] - MODIFY - a modify command allows the operator to replace [old text] by [new text], insert [new text] after [old text] or append (i.e., truncate and add) [new text] after [old text]. For instance:

```
:M [old text]<[new text]
```

would replace [old text] on the pointed line with [new text]. The command:

```
:M [old text]>[new text]
```

would insert [new text] immediately following [old text] on the pointed line. The command:

```
:M [old text]\[new text]
```

would truncate the pointed line immediately following [old text] and then append [new text].

If [old text] is not found in the pointed line, the machine will beep and return to the command line without making any modification to the pointed line.

For the many various forms of this command see the 'MODIFY Command' section.

2.4.7 LOCATEing a Line

:L - LOCATE next - typed exactly :L[ENTER], finds the next line of text. If positioned at the end of the file, the 'next' line will be the first line of the file.

:L [old text] - LOCATE match - this command searches for a line containing imbedded text matching [old text]. Leading spaces should be supplied if meaningful.

For additional approaches to locating a line, see the 'File Search Commands' section.

2.4.8 ENDing the EDIT

:E* - EOF without display - searches for the end of the file and, when it is reached, displays the last screen of text. The search may be aborted by pressing the KEYBOARD key.

:E - END - the END command causes the remainder of the logical source file to be copied to the logical scratch file and then, if the logical scratch is not the physical input file, the scratch file is copied back to the source file.

The command line will be left on the screen as long as the copy from source to scratch is in progress; it is erased during the final copy from scratch back to source.

The END may be aborted as long as the command line is still displayed, by pressing the KEYBOARD and DISPLAY keys simultaneously. When the final copy is completed, control is returned to DOS.

2.5 Intermediate Commands

Most of the following commands are expansions of the ones above. One additional concept is that of "fields". A field is a portion of the line between two consecutive tabs. Field one is between the left margin and the first tab, field two is between the first and second tab, etc. Even though up to twenty tabs may be set, only the first nine fields may be referenced.

2.5.1 Changing Special Characters

:[tab key] - change tab key character to any non-alpha, non-numeric, non-COLON, non-ENTER character typed after a leading colon on the command line.

:[old modify operator] [new modify operator] - change the old modify operator to the new one specified.

:CH - display the current special characters.

For instance, if the user wanted to use "]" for the tab key, "=" for the modify replace operator, "-" for the modify insert operator, and "|" for the modify append operator, he would type in:

```
:]
:< =
:> -
:\ |
```

Then to check that they were properly changed, the user would type:

```
:CH
```

which would display:

```
:CH TAB KEY: ]  MODIFY REPLACE: =  MODIFY INSERT: -  MODIFY APPEND: |
```

At the end of the EDIT, the special characters and tabs are stored in the EDITed file. The next time that the file is EDITed, the same characters will be used. The tabs and special characters will then be displayed below the signon message.

2.5.2 Changing Tabs

:T - TAB set - this command enables the user to reset the tab stops during execution. The command causes a line of numbers to be displayed across the bottom of the screen.

The operator should space over to each position where a tabstop is desired and type any non-blank character. These tab stops are meaningful during data entry and for referencing fields (the portion of the line between consecutive tab stops). A maximum of 20 tab stops may be set.

:T [nn][,nn]... - TAB set by column number - this command enables the user to reset the tab stops by column number. For instance, entering ":T 9,15,30" would set the tabs to columns 9, 15, and 30. A maximum of 20 tab stops may be set.

At the END of the EDIT, the tab positions and special characters are stored in the EDITed file. The next time that the file is EDITed, the same tabs and special characters will be used. They are displayed immediately below the signon message.

Below are commands for setting tabs to pre-determined default values.

:TA - Set Assembler tabs at columns 9, 15, and 30.

:TD - Set Datashare/Databus tabs at columns 10 and 20.

:TS - Set SNAP tabs at columns 11, 21, and 38.

:RH - RPG Header - sets tab stops for RPG header specification at columns 6 and 15.

:RF - RPG File - sets tab stops for RPG file description specification at columns 6, 15, 24, 33, 40, 54, 66 and 70.

:RE - RPG Extension - sets tab stops for RPG extension specification at columns 6, 11, 19, 27, 33, 36, 40, 46, 52 and 58.

:RL - RPG Line - sets tab stops for RPG line counter specification at columns 6, 15 and 20.

:RI - RPG Input - sets tab stops for RPG input specification at columns 6, 15, 21, 44, 53, 59 and 65.

:RC - RPG Calculation - sets tab stops for RPG calculation specification at columns 6, 18, 28, 33, 43, 49, 54 and 60.

:RO - RPG Output - sets tab stops for RPG output specification at columns 6, 15, 23, 32, 38, 40 and 45.

:RS - RPG Summary - sets tab stops for RPG summary specification at columns 6, 14 and 23.

2.5.3 Changing Modes and Options

:X - TEXT - this command enables word wrap-around and disables shift key inversion and space insertion after leading periods, pluses, and asterisks. It automatically enters the tab set command (:T), so that tab stops may be cleared by the operator. The tab key character is not changed; therefore, the ":[tab key]" command must be used to set a new tab key character if one is desired.

:XI - Invert TEXT - this command enables shift key inversion and disables word wrap-around and enables space insertion after leading periods, pluses, and asterisks. It automatically enters the tab set command so that tab stops may be reset by the operator.

:K - Keyclick - this command causes the machine to 'click' every time a key is struck.

:KI - Invert Keyclick - this command turns off the 'click' set above.

2.5.4 Deleting Lines

The user may delete the leading part of a line, the whole line, or multiple lines.

:D - DELETE - in all but Comment mode this command deletes the entire pointed line. In Comment mode, only the comment field is deleted. However, the CANCEL key may be used to delete the preceeding fields in the line.

The cursor is left on the now null line where new text may be entered. If no replacement text is needed, pressing the ENTER key in the first column of the pointed line returns the cursor to the command line.

Pseuco-ENTER may be used to generate additional lines at this area of the screen. Word wrap around, if enabled, will apply to text entered on a deleted line. Pressing the ENTER key will return the cursor to the command line.

:D [old text] - DELETE through - this command deletes all characters from the left edge of the pointed line through (and including) the specified [old text]. The remaining characters will be left justified and re-displayed.

The cursor returns automatically to the command line.

:D[#] [old text] - DELETE through field - this command deletes all characters from the left edge of the pointed line through (and including) the specified [old text] in the specified field. The remaining characters will be left justified and re-displayed.

The cursor returns automatically to the command line.

:SC - SCRATCH above - in all but Comment mode this command erases the lines from the top of the screen down to the pointed line, inclusive. (In Comment mode, only the comment fields are erased.)

The cursor is left on the pointed line where data entry may proceed.

:SB - SCRATCH below - in all but Comment mode this command erases the lines from the pointed line to the bottom of the screen, inclusive. (In Comment mode, only the comment fields are erased.)

The cursor is left on the pointed line, where data entry may proceed.

2.5.5 MODIFY Command

The general form of the MODIFY command is:

:M[V][#] [old text][modify operator][new text]

where [V] is the VERIFY option, [#] is the optional field number, and [modify operator] is the command separator which defines the action of the command. Both [old text] and [new text] fields are optional. If [old text] is omitted, the command will take effect at the left most edge of the pointed line (or at the left edge of the specified field). If the [new text] field is omitted, a null field will be used to execute the modification.

The VERIFY option causes the cursor to blink at the first character to be modified. The user then has three responses. If he presses 'Y' for 'YES', the modification will take place. If he presses the 'ENTER' key, control will then return to the command line. If he presses any other key, the modify command will continue to search the line for another occurance of [old text] to modify.

2.5.5.1 Line Modification

The following descriptions are of the line modification version of the MODIFY command.

:M[V] [old text][replace operator][new text] - MODIFY (replace) - replace the specified [old text] by the specified [new text]. The "less than" character (<) is the default command separator which indicates replacement and, therefore, the [old text] may not contain this character. If [new text] field is omitted, the [old text] will simply be deleted and the line will be compressed to the left.

For example to modify the text line:

```
THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG'S BACK.
```

The command: ":M BROWN<RED" would cause the line to be redisplayed like this:

```
THE QUICK RED FOX JUMPED OVER THE LAZY DOG'S BACK.
```

The command: ":M .< 1234 TIMES." to the above line would generate a line like:

```
THE QUICK RED FOX JUMPED OVER THE LAZY DOG'S BACK 1234 TIMES.
```

If the replacement causes the line to become longer than 79 characters and word wrap-around is enabled, the trailing word will be wrapped around and a new line will be inserted containing the entire last word. If the [new text] is shorter than the [old text] it replaces, the line will be shortened.

After the pointed line is redisplayed, the cursor is returned to the command line.

:M[V] [old text][insert operator][new text] - MODIFY (insert) - the command separator "greater than" (>) is the default character causing the [new text] to be inserted in the pointed line immediately after the [old text].

If the line becomes longer than 79 character, and word wrap around is not in effect, the trailing characters are truncated. If, however, word wrap around is on, the last word(s) are inserted on a new line.

:M[V] [old text][append operator][new text] - MODIFY (append) - the "backslash" (\) is the default command separator causing everything in the pointed line past the [old text] to be replaced by the [new text].

As in all MODIFY commands, if the pointed line becomes longer than 79 characters, truncation occurs if word wrap around is not enabled.

:M or :M[#] - MODIFY repeat - typed exactly :M[ENTER] or :M[#][ENTER], uses the [old text][sep][new text] from the last MODIFY command. This is useful when making the same change repeatedly. Note that the field number is not saved, and must, therefore, be supplied if necessary.

:MV or :MV[#] - MODIFY VERIFY repeat - typed exactly :MV[ENTER] or :MV[#][ENTER] is the same as the above command except that it invokes verification.

:M* - MODIFY display - display the expression entered for the last MODIFY. After the saved command is displayed, the cursor is turned off and the operator must press ENTER to proceed. No MODIFY is actually performed.

2.5.5.2 Field Modification

In field modification mode, the MODIFY command acts only on a specific field and does not expand or contract the entire line but maintains the integrity of all fields before and after the affected field.

:M[V][#] [old text][modify operator][new text] - MODIFY field - where the pound sign [#] is a number from 1 to 9 designating the field to be modified (or the starting point to search for matching [old text]). In Assembler mode, field 1 is the label, field 2 is the op code, field 3 is the expression and field 4 is the comment. This command may be executed in any of the previous Modify forms. However, modification is performed within the specified field only. As long as the text being modified is unique, field 1 may be specified, since the field number indicates only where to start looking for matching text. (Note that if the field number is omitted, line modification is assumed.)

Thus, a replacement or append shorter than the original field will be blank filled and subsequent fields will maintain their position and content. An insertion longer than the specified field will be truncated (with the exception of the last field whenever word wrap around is in effect).

For example, in Assembler mode, the line:

```
LABEL     OP        EXP                 COMMENT
```

the label may be deleted by the command:

```
:M1 \
```

with the resultant line:

```
          OP        EXP                 COMMENT
```

Or, the expression field (EXP) could be changed to EXP+1 without disturbing the comment field position, by the command:

```
:M3 EXP>+1
```

which generates:

```
LABEL     OP        EXP+1               COMMENT
```

To add a comment to a line previously containing none or to replace an existing comment field, enter:

```
:M4 \[new comment]
```

NOTE: Remember when using the repeat form of the MODIFY command that the field number may need to be supplied.

2.5.6 Line Splitting

:V [nn] - SPLIT - split the pointed line at column [nn], inserting the remainder of the line past column [nn] below the pointed line. This is useful for INSERTing sentences in the midst of text without doing a group of cumbersome MODIFYs.

2.5.7 Line Concatination

:W - CONCATINATE - append the line below the pointed line to the pointed line. The pointed line will be assumed to have a trailing space if word wrap-around is in affect. This is useful in text mode where a MODIFY has caused words to wrap-around to the next line and the operator wished to include it into the following line.

2.5.8 File Search Commands

Manual, operator controlled, searches may be performed by depressing the KEYBOARD and DISPLAY keys simultaneously to cause data to be fetched from the file (forward or backward depending which key was pressed first) and displayed on the screen. This will continue until either key is released. The :EO command performs the same function automatically, i.e., it causes lines to be fetched and displayed until the end of the file is reached. Pressing the DISPLAY key will stop the :EO command until it is released. To terminate an :EO command, press the KEYBOARD key. To fetch a single line use the Pseudo-ENTER key (shift DEL).

The :E- command works the same way as the :EO except that it fetches lines backwards through the file in memory rolling the screen down.

To find the end of a file without displaying the entire file (since the display is time consuming) use the :E* command. This will search for the end of file and display the last screen of data.

FINDs and LOCATEs search the file for a line containing specific text. When a line has been found, it is aligned with the pointer on the screen so that it may be operated on. Lines above and below the line are also displayed. FINDs and LOCATEs allow field specification, that is, if a field is specified, only lines with the specific text in that field will be found.

A FIND or LOCATE will wrap entirely around the file. If the requested text is not found, the last screen image when the FIND or LOCATE was executed will be displayed and the machine will beep. A FIND or LOCATE may be aborted by pressing the KEYBOARD key.

The [old text] specified for a FIND (or LOCATE) command is saved. The saved match may be redisplayed or used again.

:F [old text] - FIND match - the input file is searched for a line starting with the specified [old text]. Leading spaces in the file's lines will be ignored and need not be entered as part of [old text] unless meaningful. Note that this command should be typed exactly :F[SPACE][old text].

:F[SPACE] - FIND same match - if the FIND command is followed by exactly one space and the ENTER key, the previous FIND (or LOCATE) [old text] will be used for this FIND. Several occurrences of the same text may be searched out in this manner.

:F* - FIND display - the asterisk (*) after the FIND command causes the [old text] of the previous FIND or LOCATE command to be displayed. The cursor is turned off and the operator must press ENTER to proceed. No FIND is performed.

:L - LOCATE next - typed exactly :L[ENTER], finds the next line of text. If positioned at the end of the file, the screen is cleared and the 'next' line will be the first line of the file.

:L [old text] - LOCATE match - similar to FIND match except that the locate command searches for a line containing imbedded text matching [old text]. Leading spaces should be supplied if meaningful.

:L[space] - LOCATE same match - typed exactly :L[SPACE][ENTER], uses the [old text] specified by either the previous LOCATE or FIND command to perform a search.

:L* - LOCATE display - display the [old text] entered for the previous LOCATE or FIND command. As in the FIND display, the cursor is turned off and the operator must press ENTER to continue. No LOCATE is actually performed.

There are several variations on the GET command that allow the user to quickly jump forward or backward through a file or to a specific line by number.

:G - GET next screen - This will clear the screen and refill it with the next screen image.

:G- - GET prior screen - This will clear the screen and refill it with the prior screen image.

:Gnnnnn - GET nnnnn lines - This will fetch nnnnn (from 1 to 65,535) lines from the file or until the end of file. For example, ":G1" will roll the screen up one line.

:G-nnnnn - GET nnnnn lines backward - This is the same as the above command except that it fetches backwards through the file in memory.

:GAnnnnn - GET Absolute [nnnnn]th line - This command will position the file so that the [nnnnn]th line of the file (counting from the beginning of the file) is displayed on the bottom line of the screen.

2.5.9 BYPASS End of File

:B - BYPASS - fetch a line from the file, bypassing the end of file. This may be a true end of file or one caused by RECORD FORMAT errors, PARITY errors, or a RANGE TRAP (see the section on Recovery Procedures). Subsequent lines may then be fetched by the normal mechanisms. This command is intended as a recovery tool for use only if the file has been accidentally shortened or contains badly formatted records.

2.5.10 Terminating the EDIT

:E* - EOF without display - searches for the end of the file and, when it is reached, displays the last screen of text. The search may be aborted by pressing the KEYBOARD key.

:EO - EOF with display - causes the data to be fetched and displayed on the screen continuously until end of file is reached. The search may be stopped by pressing the DISPLAY key or terminated by pressing the KEYBOARD key.

:E- - Display to the beginning of file - works exactly as the :EO command above except that it fetches backward through the file in memory, rolling the screen down.

:E - END - the end command causes the remainder of the logical source file to be copied to the logical scratch file and then, if the logical scratch is not the physical input file, the scratch file is copied back to the source file.

The commanc line will be left on the screen as long as the copy from source to scratch is in progress; it is erased during the final copy from scratch back to source.

The end may be aborted as long as the command line is still displayed, by pressing the KEYBOARD and DISPLAY keys simultaneously. When the final copy is completed, control is returned to DOS.

:E/ - END/DEL - this command causes the remainder of the source file to be deleted (the lines currently on the screen will be written out), and, if the logical scratch file is not the physical source file, the scratch file is copied back to the source file. When the file is completely updated, the system is reloaded.

:E\ - END/DEL - this command causes the prior portion of the source file to be deleted (the lines currently on the screen will be written out), and the remainder of the file to be written out. If the logical scratch file is not the physical source file, the scratch file is copied back to the source file. When the file is completely updated, the system is reloaded.

:EX [DOS command string] - END and Execute - this command does an END (":E" above) and then executes the DOS command string.

:Q - this command will cause the EDITOR to return to DOS without updating the source file.

:QX [DOS command string] - this command will cause the EDITOR to return to DOS without updating the source file and then execute the DOS command string.

2.6 Advanced Commands

:0 [user defined command string] - Define the user defined command string zero.

:0 - Execute user defined command string zero.

The user may define and execute up to ten EDITOR command strings. These strings may use themselves, do conditional skips of commands, and request operator response. This gives the EDITOR the capability of doing sophisticated file modification in a semi-automatic manner (the user always has complete control).

The user may define commands ":0" through ":9" as one or more EDITOR commands. It may be single command (e.g. a complicated MODIFY command) that is used quite often, but one the user doesn't want to type in every time he needs it. Or it may be a string of commands separated with Pseudo ENTER (shift DEL) characters. The Pseudo ENTER character appears as a solid triangle on a High-Speed or "RAM" screen and as a carat (^) on a slow screen. The command string should be terminated with a Pseudo ENTER (shift DEL) if trailing spaces are significant. A carat is used in the examples below. For instance, if the user typed:

```
     :2 :M abcdefghijklmnop<ponmlkjihgfedcba
```

he would have defined command 2. Every time that he typed in:

```
     :2
```

the pointed line would be modified in the specified manner. If the user typed:

```
     :5 :M abcde<edcba^:M fghij<jihgf
```

he would have defined command 5 as a pair of modify commands. Every time that he typed in:

```
     :5
```

the pointed line would have both modifications done in sequence (if the first modification failed, the second one would not be performed) just as if the user had typed them in separately.

If the user needed to replace every occurance of the string "LABEL" in his file with "LABEL1", he may define a command as:

```
     :2 :L LABEL ^M LABEL <LABEL1^2
```

Note: a new definition discards the old definition. Also, the colon following the Pseudo ENTER character is optional. The user may then type:

```
     :2
```

which will then hang in a loop changing "LABEL" to "LABEL1" everywhere it is found in the file. The command string will terminate when the LOCATE fails. Of course the user can always terminate the LOCATE with the KEYBOARD key.

For a more complicated example, the user may be EDITing the disk file which may be created by the DOS FILES Command, and define a command as:

```
     :9 :M2 \:DR0,:DR1^M  <^M^M^M^M^M^M^M <COPY^G1^9
```

Then he would set a tab at column 13 (immediately after the file extension in the FILES-created file). By typing:

```
     :9
```

the user would create a CHAIN file for copying all the files from drive zero to drive one.

The execution of any command string may be temporarily stopped by holding down the DISPLAY key or terminated by pressing the KEYBOARD key.

:0* - Display the user defined command string zero.

:00 - Display the user defined command strings zero through nine.

The above commands allow the user to examine command strings individually in relation to the text on the screen or to examine them all at once in relation to each other.

:0< - Insert user defined command string zero into the file text immediately below the pointed line.

:0> - Define user defined command string zero as the pointed line on the screen.

The above commands allow the user to save the command strings in the text of his file. It also simplifies the modification of command strings as the user can use MODIFY on the string rather than keying the entire string in again. It also assists in defining several similar command strings.

A definition file with a default name of EDIT/DEF may be created by the user to contain a set of user pre-defined command strings. These are loaded automatically every time that EDIT is executed. This is an EDITable file. Remember, the user can force a colon as the first character on a line by starting the line with a double colon. The definition file may contain comment lines (e.g. lines starting with "+", "*", or ".") or null lines. The sequence of the defined command strings has no affect. Command strings may be multiply defined; the last definition will be the one in affect.

:99 [user defined command string] - Define an initialization command string in the definition file.

If an initialization command is defined in the definition file, it is executed automatically when EDIT is executed. It may be defined to do things such as change the tab key, turn on key click, change the modify operators, set tabs stops, or even do file modification without operater intervention. The automatic execution of the initialization command may be overridden by pressing the KEYBOARD key when executing EDIT.

:Z* - Terminate execution of the user defined command string and return control to the bottom kevin line.

:Z - Skip over one command after the following command in the user defined command string if the following command fails.

:Z[n] - Skip over [n] commands (0 to 9) after the following command in the user defined command string if the following command fails.

Almost every EDITOR command either "succeeds" or "fails". For instance, LOCATE succeeds if it finds a line containing the specified text and fails if it doesn't. The MODIFY command fails if the string to be modified is not found or if, using the VERIFY option, the user has pressed the ENTER key. The GET fails if the end of file is reached. The use of the above commands allows conditionally skipping over commands in the user defined command depending on the success of a command.

:U[n] - Unconditionally skip over [n] commands (1 to 9) in the user definec command string.

This allows skipping over commands that might be jumped to by a conditional skip.

:Q [string] - QUERY, setting a succeed or fail condition, if the specified string is contained in the pointed line.

This command works similar to the FIND or LOCATE command in that it uses the line save area and allows field specification. It does not affect the pointed line at all but sets up a conditional skip. For instance, if command 3 were defined:

```
:3 :Z^Q2 ABC^Z*^G1^3
```

then its execution would GET lines until the pointer pointed to a line containing the string "ABC" in field 2. While the QUERY fails, the first conditional skip command (":Z") caused execution of the command string to skip over the ":Z*", GET a line, and then start over. When the QUERY succeeds, the ":Z*" is executed which terminates the command string. This example effectively does a LOCATE while displaying all the lines examined.

2.7 Recovery Procedures

A 'FORMAT TRAP' occurs when a record not belonging to the current file is encountered. This can be caused either by a physical misalignment of the disk read head or because a record has erroneously been written into that file by some other program.

A 'RANGE TRAP' occurs when the physical limit of the file is reached and no end of file is present.

A 'PARITY TRAP' occurs when a record is misread from the disk. This may be caused by physical misalignment of the disk read head or a bad surface on the disk.

These three above errors will cause the EDITOR to believe that it has reached the end of file. To read past past an end of file, use the BYPASS command, ":B", repeatedly if necessary.

2.7.1 File Recovery

If the source file is lost (e.g., erroneously KILLed), one of the scratch files may contain a useful copy. Since the scratch files (SCRATCH/TXT or SCRATCH/XTX) usually contain a copy of the last file edited, they may be used to recover only that file.

Note: If the file fit completely within the memory buffer, scratch files are never used.

2.8 Glossary

Assembler mode - assumed mode of execution. Tab stops at 9, 15 and 30. The space bar is assumed as the tab key character (this may be changed in parameter list or during execution). Shift key inversion and no word wrap around are assumed. Leading periods (.), pluses (+), and asterisks (*) generate a following space (.) or (+) or (*) for comment lines.

Command - characters typed at the left edge of the command line following a COLON (:) which have special meaning to the editor.

Command line - the bottom line of the screen where most data is entered, lines are fetched and commands are typed.

Comment field - in assembler code the fourth field which is generally used for programmer comments.

Comment mode - assumed if 'C' in parameter list. Facilitates changing or adding comments to assembler code. The space bar is assumed to be the tab key character (this may be changed in parameter list or during execution). Shift key inversion and no word wrap around are assumed. Leading periods (.), pluses (+), and asterisks (*) generate a following space (.) or (+) or (*) for comment lines. Pseudo-ENTER positions to comment field of following line and deletes the comment. Delete and Scratch commands affect only the comment field.

DATABUS mode - assumed if 'D' in parameter list. Tab stops at 10 and 20 (may be changed during execution). The space bar is assumed to be the tab key character (this may be changed in the parameter list or during execution). Shift key inversion and no word wrap around are assumed. Leading periods (.), pluses (+), and asterisks (*) generate a following space (.) or (+) or (*) for comment lines.

Definition file - this is an EDITable file containing pre-defined user command strings which is automatically loaded when the Editor is executed. The definition file may also contain an initialization command (":99") which is automatically executed unless the "KEYBOARD" key is pressed. The default name for the file is EDIT/DEF.

Field number - a digit used in by commands to designate the portion of the pointed line between two consecutive tab stops. Field '1' is always from column 1 to the first tabstop; thus, in Assembler mode, '1' designates the label field, '2' the opcode field, '3' the expression field and '4' the comment field. During field modification, leading and trailing fields are preserved.

Line insert - results from an INSERT command, data entry or modification when word wrap around is in effect or a Pseudo-ENTER key in any mode other than Comment. The lines below the pointed line are rolled down and a new, blank line is generated at the pointed line.

Logical scratch file - current output file.

Logical source file - current input file.

Modify operator - the character in a MODIFY command which indicates what is to be done. The default replace operator is the "less than" symbol (<), the default insert operator is the "greater than" symbol (>), and the default append operator is the "backslash" (\).

New text - a group of characters, typed immediately after a modify operator in a modify command, which will become part of the line being modified.

Old text - a group of characters, including spaces, which are searched for, either in the pointed line (as in the MODIFY command) or in the file (as in the FIND or LOCATE commands).

One-pass option - assumed if 'O' in parameter list. The one-pass option does not update the physical source file.

Parameter list - initialization information provided when the editor is first executed. Following file specifications, a SEMI-COLON (;) indicates the presence of a parameter list. The mode (A, C, D, or T), one-pass option (O), tab character, margin bell column, key-click (K), and (in text mode - T) 'no shift inversion' (S) and 'no word wrap-around' (L) may be set.

Pointed line - a pointer (>) in the left hand margin is used to reference lines for modification by command. The line to the right of the pointer is the pointed line.

Physical scratch file - specified (or implied SCRATCH/TXT) output file.

Physical source file - specified input file.

Pseudo-ENTER - the key marked DEL (always shifted) is referred to as the Pseudo-ENTER key. If pressed in the first column of the command line, one line of text will be fetched from the source file.

If pressed while entering a command, a user defined command string separator will be entered.

In comment mode, if pressed on any but the bottom screen line or command line, it will cause the cursor to be positioned to the comment field of the following line and that field will be erased.

In all other modes, the Pseudo-ENTER key causes a new line to be inserted so that data entry may proceed in the same area of the screen. If pressed on the last screen line, the Pseudo-ENTER key simply places the cursor on the command line.

Scratch file - at any point in time, the logical scratch file is the output file.

Screen line - any of the lines on the screen which may be referenced by the command pointer. The command line (bottom line) is not, therefore, included.

Shift key inversion - reverse the function of the shift key for all alpha characters so that, in lower case, alpha characters will appear upper case.

Source file - originally this is the input file specified at initial execution. The term source file refers to the current input file; thus, at any point in time, the logical source file may be either the specified input file or the file specified as the scratch file.

Text mode - assumed by a 'T' in the parameter list. No tab stops are set (tabs may be set during execution). The SEMI-COLON (;) is the assumed tab character (the tab key character may be changed in the parameter list or during execution). No shift key inversion is performed (this may be selected in the parameter list with an 'S'). Word wrap around is performed (this feature may be turned off by an 'L' in the parameter list).

Word - a word is defined as any group of less than 70 characters preceeded by a space.

Word wrap-around - a feature of text mode. During data entry a space to the right of the margin bell column causes an immediate carriage return. If this occurs on a screen line, a line insert is performed so that data entry may proceed at the same area of the screen. If a character is typed over the last column of the screen, the last word is removed, a line insert performed and the removed word is placed at the beginning of the inserted line where data entry may proceed. If a modify command causes the line to become longer than 79 characters, the trailing characters, including the last word on the line, will be moved to a new line which will be inserted below the original line. Control will then return to the command line.

2.9 Command List

The full set of EDITOR commands are listed below. All legal combinations of options are included. [SP] represents a space. [ENT] represents the "ENTER" key. [#] represents the field number, a number in the range 1-9. Commands may be either upper or lower case.

:[NEW TAB KEY][ENT] - Replace the old tab key character with the new one. The default for the tab key character is the space bar.

:[OLD REPLACE OPERATOR][SP][NEW REPLACE OPERATOR][ENT] - Replace the old modify-replace character with the new one. The default for the modify-replace character is the "less than" (<) symbol.

:[OLD INSERT OPERATOR][SP][NEW INSERT OPERATOR][ENT] - Replace the old modify-insert character with the new one. The default for the modify-insert character is the "greater than" (>) symbol.

:[OLD APPEND OPERATOR][SP][NEW APPEND OPERATOR][ENT] - Replace the old modify-append character with the new one. The default for the modify-append character is the "backslash" (\).

::[string][ENT] - Force a line beginning with a colon to be enterec into the text of the file.

:A[ENT] - Copy the pointed line to the bottom of the screen and roll the screen up one line and move the pointer up one line.

:A*[ENT] - Copy the pointed line to the bottom of the screen and roll the screen up one line without moving the pointer. This allows defining a command that does multiple ":A*"s to copy multiple lines.

:B[ENTER] - BYPASS the end of file which may be caused by RECORD FORMAT, PARITY, or RANGE errors.

:C[ENT] - Copy the pointed line to the bottom of the screen, delete the pointed line, and key in a new line.

:CH[ENT] - Display the current tab key character and MODIFY operators.

:D*[ENT] - Display the previously defined string used by DELETE, FIND, LOCATE, and QUERY.

:D[ENT] - Delete the pointed line and key in a new line.

:D[SP][ENT] - Delete up through a previously defined string (defined by a previous DELETE, FIND, LOCATE, or QUERY) in the pointed line.

:D[SP][string][ENT] - Delete up through the specified string in the pointed line.

:D[#][SP][ENT] - Delete up through a previously defined string (defined by a previous DELETE, FIND, LOCATE, or QUERY) in the specified field of the pointed line.

:D[#][SP][string][ENT] - Delete up through the specified string in the specified field in the pointed line.

:E[ENT] - END the EDIT by copying the modified file back to the original source file.

:E/[ENT] - END the EDIT by copying the modified file back to the original source file up to the bottom line displayed on the screen. This deletes the remainder of the file.

:E\[ENT] - END the EDIT by copying the modified file back to the original source file starting with the top line of the screen up to the end of the file. This deletes the preceding portion of the file.

:EO[ENT] - Display the file forwards (rolling the screen up) to the end of file. The "DISPLAY" key will stop the display until it is released. The "KEYBOARD" key will terminate the display and return to the command line.

:E-[ENT] - Display the file backwards (rolling the screen down) to the beginning of the file contained in memory. The "DISPLAY" key will stop the display until it is released. The "KEYBOARD" key will terminate the display and return to the command line.

:E*[ENT] - Display the last line of the file at the bottom of the screen (immediately above the COMMAND LINE).

:EX[SP][DOS command string][ENT] - END the EDIT by copying the modified file back to the original source file, then execute the DOS command string.

:F*[ENT] - Display the previously defined string used by FIND, DELETE, LOCATE, and QUERY.

:F[SP][ENT] - Find a line starting with the previously defined string.

:F[SP][string][ENT] - Find a line starting with the defined string.

:F[#][SP][ENT] - Find a line starting in the specified field with the previously defined string.

:F[#][SP][string][ENT] - Find a line starting in the specified field with the defined string.

:G[ENT] - Display the next screen-full of lines.

:G-[ENT] - Display the prior screen-full of lines.

:G[nnnnn][ENT] - Roll up the screen [nnnnn] lines where the number may range from 1 to 65,535 lines. This will stop at the end of file.

:G-[nnnnn][ENT] - Roll down the screen [nnnnn] lines. This will stop if the beginning of the memory buffer is reached.

:GA[nnnnn] - Display the [nnnnn]th line of the file. This will roll forward or backward through the file depending on the current location in the file.

:I[ENT] - Insert by keying in a new line immediately below the pointed line.

:I[SP][string][ENT] - Insert the specified string immediately below the pointed line.

:K[ENT] - Turn on the key click.

:KI[ENT] - Turn off the key click (Key click Invert).

:L*[ENT] - Display the previously defined string used by LOCATE, DELETE, FIND, and QUERY.

:L[ENT] - Roll up the screen one line. At the end of the file, this will cause the file to wrap around to the beginning.

:L[SP][ENT] - LOCATE a line containing the previously defined string.

:L[SP][string][ENT] - LOCATE a line containing the defined string.

:L[#][SP][ENT] - LOCATE a line containing the previously defined string in the specified field.

:L[#][SP][string][ENT] - LOCATE a line containing the defined string in the specified field.

:M*[ENT] - Display the previous MODIFY command line.

:M[ENT] - MODIFY the pointed line using the previous MODIFY commanc line.

:MV[ENT] - MODIFY with VERIFY the pointed line using the previous MODIFY command line.

:M[#][ENT] - Repeat the previous MODIFY command line applied to the specified field.

:MV[#][ENT] - Repeat the previous MODIFY command line with VERIFY appliec to the specified field.

:M[SP][modify string][ENT] - MODIFY the pointed line as specified by the modify string.

:MV[SP][modify string][ENT] - MODIFY with VERIFY the pointed line as specified by the modify string.

:M[#][SP][modify string][ENT] - MODIFY the specified field in the pointed line as specified by the modify string.

:MV[#][SP][modify string][ENT] - MODIFY with VERIFY the specified field in the pointed line as specified by the modify string.

Modify strings are of the following formats. The default replace operator is the "less than" symbol (<). The default insert operator is the "greater than" symbol (>). The default append operator is the "backslash" symbol (\). [string1] and [string2] are optional.

[string1][replace operator][string2] - Replace string1 with string2.

[string1][insert operator][string2] - Insert string2 immediately following string1.

[string1][append operator][string2] - Truncate the line immediately following string1 and append string2.

:O[ENT] - Return to the Operating System without updating the original source file.

:OX[SP][DOS command string] - Return to the Operating System without updating the original source file and execute the given DOS command string.

:Q*[ENT] - Display the previously defined string used by QUERY, DELETE, FIND, and LOCATE.

:Q[SP][ENT] - QUERY (setting a succeed or fail condition) if the previously defined string is contained within the pointed line.

:Q[SP][string][ENT] - QUERY if the given string is contained within the pointed line.

:Q[#][SP][ENT] - QUERY if the previously defined string is contained within the specified field of the pointed line.

:Q[#][SP][string][ENT] - QUERY if the given string is contained within the specified field of the pointed line.

:RC[ENT] - Set RPG Calculations tab stops (columns 6, 18, 28, 33, 43, 49, 54, and 60).

:RE[ENT] - Set RPG Extension tab stops (columns 6, 11, 19, 27, 33, 36, 40, 46, 52, and 58).

:RF[ENT] - Set RPG File tab stops (columns 15, 24, 33, 40, 54, 66, and 70).

:RH[ENT] - Set RPG Header tab stops (columns 6 and 15).

:RI[ENT] - Set RPG Input tab stops (columns 6, 15, 21, 44, 53, 59, and 65).

:RL[ENT] - Set RPG Line tab stops (columns 6, 15, and 20).

:RO[ENT] - Set RPG Output tab stops (columns 6, 15, 23, 32, 38, 40, and 45).

:RS[ENT] - Set RPG Summary tab stops (columns 6, 14, and 23).

:SB[ENT] - SCRATCH BELOW deletes all the lines on the screen from the pointed line down through the bottom line line and then allows keying in a new line at pointed line.

:SC[ENT] - SCRATCH deletes all the lines on the screen from the top line down through the pointed and then allows keying in a new line at the pointed line.

:T[ENT] - Set TAB stops by displaying a line of column numbers and allowing the user to space across setting tabs by typing non-blanks. Up to 20 tabs may be set.

:T[SP][nn][,nn]...[ENT] - Set tab stops to the columns specified by [nn][,nn]... where [nn] ranges from 1 to 79. Up to 20 tabs may be set.

:TA[ENT] - Set TAB stops for Assembler (columns 9, 15, and 30).

:TD[ENT] - Set TAB stops for Databus or Datashare (columns 10 and 20).

:TS[ENT] - Set TAB stops for SNAP (columns 11, 21, and 38).

:U[ENT] - UNCONDITIONAL skip over the following command in the user defined command string.

:U[n][ENT] - UNCONDITIONAL skip over [n] (1 to 9) following commands in the user defined command string.

:V[SP][nn][ENT] - Split the pointed line at column [nn], inserting the remainder of the line immediately below the pointed line.

:W[ENT] - Concatinate the line below the pointed line to the pointed line. If word wrap-around is in effect, the pointec line is assumed to have a trailing space.

:X[ENT] - Change to TEXT mode with word wrap around and no shift inversion and then set tab stops (as in :T above).

:XI[ENT] - Change to Assembler mode with shift inversion and no word wrap around and then set tab stops (as in :T above).

:Z[ENT] - Skip over 1 command after the following command in the user defined command string if the following command fails.

:Z[n][ENT] - Skip over [n] (0 to 9) commands after the following command in the user defined command string if the following command fails.

:Z*[ENT] - Terminate execution of the user defined command string and return control to the bottom key in line.

NOTE: The following commands, though refering to zero, actually refer to all the user definable commands zero through nine.

:0[ENT] - Execute user defined command zero.

:0*[ENT] - Display user defined command zero.

:0[SP][user defined command string][ENT] - Define user defined command zero. The command string consists of one or more EDIT commands separated with a Pseudo ENTER (shift DEL) character. The colon immediately following the Pseudo ENTER character is optional.

:0<[ENT] - Insert user defined command zero into the file text immediately below the pointed line.

:0>[ENT] - Define user defined command zero as the pointed line on the screen. The combination of this and the above command allow user defined commands to be saved in the text of the file and to be EDITed.

:00[ENT] - Display the user defined commands zero through nine.

:99[SP][user defined command string][ENT] - This is the initialization command which is executed when the EDIT is started. It may be overridden by pressing the KEYBOARD key while bringing up EDIT. This command may only be used in the definition file (/DEF) and not during the actual EDIT.