March 1981

MEDIT USER MANUAL

by

Robert N. Goldberg

DCS-TR-105

Department of Computer Science
Rutgers University
Hill Center
Busch Campus
New Brunswick, NJ 08903
**** GETTYSBURG ADDRESS **** 1

*BF*
Four score and seven years ago, our fathers brought forth on this continent a new nation conceived in liberty and dedicated to the proposition that all men are created equal.

*EF*
MEDIT USER MANUAL

A Machine Independent Text Editor

First Edition - Draft of September 17, 1981

This document describes version 1.0 of the software

Copyright (C) 1981 Robert N. Goldberg
Acknowledgements

The command language for most of MEDIT's line editing commands comes from the SOS text editor, which was designed and written at Stanford University during the 1960's for the PDP-10. The first CRT screen editor also had its origins at Stanford and led to the development of the TVEDIT editor. MEDIT's full screen editing mode is an extension of SOS intraline editing to provide a TVEDIT-like screen editor while retaining the original flavor of the SOS intraline editing command language. MEDIT also contains significant new commands and features not present in either SOS or TVEDIT. Some capabilities were suggested by MIT's EMACS editor.

Aside from its obvious heritage of command language and editing capabilities from previous successful editors, MEDIT is an original machine independent editor which I designed and implemented at Rutgers University. The internal representation of edit buffers, the handling of temporary files, the highly efficient checkpointing features, the implementation of terminal support through simulation of a prototypical terminal, and the editor interface to user supplied MAINSAIL programs are all original.

Any person or organization which receives or uses this software must include this acknowledgement of authorship and history in documentation and code associated with this software product and its descendants, in addition to any other restrictions and obligations agreed upon.

Bob Goldberg, March 1981
# Table of Contents

## INTRODUCTORY MANUAL

1. Introduction

2. Invoking MEDIT
   2.1. Creating a New File
   2.2. Editing an Existing File

3. Line Editing Tutorial
   3.1. Creating a File
   3.2. Exiting MEDIT
   3.3. Editing a File
   3.4. P - Print command
   3.5. I - Insert command
   3.6. D - Delete command
   3.7. E - End Edit command

4. More Line Editing Commands
   4.1. Giving Line Addresses to MEDIT commands
   4.2. R - Replace command
   4.3. + - Plus command
   4.4. - - Minus command
   4.5. F - Find command
   4.6. S - Substitute command
   4.7. A - Alter command
   4.8. Entering Full Screen Editing Mode
   4.9. X - eXtend command
   4.10. C - Copy command
   4.11. T - Transfer command
   4.12. L - List command
   4.13. N - reNumber command
   4.15. G - Go command
   4.17. - invoke indirect command file
   4.18. J - Justify commands
   4.19. M - Mark Page command
   4.20. K - Kill Page Mark command

5. Full Screen Editing Tutorial
   5.1. Creating a file
   5.2. Exiting From Full Screen Mode
   5.3. Editing in Full Screen Mode
   5.4. Useful Switches for Full Screen Editing

6. Text Justification
   6.1. J - Join
   6.2. JL - Justify Left
Table of Contents

6.3. JR - Justify Right
6.4. JC - Justify Center
6.5. JW - Justify Without padding
6.6. JU - Justify

REFERENCE MANUAL

7 Invoking MEDIT
7.1. Creating a file
7.2. Editing an existing file

8 Line/Page Numbers
8.1. Files With More Than One Page
8.2. Syntax of Line Addresses and Ranges
8.3. Line and Page Number Arithmetic

9 Inserting, Deleting, and Replacing Lines
9.1. D - Delete command
9.2. I - Insert command
9.3. R - Replace command

10 Printing the File
10.1. P - Print command
10.2. L - List command
10.3. V - Case Inversion commands

11 Writing Out the File and Exiting
11.1. E - End Edit command
11.2. W - Write command
11.3. G - Go command

12 Manipulating Line and Page Numbers
12.1. . - Set the Current Position
12.2. M - Mark Page command
12.3. K - Kill Page Mark command
12.4. N - reNumber command
12.5. NP - reNumber over Pages command

13 Text Search and Substitution
13.1. F - Find command
13.2. Find Options
13.3. Responses in Find Decide Mode
13.4. S - Substitute command
13.5. Substitute Options
13.6. Responses in Substitute Decide Mode
13.7. Special Matching Characters
13.7.1. Special Matching Characters for the Search String
13.7.2. Special Characters for the Replacement String
13.7.3. Some Useful Example Patterns
13.8 Hints on Using F and S
# Table of Contents

14 Alter Mode and Full Screen Editing
- 14.1. A - Alter command
  - 14.1.1. Single Line Mode
  - 14.1.2. Window Mode
- 14.2. X - eXtend command
- 14.3. Alter Mode Commands
  - 14.3.1. Modifiers
  - 14.3.2. Cursor Positioning Operations
  - 14.3.3. Insert/Delete Operations
  - 14.3.4. Change Operations
  - 14.3.5. Justification Operations
  - 14.3.6. Miscellaneous Commands
- 14.4. Defining Alter Command Macros
- 14.5. Setting Up Special Key Macros
- 14.6. Terminal Support

15 Copying Text and Multiple File Editing
- 15.1. B - Buffer command
- 15.2. C - Copy command
  - 15.2.1. Copying from the current file buffer
  - 15.2.2. Copying from another file or buffer
- 15.3. T - Transfer command
- 15.4. Buffer Examples

16 Programming New MEDIT Commands
- 16.1. - Invoke Indirect Command File
- 16.2. Automatic Indirect Command Files
- 16.3. : - Comments
- 16.4. O - On command
- 16.5. Indirect Command File Examples
- 16.6. U - Invoke User Program Module

17 Customization Using Global Switches
- 17.1. / - Set Switch command
- 17.2. = - Give Switch Value command
- 17.3. Initial Switch Settings
- 17.4. String Syntax
- 17.5. Table of Global Switches

Appendix I. Table of ASCII Characters
Appendix II. MEDIT Command Summary
Appendix III. MEDIT Compared to Other Editors
Appendix IV. Temporary File Implementation
Appendix V. User Module Library
Appendix VI. Terminal-Dependent Information
- 17.6. Visual 200 Information
- 17.7. DEC VT52 Information
- 17.8. Health (Zenith) 19 Information
- 17.9. Datamedia 2500 Information
Table of Contents

17.10. Delta Data 7000 (alias NIH 7000) Information
17.11. Infoton 400 Information

Index
INTRODUCTORY MANUAL
CHAPTER 1
INTRODUCTION

A text editor is a program used to create and modify text files stored in the computer. Text editors are useful for creating and modifying computer programs, documentation, data files, letters, and many other kinds of text.

MEDIT is a text editor written in MAINSAIL. It can run on any computer for which a MAINSAIL implementation exists, and on any terminal type. Both with regard to the computer on which it runs and the terminal used, it is machine-independent.

The basic line editing commands have been adopted from the powerful SOS text editor for the PDP-10. The intraline editing capability of SOS has been extended in MEDIT to a video editor which keeps the screen of the terminal constantly updated to reflect the true state of the file as modifications are made. Video editing greatly increases user productivity, especially in editing of English and other unrepetitive text. On hard copy terminals and on unsupported CRT terminals, MEDIT cannot provide video editing, but it knows how to utilize full duplex capabilities to provide character by character modifications within a line of text. The commands for making changes within a line are the same for all terminals.

This document contains both an introductory user's manual and as a reference manual to MEDIT. The introductory manual introduces the user to the editor while the reference manual provides complete, detailed specifications of the editor's features. The style of the introductory manual is informal to aid the learning process. For clarity, when examples of interaction with MEDIT are given, those parts that were typed by the user will be shown in italics while those parts typed by MEDIT will be in a more standard font.

Major Features of MEDIT

Text Search MEDIT can search the file for occurrences of strings of characters. It provides the ability to search for several patterns simultaneously, where patterns may include both the actual characters to search for and also special characters to allow searches for classes of characters when necessary.

Global Substitution MEDIT can substitute one string of characters for another in selected portions or throughout the entire file being edited. In addition to simple substitution, MEDIT provides pattern substitution and the ability to specify multiple sets of substitutions in a single command. This feature makes it possible to perform substitutions such as changing all "X"'s to "Y"'s and all "Y"'s to "X"'s, which could not be done easily without multiple simultaneous substitutions.

Decide Mode If the optional "decide mode" is specified for a Find or
Introduction

Substitute command, the user can accept or reject text substitutions and searches, possibly performing some additional action on the lines found. This feature can be used to prevent mistakes during global substitution commands, and to use the Find command to find lines on which editing commands are to be performed. It is especially useful for making repetitive context sensitive changes interactively.

Text Copying

MEDIT allows copying from one portion of the file being edited to another portion. It also allows copying from other files on the system, and provides a convenient way to "scan" the file from which the copying is being done.

Text Justification

MEDIT provides commands for left and right justifying, centering, justifying and padding with blanks, and joining lines together.

Case Inversion

Text can be converted to upper case, lower case, or the opposite case, on a line-by-line or character-by-character basis.

Indirect commands

MEDIT can be told to accept text editing commands from a previously created command file. This feature is useful for frequently repeated sequences of commands. A range of lines to be edited can be passed to the command file, if desired, at the time the indirect command file is invoked. Loops and conditional commands may be used in indirect files.

User Modules

A user who is familiar with the MAINSAIL programming language may easily write a MAINSAIL module (program) and call it from MEDIT to perform processing on the file. Thus, the full power of an ALGOL-like language may be used to perform complex editing tasks. As such modules are written, a library of them will be collected and made available to all users.

External Programs

MEDIT provides a command to exit to an external program directly. On DEC systems, this same facility may be used to repeat the most recent COMPILE, LOAD, DEBUG, or EXECUTE command issued at monitor level.

Editing History

MEDIT provides an option to keep a file editing history at the beginning of the file in an appropriate format for the type of file being edited. This automatically generated comment line contains the date, the time, and a short comment supplied by the user. The format of the comment line may be set according to the type of file being edited.

Full ASCII

MEDIT provides a special "flag" character to allow input of the full 128 ASCII character set. Without this feature, some of the ASCII characters would be difficult to enter into a file (such as $C on DEC systems).

Multiple Buffers

MEDIT provides the option to edit more than one file at a time.
in separate edit buffers. Lines can be copied or transferred between buffers easily to achieve editing across files. Multiple buffers provide added flexibility in editing. For example, buffers can be used to alternately work on two files or to make a quick change to an auxiliary or command file while editing the main file.

Customization

Many user-specified switches are provided to allow the customization of the editor to individual tastes. These switches are set through commands to the editor which may be given explicitly or included in an initial command file that is read automatically whenever MEDIT is started. It is possible to have certain switches set up automatically depending on the type of the file being edited or the type of terminal being used.
CHAPTER 2
INVOKING MEDIT

Using MEDIT you may create a new file or edit an existing file on the system. When you are done editing the file, you leave MEDIT with a special command. It then writes out your file in the newly edited form. In this chapter, we will describe the commands used to create a new file and edit an existing file with MEDIT. Chapters 3 and 4 introduce the commands MEDIT has for editing lines of the file on a hard copy terminal. Chapter 5 describes the most basic commands for full screen editing of a file on a CRT terminal.

If you are using a supported CRT terminal and you have told the operating system its type, you may optionally tell MEDIT to go immediately into full screen editing mode using the \("WINDOW\) switch on the command line that invokes MEDIT. If the \("WINDOW\) switch is not used, MEDIT will start in line editing mode, which works on all terminal types.

On some systems, it may be possible to invoke MEDIT using monitor commands such as EDIT and CREATE. If so, MEDIT will understand these commands correctly. Consult your system's documentation for details on how to specify MEDIT as your default editor to be invoked by these commands.

When you edit a file using MEDIT, it does not change the original copy of your file; instead, the editing is performed on a temporary copy of the file which is called a buffer. When you have finished editing, you will normally issue a command that tells MEDIT to write the contents of the buffer out to form a new version of the file on disk, with all the changes you have made. Occasionally, you may use the buffer to create a file with a different name, or ask MEDIT to discard the changes you have made without writing them out to a file.

2.1. Creating a New File

To create a new file, use the command

```
@EDIT/CREATE new-file-name
```

or

```
@EDIT/CREATE\ WINDOW new-file-name
```

where "new-file-name" is a file name you make up and the "@" sign is a prompt typed by the system. On some systems it may be necessary to start MEDIT without specifying a file name. In this case it will respond by prompting.

---

1 Chapter 5 explains how to make this action the default if you desire.

2 See the user's guide for your system if you don't know the allowable forms that file names may take.

3 Your system may use a different prompt, such as "#".
Invoking MEDIT

File: LETTER.TXT

to which you should respond by typing the name of the file you wish to create and the
switch "/CREATE", e.g.

File: LETTER.TXT /CREATE

or

File: LETTER.TXT /CREATE /WINDOW

Precisely what happens next depends on whether or not you used the /WINDOW
switch, but basically MEDIT is going to prompt you to type in lines of the file. For
example, suppose you used the command

@MEDIT /CREATE LETTER.TXT

MEDIT will reply

[LA36] Input: LETTER.TXT

The "[LA36]" shows what kind of terminal MEDIT thinks you are using, in this case a
DECwriter LA36 hard copy terminal. Next it will prompt you with:

00100

and pause for you to type in the first line of text. The number "00100" is the line
number assigned to the first line by MEDIT. The numbers go up by 100's so that you can
insert new lines between existing lines if necessary. You now type in the first line of the
text

00100  This is an example of a line of text.

When you wish to start a new line, enter a carriage return by hitting the "RETURN" or
"ENTER" key on your terminal keyboard (indicated by <CR> from now on). This key is
usually on the right side of the keyboard. MEDIT will respond by going to a new line and
prompting

00200

for you to type the next line of text.

Mistakes that you make while typing in a line can often be corrected as you type by
using one or more special keys on the terminal. The editing keys depend on the particular
system you are using, e.g. on DEC systems the "RUBOUT", "DELETE", or "DEL" key (indicated
by <RUBOUT> from now on) may be used to delete the previous character typed and "U"
may be used to delete the line being typed4.

When you are done entering text, or if you wish to correct an error you have made
on a previous line, use the <ESC> key, labeled "ESCAPE" or "ALT MODE". Use of this key
will let you "escape" from "input" mode and go into editing mode. At this point, you may
type line editing commands as described in chapter 3 (if you did not use /WINDOW) or
window editing commands as described in chapter 5 (if you did use /WINDOW).

4Later on in this manual you will learn about commands for renumbering the lines in a file and about ways
to make MEDIT choose an increment other than 100 between successive line numbers, but for now you can
use the default values. MEDIT can also be made to suppress printing of line numbers.

5Other simple correcting functions are available on DEC systems also, such as redisplaying the current line.
Refer to the user's guide on your system for information about these special editing characters.
Invoking MEDIT

2.2. Editing an Existing File

To edit an existing file, use the command

@MEDIT file-name

or

@MEDIT\!WINDOW file-name

On some systems it may be necessary to start MEDIT without specifying a file name
and then enter the name of the file you wish to edit in response to a prompt from MEDIT.

E.g.

File: LETTER.TXT

or

File: LETTER.TXT\!WINDOW

For example, say you gave the command

@MEDIT LETTER.TXT

If a file with that name exists then MEDIT will reply

[LA36] Edit: LETTER.TXT

If a file with that name does not exist, MEDIT will inform you and ask if you want to
create it:

LETTER.TXT does not exist, shall we create it?

If you reply "y" then MEDIT will act as if you had used the /CREATE switch. If you reply
"n" then MEDIT will return you to the operating system where you may reissue a command
to edit an existing file.

What happens now depends on whether you used the /WINDOW switch or not.
Chapter 3 describes the commands that are available in line editing mode, if you did not
use /WINDOW; chapter 5 describes the full screen editing commands that are available if
you did.
CHAPTER 3
LINE EDITING TUTORIAL

This chapter describes how to use MEDIT to perform line editing, which is available on all terminal types. If you did not specify the /WINDOW switch when you called MEDIT, it will go into line editing mode automatically when it starts. In line editing mode, MEDIT prompts for commands with an asterisk "*" and waits for you to type a command line, terminated by <CR>. It then processes your command and prompts for the next command.

3.1. Creating a File

When MEDIT is used to create a file, it will initially prompt you to type lines of the file before prompting with "*" for commands. An example editing session in which a file is created is:

@MEDIT/CREATE FOO.TXT
[LA36] input: FOO.TXT
00100 Now is the time for all good men to
00200 come to the aid of their country.
00300 *

Notice that MEDIT prints "*" when you use the <ESC> key to get out of input mode. It then prompts you to type a command with "*".

3.2. Exiting MEDIT

If you are satisfied with the file at this point and you do not want to change or add to it, use the E command to end the edit and write the file on disk:

*E
[F00.TXT]
@

MEDIT types the file name you gave it when the edit began (see above), and then exits.

3.3. Editing a File

To edit a file that already exists, issue a command like:

@MEDIT FOO.TXT
[LA36] Edit: FOO.TXT
*

MEDIT retrieves the file and prompts with "*" for you to type a command.
Line Editing Tutorial

Four very simple line editing commands you can type to MEDIT when it is in line editing mode and is prompting you with "*" are:

- **P** - to print lines
- **I** - to insert new lines
- **D** - to delete lines
- **E** - to end the edit and write the file

3.4. **P** - Print command

The P (print) command prints lines on your terminal. To print line 100, you use the command

```
*P100
```

and to print lines 100 through 500, you use the command

```
*P100:500
```

Often you will want to print the entire file, which you can do by using the command

```
*PEF
```

3.5. **I** - Insert command

The I (insert) command lets you insert new lines into the file. For example,

```
*I500
```

will insert line number 500 if line 500 does not already exist, or it will insert a line after line 500 if line 500 does already exist. The I command never causes existing lines to be replaced. This feature of the I command prevents accidental deletions of existing lines of text.

3.6. **D** - Delete command

The D (delete) command deletes lines from the file. For example, the command

```
*D150
```

will delete line 150 from the file, and

```
*D100:300
```

will delete lines 100 through 300, inclusive. Question: What do you think

```
*DEF
```

will do?

---

6. The R (replace) command described in the next chapter can be used to replace an existing line, if desired.
3.7. E - End Edit command

The E (end edit) command writes out the file to disk and ends the editing session. This command should be used when you are satisfied with the text and you do not wish to edit it further. MEDIT will type out the name of the file being written to disk and return you to the operating system.

*E

[FDD0.TXT]
E
CHAPTER 4

MORE LINE EDITING COMMANDS

The four basic commands described in Chapter 3 enable you to edit the file as you desire but more powerful commands exist which make editing easier. This chapter describes the simple forms of most MEDIT line editing commands other than "P", "I", "D", and "E" (which were covered in chapter 3). Later chapters describe these commands in full detail for reference purposes and advanced use.

If you have access to a video terminal, you may wish to skip this chapter and go on to chapter 5 to find out how to do full screen editing.

4.1. Giving Line Addresses to MEDIT commands

Most MEDIT line editing commands perform an action on a line or range of lines. For example, the P (print) and D (delete) commands introduced in chapter 3 accept a range of lines to be printed and deleted, respectively. The I (insert) command accepts a single line address at which to insert a new line.

The simplest way to give the address of a line is to specify its line number, e.g. "D200" to delete line 200. The simplest way to give a range of line addresses is to give the first and last line number of the range separated by a colon, e.g. "D200:800" to delete lines 200 through 800, inclusive.

Some special symbols may be used in place of line numbers. For example the symbol "*" stands for the first line and "*" stands for the last line. Thus, "I*" means insert lines starting after the last line. The special symbol "EF" means "entire file". Thus, "PEF" prints the entire file.

The special symbol "." stands for the current line. The current line is the line on which the most recent command performed its action. Thus, if you used the command "P500" to print line 500, and then used the command "D.:", lines 500 through the last line would be deleted.

Other more advanced ways of specifying addresses and ranges of lines are available and are covered in later chapters. Some of the advanced ways let you create more than one "page" within the file, each with its own set of line numbers! If you use MEDIT to edit a large file that does not have line numbers already, MEDIT will put lines of the file on separate pages (since it will run out of line numbers on a single page). In this case, you will have to refer to chapter 8 to learn how to deal with multiple page files.
More Line Editing Commands

4.2. R - Replace command

To replace a line, e.g. line 500, you can use the command "D500" to delete it and then "I500" to insert a new line there. This operation is so common that a special command, R (replace) exists for doing it. For example:

```
*PEF
00100  C   PROGRAM TO COMPUTE MATRIX PRODUCT
00200   INTEGER I, J
00300   TYPE 10
00400   10  FORMAT ( ' ENTER NUMBERS:' )
00500   CALL READM
00600   CALL PROD
00700   STOP
00800   END
*R500
00500   CALL READM
*```

You may replace more than one adjacent line by giving a range to the R command, e.g. "R200:500". The lines will be deleted and an I command will be performed at the first line in the range, in this case "I200".

4.3. + - Plus command

Often you will want to print the line after the current line in the file. To print the next line you can use the "+" (Plus) command. MEDIT will print the next line. On some systems you may invoke the Plus command by depressing the "LINE FEED" or "LF" key (denoted <LF> from now on). MEDIT will respond immediately without waiting for you to type <CR>.

4.4. - - Minus command

To print the previous line, use the "-" (Minus) command. On some systems you may invoke the Minus command by depressing the <ESC> key. <ESC> is also used to terminate insert mode and go into line editing mode. When you are in line editing mode, depressing the <ESC> key will print the line before the current line.

4.5. F - Find command

The F (find) command finds occurrences of patterns in the file. You give it a pattern to look for and it finds a line on which that pattern occurs. For example, suppose you want to find a line on which "FORMAT" appears in the file. You use the command

```
*FORMAT$EF```

to find the first occurrence of "FORMAT". If MEDIT finds a line containing "FORMAT", it will type out that line. If it cannot find a line containing "FORMAT", it will type

---

7 Recall that the system prints "*" when you use the <ESC> key.
More Line Editing Commands

%Search Fails (first: last)
where "first" and "last" are the first and last lines which MEDIT searched. To find the next
occurrence, you use the F command with no text argument, i.e.

*F

When no pattern or range of lines is given to the F command, it starts searching on the
line after the current line and continues the search specified by the previous F command.

To find a pattern somewhere after the current line of the file, use the F command
with a pattern but no range. For example,

*FORMAT$S

will search for "FORMAT" starting at the next line of the file and continuing until the end
of the file if necessary.

There are many advanced features of the F command other than those described
here. Refer to chapter 13 for a complete description of the F command.

4.6. S - Substitute command

The S (substitute) command changes occurrences of a pattern to a specified string.
For example, if you habitually mistype the word "CALL" as "CAKL", you can use the
command

*SCAKL$CALL$EF

to change "CAKL" to "CALL" wherever it occurs throughout the entire file. Any range of
lines may be used in place of "EF" above.

To repeat the most recent S command on a range of lines, e.g. line 100, use the
command

*SS100

4.7. A - Alter command

The line editing commands described thus far perform their action on one or more
lines of the file. To correct mistakes within a line you can use the R command, but that
requires retyping the entire line and not just the mistake. You could also use the S
command to substitute the correction for the mistake, but you have to be very careful to
specify the mistaken part carefully enough so that only it and no other part of the line is
replaced. In addition, if there are two or more mistakes on a line, many S commands may
be required to fix it.

The A (alter) command lets you perform character-by-character correction of a line.
For example, suppose a line reads

00100 This lines has lots of mistakes in it.

You would like to tell MEDIT to delete the extra "e" in "line", replace "x" with "s", and
replace "eak" with "aks". To do so, go into alter mode on this line using the command

*A

More Line Editing Commands

* A100

On a hard copy terminal, MEDIT will type

00100

and pause with the print head above the "_" ready to print the first character. On a supported video terminal, MEDIT will display the entire line (or as much of it as fits on a line of the terminal), position the cursor underneath the first character in the line, and pause:

00100 This is the line being altered.

In either case, MEDIT is now waiting for you to begin typing alter mode commands. These commands are the same commands available in full screen editing mode as introduced in chapter 6. Unlike line editing commands, which generally act on entire lines, most alter mode commands apply to single characters. Some simple alter mode commands you can use are

- space-bar to move the cursor to the next character in the line
- D to delete the next character
- Sc to skip over to character "c", i.e. advance until you are just before "c"
- I to start inserting characters at the current position in the line.
- <CR> to stop altering this line.

In this case, by typing "sedsxdis&sediaw" and then <CR>, you can fix up the line. As you type alter mode commands, the command letters themselves do not appear on the terminal. Instead, the action they cause appears. Try this example yourself to see what happens.

Although this form of editing may at first seem unfamiliar to you, alter mode will soon become natural and easy to use with practice. Many additional alter mode commands exist, most of which are useful on both video and hard copy terminals. They are described in chapter 14 in full detail.

4.8. Entering Full Screen Editing Mode

If you have a supported video terminal, you may use the A command to enter full screen editing mode by omitting the line number in the A command. MEDIT will display a screenful of lines with the current line of the file in the middle of the screen. To put a line other than the current line in the middle of the screen, specify the line number in the A command and use the option ",W" (window), e.g. "A500,W" will put line 500 in the middle of the screen. For example, if you invoke MEDIT without using the /WINDOW switch and wish to go into full screen editing mode immediately, use:

---

8 In fact, full screen editing mode is simply a special form of alter mode in which the entire screen (rather than a single line) is displayed.

9 Remember that "S" stands for using the <ESC> key.

10 There are better ways to perform this edit using commands not yet described.
More Line Editing Commands

`@MEDIT FOO`

[V200] Edit: F00.

*A*

MEDIT will now go into full screen editing mode as if the /WINDOW switch had been used. An introduction to full screen editing is given in chapter 5.

4.9. X - eXtend command

To add something to the end of an existing line of text, you can use the A command to alter the line, advance the pointer to the end of the line, and then use the video editing I command to insert characters there. However, a special command X (eXtend) exists to do this operation automatically. For example

*A100
*00100 This is true.
*X100
00100 This is true. Did you know that?
*A*

After the X command is typed, MEDIT types out the line and pauses for you to type the rest. In this case, the user typed the sentence that appears in italics. When you have typed the rest of the line, type <CR>.

4.10. C - Copy command

Sometimes it is useful to make copies of lines already typed. For example, you may wish to use a line of asterisks to separate parts of a program. Rather than retyping the line each time it is needed, you can use the C (copy) command to copy it. For example

*CPEF
00100 C
00150 C***********************************************************************
00175 C
00200 FUNCTION A(B)
00300 A=SIN(B)*COS(B)
00400 RETURN
*C500,100:175
[3 lines copied]
*CPEF:* 
00500 C
00600 C***********************************************************************
00700 C
*C *

In this case the C command was used to copy to line 500 the contents of lines 100:175.

There are many other ways the C command can be used, such as copying from files other than the one being edited. For a complete description of the C command refer to chapter 15.
More Line Editing Commands

4.11. T - Transfer command

Sometimes one or more lines in a file is misplaced and must be moved. To transfer the lines you could copy them to the right place in the file and then delete the original lines copied. This operation is so common that a special command T (transfer) exists to do it for you. The T command is used the same way as the C command described above. For example:

*PEF
00100         TYPE 10
00200         10 FORMAT(' HELLO')
00300         END
00400         STOP
*T*,300
[1 lines transferred]
*PEF
00100         TYPE 10
00200         10 FORMAT(' HELLO')
00400         STOP
00500         END
*

The T command was thus used to transfer line 300 to the end of the file. A range of lines may be transferred instead of just one line as with the C command.

4.12. L - List command

The L (list) command is like the P command except that the output is sent to the line printer instead of being typed on your terminal. Each page printed on the line printer begins with a heading giving the name of the file, the page number, and possibly other information depending on the implementation. The L command is very handy if you are working on a video terminal and the line printer is nearby.

*LEF
[Output sent to LPT1:]

4.13. N - rNumber command

The N (rNumber) command renumbers the lines in the file. The simplest form of this command is:

*N

which renumbers the entire file starting at 100 and adding 100 to each successive line number. For example
More Line Editing Commands

*PEF
00050 This is an example of a file that
00100 needs to be renumbered because too many
00105 lines were inserted.
00106 There is no way to insert a line above this one
00200 without producing two lines numbered 105!
*NN
*PEF
00100 This is an example of a file that
00200 needs to be renumbered because too many
00300 lines were inserted.
00400 There is no way to insert a line above this one
00500 without producing two lines numbered 105!
*


The V (inVert) commands change text from lower to upper case and vice versa. “VL” changes to lower case, “VU” changes to upper case, and “VV” inverts the case of each letter. For example,

*PEF
00100 THIS FILE WAS TYPED IN ALL UPPER CASE BUT WE
00200 WANT TO CONVERT IT TO LOWER CASE.
*VLI00:200
00100 this file was typed in all upper case but we
00200 want to convert it to lower case.
*

The V commands convert every character in a line. Alter commands can be used to selectively convert single characters in a line to upper or lower case.

4.15. G - Go command

On DEC systems, the G (go) command ends the edit the same way the E command does, but it then causes the most recent system “EXECUTE”, “LOAD”, “COMPILE”, or “DEBUG” command to be repeated. This command is useful when debugging a program since one usually wants to recompile the program after editing it. The G command can also be used to run another program.


The H (help) command types short sections of the MEDIT user manual. The sections are selected by a keyword which may be either a command name or a topic. The command “?H” causes MEDIT to list the topics. The command “H” by itself gives instructions on the H command.
More Line Editing Commands

The "?" command causes MEDIT to type a summary of valid line editing commands, their syntax, and a one line description of each.

The "=?" command causes MEDIT to type a list of switches that can be set to modify its behavior.

4.17. - Invoke Indirect command file

The "@" (indirect command file) command causes MEDIT to read commands from a previously prepared file. When the end of file has been reached, control will return to the terminal. For example,

 *@tab.med

will invoke a command file named "TAB.MED". A range of lines may be passed to the indirect file as follows:

 *@tab.med,100:700

The indirect command file may refer to the range passed to it as "IR".

4.18. J - Justify commands

The commands "J", "JU", "JL", "JR", "JC", and "JW" perform text justification. The "J" command (Join) joins one or more lines together, e.g.

 *P100:200
  00100 This is the first line,
  00200 this is the second.
 *J100
 *P100:200
  00100 This is the first line, this is the second.
 *

"JL", "JR", and "JC" perform left justification, right justification, and centering, respectively. They may be applied to one or more lines, e.g.

 *P100:300
  00100 These lines
  00200 will be
  00300 centered
 *JC100:300
 *P100:300
  00100 These lines
  00200 will be
  00300 centered
 *

"JU" and "JW" perform justification by rearranging lines of text to make them as full as possible. "JU" will pad the resultant lines with blanks to make them align evenly at both
the left and right margins, while “JW” will not. These commands are described in more
detail in chapter 6.

4.19. M - Mark Page command

The M (mark page) command inserts a "page mark" before the line specified. A file
may be divided into more than one logical page, each of which has its own set of line
numbers. Each chapter of a book or subroutine of a program might be placed on a
separate page of the file. For example:

```
*PEF
00100 This is the first line.
00200 This is the last line.
*M200
*PEF
00100 This is the first line.
  Page 2
00200 This is the last line.
```

The use of multiple page files is best avoided until you have become an experienced
MEDIT user.

4.20. K - Kill Page Mark command

The K (kill page marks) command kills (deletes) page marks in the file. No text is
deleted by this command. Only the mark between pages of the file is deleted. For
example,

```
*PEF
00100 This is the first line.
  Page 2
00200 This is the last line.
*K12
*PEF
00100 This is the first line.
00200 This is the last line.
```

It is usually necessary to renumber the pages to be combined to prevent out of order line
numbers on the combined page. If a K command causes line numbers to become out of
order, MEDIT will print the warning

```
% Line numbers Out of Order
```

Use the N command to renumber the file.

If you edit a large file and MEDIT creates more than one page, you may remove all
the page marks by using the commands
More Line Editing Commands

*K/2:/*
*N10

where "N10" renumbers using an increment of 10, rather than 100, between adjacent line numbers.
CHAPTER 5
FULL SCREEN EDITING TUTORIAL

This chapter introduces full screen editing in MEDIT. If the /WINDOW switch is
given when MEDIT is invoked, it will start up in full screen editing mode. Full screen
editing is possible only if you are using a supported video terminal (see appendix VI for a
list of supported terminals and special information about them).

5.1. Creating a file

When MEDIT is invoked to create a file and /WINDOW has been specified, it
behaves as follows:

@MEDIT|CREATE|WINDOW FOO.TXT
[V200] Input: FOO.TXT

MEDIT then blanks the screen and initializes it with:

```
***** FOO.TXT *****
*BF*
00100
*EF*
```

with the cursor on line 100. You may now begin to type into the file. Notice that the
name of the file appears on the top line of the screen and the "I" on the top line indicates
that you are inserting characters into the file. The lines marked *BF* and *EF* indicate,
respectively, the beginning and end of the file.

5.2. Exiting From Full Screen Mode

When you have finished typing in text, or when you wish to correct a mistake you
have made on a previous line, use the <ESC> key to leave insert mode. The "I" at the top
of the screen will now disappear to indicate that MEDIT is going to interpret letters you
type as commands, rather than inserting them into the text. If you simply wish to write
out the file at this point, use the "E" command to leave full screen editing mode and enter
line editing mode. Then use the line editing E command to write out the file and exit from
MEDIT.

\[11\] Full screen mode may also be entered from line editing mode using the A (alter) command as described
in section 4.6.
Full Screen Editing Tutorial

***** F00.TXT *****

*BF*
00100 *This is all I want to type into this file.*
00200 *I will now use the escape key followed by "E"
00300 *and MEDIT will prompt with a "*" below
*E<CR>
[F00.TXT]
@

Notice that the first "E" command does not appear on the screen. It causes MEDIT to leave full screen editing mode and prompt with a "*" for you to type a line editing command. In this case the line editing E command is used to write out the file and end the editing session. MEDIT types the name of the file being written and exits. Instead of using the line editing E command, the user could have issued line editing commands, possibly returning to full screen editing mode using the line editing command A, as described in section 4.8.

5.3. Editing in Full Screen Mode

When MEDIT is invoked with the /WINDOW switch on an existing file, it behaves as follows:

@MEDIT/WINDOW F00.TXT
[V200] Edit: F00.TXT

It then blanks the terminal's screen and fills the screen with lines of the file. We will assume that the terminal being used can fit only 6 lines on the screen at once:

***** F00.TXT *****

*BF*
00100 "Four score and seven years ago, our fathers
00200 brought forth on this continent a new nation
00300 conceived in liberty and dedicated to the
00400 proposition that all men are created equal"
00500 Thus begins Abraham Lincoln's famous Gettysburg

At this point you may typealter mode commands to change text that is on the screen. You may also use commands to move around in the file, displaying text that is before or after the current text on the screen. For example, you may type "WN" to display the next "window" in the file. MEDIT will modify the screen to:

***** F00.TXT *****

*BF*
00300 conceived in liberty and dedicated to the
00400 proposition that all men are created equal"
00500 Thus begins Abraham Lincoln's famous Gettysburg
00600 Address, a speech that was little noted at the
00700 time he made it but which has since become one

Notice that there is some overlap between windows. In the case of this hypothetical small screen of 6 lines, the overlap seems excessive. But on more typical screens of 20 or more lines, the overlap is only a small proportion of the total screen size.
Full Screen Editing Tutorial

In full screen editing mode, you may type any of the available alter mode commands. This chapter will describe only the most basic commands. The following simple commands may be used to change the position of the cursor on the screen and to move around in the file. Note that the current character is the character above the cursor.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;space&gt;</td>
<td>Move forward a character</td>
</tr>
<tr>
<td>W&lt;space&gt;</td>
<td>Move forward a word</td>
</tr>
<tr>
<td>-&lt;space&gt;</td>
<td>Move backward a character (&lt;backspace&gt; also)</td>
</tr>
<tr>
<td>-W&lt;space&gt;</td>
<td>Move backward a word</td>
</tr>
<tr>
<td>N</td>
<td>Move to the next line</td>
</tr>
<tr>
<td>U</td>
<td>Move up to the previous line</td>
</tr>
<tr>
<td>WN</td>
<td>Move to the next window (screenful)</td>
</tr>
<tr>
<td>WU</td>
<td>Move up to the previous window</td>
</tr>
<tr>
<td>Sc</td>
<td>Skip over to character “c”</td>
</tr>
</tbody>
</table>

The following simple commands may be used to delete and insert relative to the position of the cursor:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>Delete the current character</td>
</tr>
<tr>
<td>WD</td>
<td>Delete the current word</td>
</tr>
<tr>
<td>I</td>
<td>Insert before the current character</td>
</tr>
<tr>
<td>†L</td>
<td>Insert a line below the current line</td>
</tr>
<tr>
<td>†D</td>
<td>Delete the current line</td>
</tr>
</tbody>
</table>

The following important commands are also available:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>Ends alter mode. You will be returned to line editing mode.</td>
</tr>
<tr>
<td>&lt;ESC&gt;</td>
<td>Cancels any partially typed command. Also ends insertion mode started with the I command above. &lt;ESC&gt; may be used at any time.</td>
</tr>
<tr>
<td>†U</td>
<td>Restores the line to its form before it was altered.</td>
</tr>
<tr>
<td>?c</td>
<td>Tells you what the command character “c” does. On the Visual 200 terminal, you must type two “?”s.</td>
</tr>
</tbody>
</table>

For a complete description of all the commands available in display editing mode, see chapter 14.

Some supported terminals have special function keys which perform some of the above operations. You may wish to label these keys on your terminal so that you can use them instead of remembering some of the command characters. See appendix VI for a description of the special function keys for the terminals you are likely to use. Appendix VI also tells you if MEDIT requires your terminal to be set up in any special way to be used properly in full screen mode.

5.4. Useful Switches for Full Screen Editing

You may set up MEDIT so that it assumes the /WINDOW switch by default. To do so, create a file in your directory called MEDIT.INI, and put in it a line that reads:

/WINDOW

It is better to tell MEDIT to assume the /WINDOW switch only if you are using a supported video terminal, which you can do by using the line
Full Screen Editing Tutorial

ODOY, /WINDOW

instead

Some other useful switches for full screen editing are:

/NONUMBER
Tells MEDIT not to display line numbers on the screen. You may want to set this switch only when you use a supported video terminal, since line numbers are very useful during line editing, so use the same conditional command as for /WINDOW above.

/STRIP
Prevents MEDIT from writing line numbers in its output file. This switch can be put in MEDIT.INI or used in another file, MEDIT.BUF, to take effect only for certain file types. For example, OTYPE="DOC", /STRIP will set /STRIP only for files whose type (extension) is "DOC".

/AUTOOCR
Causes MEDIT to automatically insert a <CR> on long lines that you type in alter insert mode. This switch is very handy for typing English text without worrying about margins, but may not be so desirable for FORTRAN programs. You may have MEDIT set it only for specific file types as described for the /STRIP switch above.

Chapter 17 contains a complete list of the switches that modify MEDIT's behavior, and section 16.4 describes the methods for conditionally setting them. Note that these switches may be typed as line editing commands to change their setting during an editing session.
CHAPTER 6
TEXT JUSTIFICATION

6.1. J - Join

*Jrange
The lines in "range" are concatenated to form a single line. Excess blanks and tabs are removed from the lines before joining them. A blank is inserted between lines if necessary. If "range" is a single line then that line and the one following are joined. If "range" is more than one line then the lines in the range are joined.

6.2. JL - Justify Left

*JLrange
Leading spaces are removed from the lines in range. LMARGIN-1 spaces are inserted. LMARGIN is 1 by default, which means that no spaces are inserted. See chapter 17 to find out how to change LMARGIN.

6.3. JR - Justify Right

*JRrange
Enough leading spaces are inserted at the beginning of each line in range so that the right end of the line is at RMARGIN.

6.4. JC - Justify Center

*JCrange
Leading spaces are removed from the lines in range. Enough spaces are then inserted so that the text of the line is centered between LMARGIN and RMARGIN.

6.5. JW - Justify Without padding

*JWrange
The JW command justifies the text in range by trying to place as many words as will fit on each line. A word is any sequence of characters other than spaces. The following parameters affect the way the justification is done. These parameters are described more fully in chapter 17.

LMARGIN Column position of the left margin (default = 1)
Text Justification

**RMARGIN**
Column position of the right margin (default = 70)

**PMARGIN**
Column position of the left margin for lines that begin a new paragraph (default = 1)

A line is treated as the first line of a new paragraph if it is the first line on a page, the first line in range, if it begins with a tab, or if it follows a completely blank line. If a line begins with a tab, the tab will be retained in the justified text even if the PMARGIN is 1.

After justification, normal words will be separated by a single space. However, words ending in ",", ",", ",", or "," will be followed by two spaces in the justified text.

By adjusting the parameters, the user can justify text in a normal paragraph style (with and without indentation). It is also possible to justify text as it is justified above in the description of the parameters. For example, the LMARGIN can be set to 17 and the PMARGIN to 6 to have the first line of each paragraph begin in column 6 and all other lines of the paragraph begin in column 17.

6.6. **JU – Justify**

**JUrange**

The JU command performs the same action as the JW command. It then inserts extra spaces throughout each line so that the right margin comes out to RMARGIN. Thus the left and right margins become even, as in this paragraph in this manual. The extra spaces are not inserted in the last line of a paragraph, however, since that often results in very sparse text.

---

\[12\] Note that a line which contains only tabs, while appearing like a blank line, is not treated as a blank line by the justification algorithm.
REFERENCE MANUAL

The following conventions are used to describe the syntax of commands.

lower case words are descriptions (meta-symbols) that stand for values that should be supplied by the user

[] encloses optional parts of commands

| indicates alternatives

UPPER CASE indicates letters that should be typed as part of a command

S indicates the <ESC> key, marked as "ESCAPE", "ESC", "ALTMODE", or "ALT" on the terminal

other symbols must appear in the command as indicated

*italics* are used in examples to indicate the parts of the example that are typed by the user, while non-italic parts indicate parts typed by MEDIT.
CHAPTER 7
INVOKING MEDIT

7.1. Creating a file

MEDIT may be invoked to create a new file by one of the following three methods:

1. `@MEDIT[,switches] fileCREATE[,switches]`
2. `@MEDIT`
   `File: fileCREATE[,switches]`
3. `@CREATE[,switches] file`

where "file" is a file name and "switches" are global switch settings as described in chapter 17. An example of creating a file is:

   `@MEDIT/CREATE HELLO.SAI`
   `[LA36] input: HELLO.SAI
    00100`

The switches that may be used are listed in chapter 17. One particularly interesting switch is `/WINDOW`, which causes MEDIT to go immediately into full screen editing mode. To make a switch such as `/WINDOW automatic`, it may be put in a file named MEDIT.INI in the user's directory. See chapter 17 to find out how to set up default switches in MEDIT.INI.

7.2. Editing an existing file

MEDIT may be invoked to edit an existing file by one of the following three methods:

1. `@MEDIT[,switches]infile[,switches] [outfile[,switches]]`
2. `@MEDIT`
   `File: [switches] infile[,switches] [outfile[,switches]]`
3. `@EDIT[,switches] infile[,switches] [outfile[,switches]]`

where "infile" and "outfile" are file names. For example,

\footnote{Some operating systems under which MEDIT runs have a CREATE command that invokes a specified editor.}

\footnote{Notice, however, that the TOPS-20 EXEC does not know about some of the valid MEDIT switches and thus will not allow you to use them on the command line with the CREATE and EDIT commands.}

\footnote{This form works only if the operating system under which MEDIT runs has a definable EDIT command and the user has in fact set it up to invoke MEDIT. For example, under TOPS20, the user defines the logical device EDITOR: to be SYS:EDIT:EXE}
Invoking MEDIT

@MEDIT HELLO.SAI HI.SAI
[LA36] Edit: HELLO.SAI (OUTPUT AS) HI.SAI

If the /WINDOW switch is used, MEDIT will go into full screen editing mode instead of prompting for a line editing command.
CHAPTER 8
LINE/PAGE NUMBERS

8.1. Files With More Than One Page

When a file is large or has natural subsections, it is often advantageous to divide it into more than one page. For example, each chapter of a book or each subroutine of a program may be put on a page by itself. MEDIT treats line numbers without an explicit page number as referring to the current page, so all simple commands will refer to the current page of the file until another page is specified. By placing each subsection of the file on a page by itself, the attention of the editor may be focused on that page. Accidental deletions and changes to other parts of the file can be avoided more easily because most destructive commands act only on the current page unless an explicit page number is given in the command.

Unlike line numbers, page numbers are sequential positive integers (1, 2, ...). The first page of the file is always page 1, the second page, page 2, and so on. If page 3 is deleted, page 4 becomes page 3, page 5 becomes page 4, and so on. There is essentially no limit to the number of logical pages that a file may have.

Each page of the file has its own set of line numbers. Line number 00100 on page 1 and line number 00100 on page 2 are distinct lines of the file. The complete address of a line in the file is specified by giving the line number and page number of that line. For example,

00100/3

specifies line 00100 on page 3 of the file.

When the file is printed on an output device, a given logical page of the file may occupy more than one physical page on the output device. Each logical page will begin on a new physical output page. Thus pages in the file may also be used to control the way the file is printed on a line printer or other output device.

8.2. Syntax of Line Addresses and Ranges

Informally, a line address consists of a line number optionally followed by a page number. Line numbers are just ordinary numbers like "500", and page numbers are ordinary numbers that begin with a "/", e.g. "/5" is page 5. "500/5" refers to line number 500 on page 5. If the page number is not given, the line number refers to the current page of the file. A range of lines is specified by two addresses separated by a "..", for example "500/5:1000/10" refers to line 500 on page 5 through line 1000 on page 10, inclusive. If a page number is given for the first address in the range, that page becomes the default page for the second address. Thus "500/5:700" is the same as "500/5:700/5", even if the current page is not page 5.
Line addresses and ranges may be specified explicitly or by using some special symbols that stand for the first line of the page, the last line of the page, and so on. Arithmetic constructs may also be used to specify relative line positions and the number of lines to be included in a range of lines. The grammar for specifying line addresses and ranges is given below more formally using Backus-Naur form:

```
n ::= a positive integer
line-number ::= n            explicit line number
                .= .            current line
                .= ^            first line of page
                .= *            last line of page
                ::= line-number - n  nth line before line-number
                ::= line-number + n  nth line after line-number

page-number ::= /n            explicit page number
                .= /            current page
                .= /^           first page of file
                .= /#           last page of file
                ::= /page-number - n  nth page before page-number
                ::= /page-number + n  nth page after page-number

address ::= line-number
             ::= line-number page-number
             ::= BF            first line of file

range ::= address            single line
         ::= address:address    multiple lines, includes
             endpoints and lines in between
         ::= address^n          includes n lines
             starting at address
         ::= page-number         all lines on that page
         ::= EF                 entire file
         ::= address:EF           address to end of file
         ::= IR                 range passed to an
             indirect command file
```

The special symbols that appear in the syntax above may be used in place of line or page numbers and have the following meaning:

- `/` (period) may be used to denote either the current line or the current page. For example, "100./" means line number 100 on the current page and "/2" means the current line number on page 2. The current line number and page are determined by the previous command. Generally the last line which the previous command printed or changed will be the new current line.

- `*` may be used to denote the last line on the page, or the last

---

16 The maximum line number is implementation dependent. On PDP-10 systems, the maximum line number is 99999. On some machines, MEDIT may not be able to provide line numbers larger than $2^{15}$, or approximately 32000.
page. For example, "#" means the last line on the current page, and "#/#" means the last line on the last page of the file.

may be used to denote the first line of the page or the first page of the file.

BF

BF stands for the first line on the first page, i.e. #/#.

EF

When used as the second address in a range, EF stands for the last line on the last page of the file.

When used by itself, EF stands for the entire file, i.e. #/#/#.

IR

When used in an indirect command file, IR stands for the range passed to the indirect command file at the time the "@" command was used to invoke the command file.

8.3. Line and Page Number Arithmetic

Addition and subtraction may be used in line and page specifications. For page numbers the arithmetic is straightforward. For line numbers, the arithmetic refers to actual lines of the file rather than line numbers themselves. The following examples of arithmetic should make its use clear:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>100+2</td>
<td>The second line after line 100</td>
</tr>
<tr>
<td>10013</td>
<td>3 lines starting at 100 (i.e. 100:100+2)</td>
</tr>
<tr>
<td>/5-3</td>
<td>Same as /2</td>
</tr>
<tr>
<td>/+2</td>
<td>Current page number plus 2</td>
</tr>
<tr>
<td>/-1</td>
<td>The previous page</td>
</tr>
<tr>
<td>-.1.+1</td>
<td>The previous line through the next line</td>
</tr>
<tr>
<td>%+2</td>
<td>The last line on the current page and the first line on the next page (if any).</td>
</tr>
</tbody>
</table>
CHAPTER 9
INSERTING, DELETING, AND REPLACING LINES

9.1. D - Delete command

The D (delete) command will delete a specified range of lines.

*Drange[,Y]

If more than 10 lines are being deleted and ",Y" was not specified, MEDIT will ask

Are you sure you want to delete nn lines?

to which the user may reply "y" to go ahead and delete the lines, or "n" to forget about deleting the lines.

Page marks that fall within range will be deleted. Notice that page marks may be deleted without deleting lines of text by the K (kill page mark) command described in section 12.3.

9.2. I - Insert command

The I (insert) command will allow new lines of text to be input from the user's terminal into the file:

*I[address[,increment]]

The new line(s) will have numbers starting at "address" if no line has that number already, or starting at a line number between "address" and "address+increment" if there is already a line number at "address". If no arguments are given, insertion will continue at the place the previous I command left off. This default is useful for correcting mistakes on previously inserted lines and then getting back into insert mode at the correct place. Example commands will illustrate some of the ways the insert command can be used:

*I/100 will insert lines starting at or after line 100 on the current page

*I/100/2,50 will insert lines starting at or after line 100 on page 2 and use an increment of 50

*I/* will insert lines starting at *+increment, where "*" is the last line number on the current page and "increment" is the current default increment (since no increment was specified)

*I/2 will insert a page mark at the end of page 2 and start inserting text on newly created page 3 (page 2 must exist when the command is given)

I/ continues insertion where it last left off on the previous insert command
Inserting, Deleting, and Replacing Lines

The I command will always allow at least one line to be inserted. If the inserted line falls between two consecutively numbered lines then the message "% Line numbers out of Order" will be printed after the line is inserted. The N (reNumber) command may be used to renumber the lines. After a line has been inserted, MEDIT will add its line number to the current increment to obtain the number of the next line to be inserted. If a line already exists with this computed number, or if it is necessary to pass over a line which already exists, or if the computed number is greater than the maximum allowed line number, then insertion mode will terminate and the user will be returned to command level.

Insertion of lines may be terminated by the user at any time by use of the <ESC> key. If <ESC> is used at the end of an inserted line with text in it then that line is inserted into the file and insertion mode is terminated. If <ESC> is typed as the first character of a line during insertion then insertion mode is terminated without inserting the (empty) line into the file.

Usually when the user requests insertion between two existing lines of the file with no explicit increment, only one line will be inserted since the default increment is 100 and consecutive lines of the file usually have numbers that differ by 100 or less. If the user wishes to insert more than one line between two existing lines, a small increment may be specified (e.g. 5 or 10). The /INCREMENT switch may be used to change the default increment for the current editing session, or permanently if it is put in MEDIT.INI.

9.3. R - Replace command

The R (replace) command is equivalent to a D (delete) command followed by an I (Insert) command:

«Rrange[,Y][,increment][,Y]

The deletion is performed on the specified range, and insertion begins with the first line indicated by that range. If more than 10 lines are being replaced and .Y was not given, the user will be asked to confirm the command as with the D command (see section 9.1). Some examples of the use of the R command are:

«R100:500 Replace lines 100 through 500, numbering the new lines starting with 100 and using the default increment to choose the next line number (until <ESC> is typed or the next existing line number is encountered).

«R/2.50 Replace all lines on page 2 and insert new lines, using an increment of 50.

«R100/2 Replace two lines beginning at line 100, i.e. line 100 and the line after it.

As with the I command, insertion may be terminated by using the <ESC> key. The R command handles the line on which <ESC> is typed the same way as the I command (see section 9.2).
CHAPTER 10
PRINTING THE FILE

10.1. P - Print command

The P (print) command prints lines of the file on the user's terminal. Unless otherwise specified through default switches or print-options, line numbers will be printed with each line.

P[range][/print-options]

When page marks are encountered within the range of lines to be printed, "Page n" will be printed to indicate the presence of a page mark at that point.

P100/1:100/2
00100 This is a line
00200 This is another line
...
...
Page 2
00100 This is a line on page 2
*

Any of the following print-options may be used with the P command. When multiple options are used they are separated by commas.

Print Option Explanation
N Print line numbers. This option is useful when the /NONUMBERS switch has been set.
S Suppress line numbers.
E Eject the paper at the bottom of a page by sending a form feed to the terminal.

The "bottom of the page" is determined by a page mark or by exceeding the length of the terminal page. The default length depends on the terminal being used. It can be changed with the /LENGTH parameter (see chapter 17).

Many operating systems provide a method for suppressing program printout without aborting the program. For example, on DEC systems the output can be suppressed by typing ^O.

Common abbreviations used with the P command include:

P Print P Lines, beginning with the current position. P Lines is 16 by default, and is set using the /PLINES switch (see chapter...
Printing the File

17 for more detail. When successive P commands without arguments are given, the current line of the file will not be retyped on the second and subsequent P <CR> command.

** or <LF>
Print the next line of the file.

*- or <ESC>
Print the previous line of the file.

*PL
Print all the lines of the current page.

*PEF
Print the entire file (4/4:*/*).

10.2. L - List command

The L (list) command prints part or all of the file on the line printer:

*Lrange[,]S[,N]

The ",S" and ",N" options for suppressing or including line numbers are the same as for the P (print) command. Line printer output will have suitable page headings on the top of each page. The device to which the output goes may be set with the /LPTDEVICE switch.

10.3. V - Case Inversion commands

The V (case InVerse) commands change the case of all letters in a specified range of lines. The format for these commands is:

*VV[range]

*VL[range]

*VU[range]

The command "VV" inverts the case of all letters, "VU" converts all letters to upper case, and "VL" converts all letters to lower case.
CHAPTER 11
WRITING OUT THE FILE AND EXITING

11.1. E - End Edit command

The E (end-edit) command ends work on the current edit buffer. Unless otherwise specified through options to the command, the E command writes the current edit buffer to disk before deleting it, thus preserving the work done on the buffer. If the current buffer is the only buffer, MEDIT will then exit to the operating system; otherwise, MEDIT will switch to another buffer.

```
*E[D][I][N][Q][S][file]
```

If "file" is not specified, the output file will have the name specified at the time the buffer was created, which is usually the same name as the input file; if "file" is specified then it is used as the name of the output file instead.

The options for the E command are:

- **D**
  - Delete the original input file and do not create an output file.
  - MEDIT will ask for confirmation before actually performing the deletion.

- **I**
  - Insert a comment line at the beginning of the file consisting of today's date and a short comment typed by the user. The string used to prefix and suffix the comment line is set using the switch /COMMENT.

- **K**
  - Kill page marks (form feeds) and line numbers before writing out file.

- **N**
  - Output line numbers and page marks. This option is useful if /STRIP has been set.

- **Q**
  - Quit the edit, i.e. do not create a new output file.

- **S**
  - Strip the line numbers before writing out the file. This option may be made the default by setting /STRIP.

To delete the current buffer without saving the edited version of the file on disk, use the **Q** option:

```
*E Q
```

The version of the file on disk will correspond to the state of the buffer when it was most recently saved as the result of a previous E, G, or W command, and will not reflect the editing done on the buffer since.
Writing Out the File and Exiting

11.2. W - Write command

The W (write) command writes the current buffer being edited out to disk (like the E command) but does not delete the buffer. Further editing may then be done to the buffer.

*W[i][k][n][s][file]

The options for the W command mean the same as the corresponding options of the E command above. The W command saves the state of the current buffer, presumably at a point in the editing when the user is satisfied with it. If a disastrous mistake is made while further editing is in progress, the user can return to the saved version of the file by ending the edit with the EQ command and starting the edit again from the file on disk.

MEDIT automatically checkpoints the temporary file it uses during editing to prevent lossage of the edit in case of a system or editor crash. This checkpointing is invisible to the user. However, the frequency with which MEDIT checkpoints can be set using the global switch /CHECKPOINT. To recover editing after a system crash, start MEDIT in the same directory that was used while editing before the crash. MEDIT will detect that its temporary files exist and will ask if the user wants to recover the editing.

11.3. G - Go command

The G (go) command ends the edit like the E command, and then runs another program. Because the G command requires that MEDIT exit to the operating system, it can only be used if the current buffer is the only buffer.

*G[d][i][k][n][q][s][file]

The options have the same meaning as in the E command above. The default action taken by the G command on DEC systems is to repeat the previous COMPILE, LOAD, DEBUG, or EXECUTE command. This action is convenient for debugging compiled programs. For example

```
COMPILE PROG.SAI
SAIL: PROG.SAI 1 2
<an error is detected and MEDIT is invoked>
Edit: PROG.SAI
*
<some editing here>
*G
[PROC.SAI]
SAIL: PROG.SAI 1 2
<perhaps more debugging here>
```

The G command may also be used to run a program rather than perform the previous COMPILE, LOAD, DEBUG, or EXECUTE command.

---

17 Note that, unlike in SOS, in MEDIT it is not necessary to use the W command to save the file to prevent lost editing in case the system crashes.

18 On some systems it may be necessary to explicitly use the /RECOVER switch.
Writing Out the File and Exiting

To set up the G command to run a program, use the global switch /RUN described in chapter 17, e.g.

\[*RUN:SYS:LISP.EXE\]
\[*G\]
\[[F00.LSP]\]
\[LISP 1.6 ...\]

Notice that the /RUN switch may be placed in the user's MEDIT.INI or MEDIT.BUF file. The line

\[ct="LSP",/RUN:SYS:LISP.EXE\]

in MEDIT.BUF would set up the G command to run LISP if the file being edited has type "LSP". Thus the action of the G command may be customized to the type of file being edited. The use of MEDIT.INI and MEDIT.BUF is described in chapter 17.
CHAPTER 12
MANIPULATING LINE AND PAGE NUMBERS

12.1. * Set the Current Position

The "." (set current position) command sets the current line address (line/page) within the file, or "pushes" the current position onto a stack of remembered addresses. The syntax of this command is

*address
*.PUSH
*.POP

The first form makes address the current position in the file. If only a line number is given, the current page is unchanged. To position to the beginning and end of the file, "BE" and "EF" may be used.

The "PUSH" and "POP" commands use a stack to remember the current position in the file and return to it later. Each time "PUSH" is used, the current address is stored away, and each time "POP" is used, the most recently stored address is recalled and the current position is set to it. For example:

*p.
00100 This is the current line.
*.push
*.end$
69500 This is the end.
*.pop
*p.
00100 This is the current line.

The "=" command types out the current address and the contents of the stack of remembered positions.

12.2. M - Mark Page command

The M (mark page) command inserts a page mark into the file immediately before the line specified.

*Maddress

Thus, the line given as an argument becomes the first line on a new page.
Manipulating Line and Page Numbers

12.3. K - Kill Page Mark command

The K (kill page mark) command removes a page mark.

*K/page-number[:/page-number]

For example

*K/2

kills the page mark for page 2, which will combine pages 1 and 2 into a single page. If the line numbers on the single page resulting from a page mark deletion are not in order, the message "% Line numbers Out of Order" will be printed. The page should be renumbered with the N command. Use of the "NP" command before using the "K" command will prevent the "% Line numbers Out of Order" message. A page mark may be conceptualized as appearing at the beginning of the page, immediately before the first line of the page.

12.4. N - reNumber command

Some or all of the lines in the file may be renumbered using the N (reNumber) command.

*N[increment][,range[,starting-number]]

increment specifies the increment to use between successive line numbers. If it is not given, the current value of the /INCREMENT switch will be used, which is 100 by default. range is the range of lines to be renumbered (the default is the entire file). starting-number specifies the line number to be assigned to the first line in the range.

12.5. NP - reNumber over Pages command

The NP (reNumber over Pages) command renumbers lines like N but does not reset the line number to "increment" when a page mark is encountered. Thus the first line number on each new page will have the value of the last line number on the previous page plus the increment. This command is useful before killing page marks using the K command. It has the same syntax as the N command described above.

*NP[increment][,range[,starting-number]]
13.1. **F - Find command**

The **F** (find) command locates one or more pattern strings within a range of lines:

```
*F[[patterns]${range}$[find-options]]
```

**MEDIT** will find the first occurrence of the pattern in the range and print the line containing that pattern. If the pattern is not found in the specified range, **MEDIT** will respond with:

```
$Search Fail $ (first : last)
```

where "first" and "last" are the first and last lines which **MEDIT** searched. In this case, the current line is unchanged from what it was before the **F** command was issued.

Once the pattern has been given once, it may be omitted and the **F** command will search for it again. If all arguments of **F** are omitted then the **F** command will continue the previous search as if it had not found the line it did, that is, it will find the next occurrence. If the pattern is given but the range is omitted, **MEDIT** will search starting on the line after the current line, until the end of file. Some examples are:

* **F**ABCDES12  
  Find the first line on page 2 containing "ABCDE".

* **F**$SI3  
  Find the first line on page 3 containing the previously given pattern.

* **F**ALWAYS$  
  Find the first occurrence of "ALWAYS" beginning with the line after the current line through the end of the file.

When no pattern or range is given to **Find**, the result depends on whether the <ESC> key is used:

* **F**  
  Continue to search for the pattern previously specified in the range previously specified, and with the previous options.

* **FS**  
  Continue to search for the previously specified pattern from the line after the current line to the end of file, and reset the options to their defaults.

* **FS,find-options**  
  Same as above, but set the given find-options to those given.

The **Find** command can find up to 10 patterns at once. If any of the patterns are found on a line, that line will be printed. More than one pattern is specified by separating them by <CR>.

```
*Fstring1<CR>
   string2<CR>
   ...
   string10<CR>
```

For example
will find a line containing either "BEGIN" or "END". Note that MEDIT prompts for additional patterns with "$F<\times>" after <CR> is typed. To abort a partially typed in F command, type a $G. Note that to include a $G in the string to find, the flag representation for $G must be used.

13.2. Find Options

Any combination of the following find-options may be given. The options are preceded by and separated by a comma.

- n
  an integer count that specifies the number of lines to be found before the search is terminated. The default value is 1. This option may be used instead of repeating "$F<CR>" n times.
- E
  Find all lines with the given pattern(s) in the range.
- Q
  Match the specified character string exactly, distinguishing upper and lower case.
- D
  Don't print the line found (quiet mode).
- A
  As each line is found, allow the user to specify options that will change the line, or allow the user to "decide" if he wants to perform the changes he specified initially in the Find command (see below).
- -
  Match only those lines which do not contain one of the given strings.

Only one of the following action options may be specified as part of the find-options:

- A
  Alter the line found.
- I
  Insert a line after the line found.
- K
  Kill (delete) the line found.
- R
  Replace the line found.

13.3. Responses in Find Decide Mode

When the decide option "$D$" is used with the F command, MEDIT will print each line found and prompt the user for a response with "$D$". The user may type one of the following action characters:

- Y
  to accept this line, perform the action specified in the F command, if any, and continue until the count is exhausted.
- N
  to prevent the chosen action from being performed on this line and continue searching until the count is exhausted.
- Q
  to quit searching without performing the specified action, if any, on the line just found.
- G
  to go perform the option on this line and all others that are found without asking again. The "$G$" response turns off decide mode for the remainder of current search.
- ? or H
  to list these responses.
Text Search and Substitution

If the original F command included no action option then the user may type one of the
action characters (A, I, K, or R) instead of one of the above characters.

13.4. S - Substitute command

The S (substitute) command substitutes a new string for all occurrences of a pattern
in the specified range and prints the lines in which a substitution has been made.

*S[patterns$newstrings]$[range],substitute-options

To substitute the same strings specified in the previous S command, omit the strings and
one $ from the command. For example:

*SOLD$NEW$100/1:* Substitute “NEW” for all occurrences of “OLD” in the specified
range.
*S$/3

Substitute “NEW” for “OLD” (as specified previously) on page 3.

If the range is omitted, MEDIT will assume that the substitution is to be performed on the
current line only.

It is possible to perform several substitutions at one time. A set of patterns to find
is specified, separated by <CR>’s as for the F command, and terminated by <ESC>. The
set of replacement strings is specified similarly.

*Smatch1<CR>

...<CR>
match$replace1<CR>
...<CR>
replace$[range],[substitute-options]

For example

*SX
SY
SY
*S$X$E

will change all X’s to Y’s and Y’s to X’s in the file. Up to 10 match strings and 10
replacement strings may be given. $G may be used while entering the strings to abort the
S command.

13.5. Substitute Options

Any combination of the following substitute-options may be given. The options are
preceded by and separated by a comma.

n
an integer count that specifies the number of lines on which the
substitution is to be performed. n defaults to a very large
number.

* Substitute all occurrences of the given pattern in the range
specified.
Text Search and Substitution

D

By specifying this option, the user can look at each line after the substitutions have been made and "decide" to accept or reject the substitution.

Q

Do not print the lines on which the substitution has been made (quiet mode).

E

Match the specified match strings exactly, differentiating upper and lower case.

A

Go into Alter mode on lines after the substitution has been made.

13.6. Responses in Substitute Decide Mode

When the decide option is used with the S command, MEDIT will print each line found with the substitutions made and then prompt the user for a response with "D=". The following responses are available:

Y

to accept the changes to this line and go on to the next

N

to reject the changes to this line and go on to the next

Q

to reject the changes to this line and quit the S command

G

to accept the changes to this line and go perform the substitution on all subsequent lines found, without asking again, until the count is exhausted. The "G" response turns off decide mode for the remainder of the S command.

A

to accept the changes to this line and go into alter mode on the line found. When alter mode is exited, the S command will be resumed.

? or H

to list these responses.

13.7. Special Matching Characters

In addition to searching for literal string, the S and F commands can search for patterns. Patterns are specified using the following control characters in the string along with normal printing characters. Some of these characters may be typed directly, but some of them have special functions (e.g. \C, \O, \Q, \S) so they must be entered using /Flag notation (see page 17.5), e.g. "\C" for \C.

13.7.1. Special Matching Characters for the Search String

\A

stands for any alphabetic character (A-Z,a-z)

\B

stands for the beginning of a line

\C

stands for any character

\D

stands for any decimal digit (0-9)

\E

stands for the end of a line

\L

stands for any lower case letter (a-z)

\NC

stands for "not C", where C is any character or pattern character.

\OC

stands for "one or more of" the character or pattern C

\P

stands for any alphanumemric character (A-Z,a-z,0-9)

\QC

quotes the character C. \Q is particularly useful when searching
Text Search and Substitution

for a pattern control character, since if the control character
were not quoted it would be interpreted as the pattern rather
than as itself.

\^T stands for a space or a tab
\^U stands for any upper case letter (A-Z)
\^W stands for any word delimiter. A word delimiter is any character
other than a legal identifier character. The identifier characters
may be set using the global switch /IDENTIFIERS.
\^Zc stands for "zero or more of" the character or pattern c

For example the pattern \^W\^A\^Z\^P\^W matches any word delimiter followed by an
alphabetic character followed by zero or more alphanumerics characters followed by a word
delimiter, i.e. an Algol identifier. The pattern \^B\^E matches beginning of line followed by
end of line, i.e. a blank line.

13.7.2. Special Characters for the Replacement String

When patterns are found it is often useful to insert them into the replacement string
of the $S$ (substitute) command. The following special characters may be used in the
replacement string, along with normal characters. They cause the appropriate pattern from
the search string to be inserted at that point.

\^S{n}\^S in the replacement string stands for the nth pattern matched in
the corresponding search string
\^_ in the replacement string stands for the next pattern matched in
the corresponding search string
\^T in the replacement string stands for the UPPER CASE of the
next pattern matched in the corresponding search string
\^V in the replacement string stands for the inverse case of the next
pattern matched in the corresponding search string

13.7.3. Some Useful Example Patterns

The $S$ command

*S\^W/S\^L+S

will match any word delimiter followed by a lower case letter and replace it with the same
word delimiter (first pattern) followed by the upper case of the lower case character
(second pattern), i.e. it will capitalize the first letter of all words that are not already
capitalized. The first use of \^L or \^T stands for the first pattern in the search string,
and the second use stands for the second pattern, etc.

Sometimes it is necessary to change the order of two patterns in the string. In this
case, the \^S construct may be used to specify exactly which pattern in the search string
is to be substituted into the replacement string. For example,

*S\^F\^N/I\^O\^C/\^O\^C/S\^F\^N/I\^S\^2\^S;\^S\^1\^S/S

will find "FN" followed by one or more of anything followed by ",", followed by one or
more of anything followed by ",", and reverse the patterns before and after the ",". Thus
"FN(ARG1,ARG2)" would become "FN(ARG2,ARG1)".

The pattern \^B\^E matches an empty line. Thus the command


Text Search and Substitution

\*F\*E$range,+/K

will find all of the blank lines in the range and delete them. The command

\*S+BS  $range

will insert a tab at the beginning of each line in the range and

\*S+BS  \$S$range

will delete one tab from the beginning of each line in the range. To delete all of the tabs or spaces at the beginning of a line, the "one or more of" and "tab or space" patterns may be combined.

\*S+BS'O7+NT$S'SS'$S$range

Notice that it is necessary to replace the non-tab character using the "$S" construct in the replacement string. The following commands will change all occurrences of the identifier "I" to "J" in a program without changing all the I's embedded in other identifiers to Js:

\*S'W/I'W$+J+/_SEF
\*SSEF

The same substitution needs to be done twice to catch all of the I's because of expressions like "I+I". The first time beginning-of-line followed by "I+I" will be matched and "J+I" substituted. At that point, the remainder of the line will be just "I" which will not match the pattern because there is no delimiter before the "I". However, on the second substitution the line will read "J+I" so "+I" will be matched and changed to "+J".

The MEDIT pattern-matching algorithm will always match the shortest leftmost string that corresponds to the given pattern. The "zero or more of" operator (\*Z) likes to match zero characters if it can since that will result in the shortest possible string. The beginning and end of line patterns (\*B and \*E) cannot possibly match anything when given in the middle of a pattern.

13.8. Hints on Using F and S

The decide option for the \*F and S commands provides for interactive examination and editing of the file on a line-by-line basis by the user. Lines that match the given search string or pattern are presented one by one to the user for examination. By typing one of several characters the user determines what should be done to the line that was found. The "decide mode" is entered when when the ",D" option is given to an F or S command.

When very complex strings are given to the F or S command, especially when those strings include one or more of the special patterns described above, it is a good idea to specify the ",D" option in the command so that its effects can be checked before any damage is done by an incorrect command. If the command processes the first few lines correctly and the user is satisfied that it will work on the remaining lines of the file, he can respond with a ",G" to the decide mode prompt and have the rest of the Finds or Substitutions done automatically.
CHAPTER 14
ALTER MODE AND FULL SCREEN EDITING

This chapter describes MEDIT's provision for character-by-character editing of a line of text, and also full screen video editing. Commands typed in this mode are not echoed on the terminal but their effect on the line is seen. Alter mode editing may be done in two ways, in single line mode or full screen 'window' mode. Single line mode may be used from any full duplex terminal, including hard copy terminals, to make changes within a single line at a time. Full screen mode may be used only on supported video terminals. In full screen mode, MEDIT maintains the screen of the terminal so that it appears like a 'window' into the file being edited. The window may be moved around in the file as needed.

Alter mode may be invoked by the A (alter) or X (xTend) command, by the "A" option to the F and S commands, or by the "A" response when "decide mode" is used with the F or S commands. If the /WINDOW switch is given on the command line that invokes MEDIT or in one of the initialization command files, MEDIT will go into full screen alter mode at the beginning of the editing session.

14.1. A - Alter command

The A (alter) command initiates alter mode. It accepts an address of the line to be altered initially:

*A[address[,W]]

If the .W option is specified or no arguments are given, full screen window mode will be in effect; otherwise, single line mode will be in effect.

14.1.1. Single Line Mode

If single line mode is in effect, the line number of the line is printed, and if a video terminal is being used, the line is displayed and the cursor returned to the beginning of the line. On a hard copy terminal, the print head is positioned ready to print the first character in the line. In both cases, the alter mode 'pointer' is now positioned immediately before the first character of the line. The alter mode commands described in section 14.3 may be used to make changes within the line or move to other lines. Note that the "W+R" command may be used in single line mode to change to full screen window mode.

14.1.2. Window Mode

If the .W (window) option is used on a video terminal, a screenful of lines (called a "window") will be displayed. The line to be altered will be placed in the middle of the screen. If no arguments were given to the A command, the current line will be placed in the middle of the screen. The alter mode commands described in section 14.3 may be used to make changes within the line or move to other lines.
14.2. X - eXtend command

The X (eXtend) command is another way to enter alter mode. It functions similarly to the A command.

```
*X[address[,W]]
```

With this command, however, the "pointer" is moved immediately to the extreme right of the line, the characters passed over are displayed on the terminal, and alter insert mode is entered. What the user types is then appended to the end of the line. In single line mode, the user may finish extending the line by typing <CR>. In window mode, <ESC>E must be used to finish extending. In either case, after typing <ESC> the user can perform any alter mode command on the line. The ".W" option may be used as in the "A" command above to select window mode.

14.3. Alter Mode Commands

The operations that may be performed in alter mode are described in the following tables. Most alter commands have the following syntax:

```
[<modifier>]<command-letter>[<arguments>]
```

where the modifier is optional and some commands take arguments.

14.3.1. Modifiers

The following modifiers may be typed prior to a command and will apply to the command which follows.

- **n**
  
  A repeat count for the command that follows, i.e. a number consisting of one or more decimal digits (0–9).

- **W**
  
  Specifies that the command that follows should be applied to the next "word" of the line or the next "window" of the file. For example, N means "next line" and WN means "next window". D means "delete next character" and WD means "delete next word". A word is defined as a sequence of identifier characters, where identifier characters are A–Z and 0–9 plus any additional characters specified using the /IDENTIFIER switch described in chapter 17.

- **-**
  
  Specifies that the command that follows should be processed in reverse. For commands that process within a line, the command will process toward the left instead of to the right, e.g. `- <space>` spaces to the left instead of to the right. For the line delete and copy commands, †D and †C, the "-" means to "undelete" (restore) the line most recently deleted or copied.

The modifiers may be combined, e.g. "-WD" means "delete the previous word".
14.3.2. Cursor Positioning Operations

The cursor may be moved to various locations in the current window, and the window may be shifted, using the following commands. Bear in mind that the commands may be preceded by modifiers to make them process in reverse, on more than one character, etc. Recall that square brackets indicate optional parts of the command.

\[-\][n]\[W\]<space> Pressing the space-bar is the basic way to move forward or backward in the current line. If no optional modifiers are typed, the cursor is advanced by one character. If the "W" modifier is used, the motion will be by words of text rather than by characters. A repeat count "n" will cause the command to be repeated that many times, and a dash "-" will cause the motion to be backward (toward the left) rather than forward (to the right).

\[-\][n]\[W\]<backspace> As a convenience to the user, the backspace key will cause the cursor to move backward. An optional repeat count may be supplied if desired. In insert mode (see "I" command below) the backspace key causes the character just inserted to be deleted.

\[-\][n]\[W\]N The basic way to move down to the next line. If no optional modifiers are typed, the cursor is advanced to the next line. If the "W" modifier is used, the cursor will move forward a window, i.e. the next screenful will be displayed. A repeat count of "n" will cause the command to be repeated that many times, and a dash "-" will cause the motion to be upward (toward the beginning of the file) rather than downward (toward the end of the file).

<CR> Move down to the beginning of the next line.

<LF> Move down to the next line, remaining in the same column.

\[-\][n]\[W\]U Like the "N" command above, but defaults to moving upward to the previous line or window, rather than downward to the next line or window.

\[\uparrow\] Move up to the previous line, remaining in the same column.

\[-\][W]\[B\] Moves to the beginning of the line or window. If "W" is not typed then the cursor will be moved to the beginning of the current line. If "W" is typed, the cursor will be moved to the beginning of the current window, i.e. to the top line of the file currently displayed on the screen. A dash "-" reverses direction, i.e. to the end of the line or end of the window.

\[-\][W]<tab> Like the "B" command above, but defaults to moving toward the end rather than toward the beginning. Thus typing <tab> by itself moves the cursor to the end of the current line and using the "W" modifier with <tab> moves the cursor to the end of the screen.
L

The "L" command exists for the benefit of hard copy terminals. It types the rest of the current line and then moves the cursor to the beginning of current line. This command may soon be reassigned; however, users that like it will be able to define it as a macro.

[-][n]Sc

Skips to the nth occurrence of character "c". If the repeat count "n" is not given, it defaults to 1. The current character is always passed over. If "-" is used, the search and motion will be toward the beginning rather than end of the line.

[-][n]Fstr.<ESC>

If no repeat count is given, the F command finds a string of characters incrementally. As each character of the string is typed, the cursor advances to the first position in the text which matches the string typed so far. This command will move the cursor forward lines (eventually, backward if "-" is typed) if necessary. While the string is being typed, <rubout> may be used to remove the most recent character from the search string and return to the place in the file that the cursor was at before the incorrect character was typed. Note that once "F" is typed, every character typed will be considered as part of the string except <rubout>, which removes the most recent character typed, and <ESC>, which is the only way to get out of incremental find mode. The string typed may include patterns (see chapter 13).

If a repeat count n is given, the F command repeats the previous search that many times. The cursor is moved to the nth occurrence of the search string previously given.

[-][n]↑Fstr<ESC><CR>

Finds a string specified all at once. This find command is more like the line editing "F" command in that the entire string to be found is typed before the search begins. It is not as informative as the "F" command above if the search fails, but it is much more convenient if the string being searched for occurs in many places, since a repeat count can be given to find the nth occurrence of the string, and the command can be issued without a string to continue the search using the previous string. Eventually, options will be allowed between <ESC> and <CR> as in the line editing Find command. The string typed may include patterns (see chapter 13).

The cursor is moved backward in the file until it is below a right parenthesis, if necessary. Then it is moved backward to the matching left parenthesis, taking into account nesting of parentheses. This command is convenient for editing complex algebraic expressions and nested function calls which use nested levels of parentheses.

The cursor is moved forward in the file until it is below a left parenthesis, if necessary. Then it is moved forward to the matching right parenthesis, taking into account nesting of parentheses.
Alter Mode and Full Screen Editing

.address<CR>
The cursor is moved to the specified address in the file (line/page). This command is convenient when the line and page number of a line to be modified is known, e.g. when a compiler complains about a particular line. It is also useful for positioning to a particular page (via "f/n" for page n), the beginning of the file (via "BF"), or the end of the file (via "EF").

.PUSH<CR>
The current location in the file is pushed (remembered) on a stack.

.POP<CR>
The cursor is moved to the location on top of the .PUSH stack (see above) and that location is removed from the stack.

14.3.3. Insert/Delete Operations

.[-][n][W]D
Deletes text in the current line. "D" by itself deletes the next character, the "W" modifier causes the command to delete words, the repeat count "n" causes the command to be repeated that many times, and the "-" causes the command to process toward the beginning rather than toward the end of the line.

.[-][n][W]<rubout>
Deletes text in the current line, like "D" above, but defaults to processing toward the beginning of the line. As a convenience when inserting text (see the "I" command below), <rubout> may be used to delete the most recently typed character. In insert mode, of course, the modifiers cannot be used because they would be interpreted as ordinary text to be inserted.

[n]†W
As a convenience, †W deletes the previous word or words in the line. †W will also work when inserting text.

[-][n]Kc
Kills (deletes) characters up to the nth occurrence of character "c" (defaults to 1 if not specified). Using "-" causes the K command to process toward the beginning of the line.

[n]str<ESC>
Inserts "str" at current location in the line, i.e. at the current position of the cursor. Note that text inserted at the current cursor position is before the character above the cursor. <ESC> is used to end the insertion. <rubout> and †W may be used to delete the most recently typed character or word, respectively, while in insert mode. If <CR> is typed while in insert mode, the result depends on whether full screen editing or single line alter mode is being performed. In full screen alter, typing <CR> in insert mode actually inserts a <CR><LF> at that point in the line, i.e. the line is broken at that point and the characters from the cursor position to the end are
inserted as a new line below. In single line alter, typing 
<CR> ends the alter of the line, but <LF> may be used to 
insert a <CR><LF>. If given, "n" is used as the number of 
characters to insert before leaving insert mode; otherwise, 
insertion continues until <ESC> is used.

[-][n]Xstr<ESC>

Inserts "str" at the end of the line (or beginning if "-" is 
typed). "n" is used as for "I" above.

[-][n][W]Rstr<ESC>

Replaces part of the current line with "str". This command 
is just like the "D" command followed by the "I" command 
(the arguments apply to "D", not to "I").

[-]Zstr<ESC>

Deletes to end of line (beginning of line if "-" is typed) 
and then performs the "I" command.

[-][n]<L

Inserts n lines (default=1) below the current line (above it 
if "-" is typed) and performs the "I" command.

<++>

Inserts a page mark below the current line.

[-][n]+D

This command does two very different things depending 
on whether "-" is typed. Without "-", it deletes "n" lines 
starting at the current line and working downward, ending 
up on the line after the last line deleted. The text of the 
deleted lines is saved in a special delete-buffer. With the 
"-", it restores the most recent "n" lines that were 
deleted, removing them from the delete-buffer. When 
deleted lines are restored, they are inserted above the 
current line. Thus, the "+D" and "-+D" commands are 
inversed of each other.

[-][n]+C

This command does two very different things depending 
on whether "-" is typed. Without "-", it copies "n" lines 
starting at the current line and working downward, ending 
up on the line after the last line copied. The text of the 
copied lines is saved in a special copy-buffer. With the 
"-", it restores the most recent "n" lines that were copied, 
removing them from the copy-buffer. When copied lines 
are restored, they are inserted above the current line. The 
"+C" command functions very similarly to the "+D" 
command, except that the original lines on which it acts 
are not deleted. Note well that the flag character must be 
used to enter a +C on DEC machines (i.e. 'C'), because a 
bare +C by itself will interrupt the editor!

14.3.4. Change Operations

[n]Cstr<ESC>

Changes characters in the line by letting the user 
type over them. After the "C" command is given, 
the next "n" characters typed will be used to
Alter Mode and Full Screen Editing

replace the n characters being changed (default for
n is 1 character). Then change mode will end. The
"C" command is more convenient than the "R"
command for certain tasks, such as replacing "n"
characters in a figure with exactly "n" characters,
because it ensures that the total length of the line
will remain the same. <ESC> may be typed if
fewer than "n" characters have been changed but
the user decides not to change the rest. In that
case, only those characters actually written over so
far will have been changed.

[-] [n]↑Str1<ESC>str2<ESC><CR>
Substitutes str2 for str1 at the next n places in the
file where it occurs.

[W]T
Transposes the next two characters (eventually
words, if "W" is typed).

[-] [n] [W] V
Inverts the case of the next "n" characters or
words, or previous "n" characters or words if "-" is
typed. This command affects only alphabetic
characters; other characters will be passed over
unchanged.

[-] [n] [W] >
Converts the next "n" characters or words to lower
case.

[-] [n] [W] <
Converts the next "n" characters or words to upper
case.

14.3.5. Justification Operations

↑A
Attaches the next line to the current line.

↑B
Breaks the current line into two lines by inserting a <CR><LF> before
the current character and inserting spaces or tabs according to the
indentation of the line that follows.

[n] [W] J
Justifies the current line, currently by breaking the line between words
if it is too long. Eventually, it will also attach the next line if the
current line is too short. Until that feature is implemented, a short
line may be extended by using ↑A first to attach the next line, and
then J to break the resultant line if necessary. [n] and [W] are not
implemented yet.

14.3.6. Miscellaneous Commands

A
Performs the most recently typed command again.
Alter Mode and Full Screen Editing

^U
Restores the line as it was before the cursor was most recently moved to it, i.e. the recent editing is undone. This command may also be used in insert mode.

E
Ends alter mode. MEDIT returns to line editing mode and prompts with a "^".

Q
Restores the line (like ^U above) and ends alter mode (like "E" above).

<CR>
In window mode, moves the cursor to the beginning of the next line. In single line alter mode, it ends alter mode and returns to line editing mode.

<ESC>
The <ESC> key may be used at any time to cancel a partially typed command, or to ensure that the editor is not in insert or incremental find mode. Note that since <ESC> is also used to delimit the search string for the ^F and ^S commands, two <ESC>'s are necessary to cancel a ^F or ^S command.

[W]^R
Refreshes the current line, or the current window if it is preceded by "W". "W^R" may be used to enter full screen alter mode from single line alter mode.

P
This command is provided for use on hard copy terminals. It prints the rest of the current line and retypes the beginning of the line to the current position.

?q<cmd>
Gives help on a specific alter mode command or types a summary of all commands and macro definitions.

[n]^N
Changes the line number of the current line to n. Eventually it will do a local renumbering of the file if no line number is given, but this feature is not implemented.

14.4. Defining Alter Command Macros

Alter macros are a convenient way to define new alter commands. A macro consists of a macro name and a macro body. The name is the sequence of characters that the user will type to invoke the macro, and the body is the sequence of commands that should be executed when the user invokes the macro.

Most macros are designed to function in alter command mode, that is, when MEDIT is waiting for a command character in alter mode. However, it is also possible to define macros that function in alter insert mode, that is, while characters are being inserted into the file. For example, a user might define a character that is not normally used to expand to some frequently typed complex string. While inserting text, the user could then type the macro character whenever she wanted the string to be inserted. Insert mode macros generally end with an "^" command to return the user to insert mode.
Alter Mode and Full Screen Editing

Alter macros may be defined in alter mode using the "M" command:

\[1\] Mn<CR>b<CR> Defines a macro with name "n" and body "b". Both the body and the name may consist of any number of characters, and existing command names may be used for the name, in which case they will be redefined. When the macro is invoked, macros in the body will not be expanded. Thus it is possible to redefine the alter commands using the macro facility without affecting previously defined macros. <ESC> may be used to get out of the macro definition mode if "M" is typed by accident. Note that macros may also be defined via the /MACDEF switch. If no repeat count is given, the macro will be available in alter command mode. If a repeat count of 1 is used, the macro will be available in alter insert mode.

For example, suppose that a ";" is to be inserted at the beginning of several lines while in alter mode and the user wants to define a macro to perform this operation on a line. The user types "M" in alter mode and MEDIT prompts:

Macro Name:;

The user chooses ";" (followed by <CR>). MEDIT now prompts:

Body for Macro :bd1.n

and the user types the commands to be executed when ";" is typed, followed by <CR>. The commands move to the beginning of the line, insert ";", and move to the next line. The user has now defined a new alter command to perform the desired operation on the line.

Macro names may be more than one character. This feature allows the user to define special keys on the terminal to perform alter commands, even when those keys transmit multiple character sequences. However, a macro name may not begin with another macro name, so if there is already a macro named "%A" you may not define one named "%" or "%ABC". This restriction results from the way commands are processed immediately after reading ";", MEDIT wants to execute the ";" macro, but if there were a "%A" macro, it would have to wait for the next character typed. To avoid this difficulty, MEDIT prevents the user from defining ambiguous macros.

The /MACDEF switch may be used to define alter macros while in line editing mode. This feature allows alter macros to be defined in the user's MEDIT.INI or MEDIT.BUF file whenever MEDIT is started or a new file buffer is created. Thus the user can define personalized macros that behave like regular commands. Also, special keys on particular terminals that transmit unique sequences of characters may be set up to perform video editing commands by using conditional commands depending on the terminal being used.

The ? command in alter mode lists the currently defined macros in addition to the built in commands.

14.5. Setting Up Special Key Macros

If a terminal has special keys that send some unique sequence of characters, it may be possible to set these keys up to perform alter commands, provided that the keys do not transmit a sequence that begins with a legal alter command. Alter macros may be
defined interactively to test them out, or they may be defined via the /MACDEF switch\(^{19}\) in MEDIT.INI according to the terminal being used. If the sequence transmitted by the terminal for its special keys begins with <ESC>, the macros should be set up on the sequence without the <ESC> at the beginning so that <ESC> will still work when typed by itself. Note that special keys are often defined in the terminal-dependent module, thus removing the burden from each user. If the keys on the Infoton 400 keypad were not defined in the 1400 module (which they are), the user could define them as follows automatically by putting the following commands in MEDIT.INI (note that we assume terminal type 6 is an 1400):

\[
\text{od}=6;\text{begin 1400}
\]

\[
\text{MACDEF}:"[A","U"
\]

\[
\text{MACDEF}:"[B","N"
\]

\[
\text{MACDEF}:"[C","C"
\]

\[
\text{MACDEF}:"[L","-W"
\]

\[
\text{MACDEF}:"[M","W"
\]

\[
\text{MACDEF}:"[P","WN"
\]

\[
\text{MACDEF}:"[h","WU"
\]

\[
\text{MACDEF}:"[j","I"
\]

\[
\text{MACDEF}:"[n","D"
\]

\[
\text{MACDEF}:"[r","WD"
\]

\[
\text{MACDEF}:"[q","1"D"
\]

\[
\text{MACDEF}:"[Z","U"
\]

\[
\text{MACDEF}:"[H","F"
\]

;\text{end 1400}
\]

14.6. Terminal Support

MEDIT uses a generic MAINSAIL module named TRMTYP to determine the type of terminal that the user has (automatically, if possible). It then associates the correct terminal handler module with the generic module DPYMOD. For example, at a given computer installation the VT52 terminal may be assigned terminal number 5. TRMTYP will determine that the user has a VT52, set the terminal number to 5, and note that the correct module to invoke for DPYMOD is the module whose actual name is "VT52".

When MEDIT is exported to another site, the TRMTYP module for the site's operating system (TOPS20, TOPS10, etc.) must be modified. Until it is, the terminal type and module name association normally made in TRMTYP will be incorrect for that site. In this case, a user may specify the associations in his or her MEDIT.INI file using the line editing command "OD=n" as follows:

\[
\text{OD}=4;\text{begin Data Media}
\]

\[
\text{TERMINAL:DM2500}
\]

\[
\text{LENGTH}:25\]

\[
\text{WIDTH}:80\]

\[
\text{DPY}
\]

;\text{end Data Media}
\]

\(^{19}\)See chapter 17 for a description of the /MACDEF switch
Alter Mode and Full Screen Editing

where "DM2500" is a terminal module that has been defined and placed in the MEDIT module library or which exists in the user's directory. If the operating system reports that the user has terminal type 4 then the DM2500 module is brought in to handle alter mode editing.

Setting the /DPY switch tells MEDIT that the terminal being used is a video CRT rather than a hard copy terminal. MEDIT chooses different default actions in alter mode when it knows it is working on a video terminal.
Multiple buffers in MEDIT let the user work on more than one file at a time. A buffer can be thought of as a temporary workspace in which a file is edited. When an editing session is begun, MEDIT copies the file to be edited into a buffer. Changes that the user makes affect the buffer and not the original file on disk. Thus, the EQ command may be used at any time to delete the buffer and leave the file on disk in its original state. If the E or W commands are used, MEDIT writes the current buffer out to the file on disk, thus preserving the results of the edit. When multiple buffers are used, each buffer may be manipulated individually as described above. In addition, lines may be copied or transferred between buffers.

15.1. B - Buffer command

The B (buffer) command may be used to create a new buffer, select a different buffer as the current buffer, or type a list of the active buffers.

\*B[n][.[filename][./switches]]

where "n" is a buffer number and "filename" is a new or existing file name. "/switches" may be any switch, but "/CREATE" and "/READONLY" are the most useful ones with this command. Examples:

\*B:F00IR
\*B:ABC/CREATE
\*B:DEF
\*B2
\*B

Creates a buffer to read file F00
Creates a buffer to create ABC
Creates a buffer to edit DEF
Makes buffer 2 the current buffer
Types the active buffer numbers

The E and W commands work on the current buffer. Thus, to write the current buffer to a file, use the W command. To write the current buffer to a file and get rid of the buffer, use the E command. To simply get rid of the current buffer, use the EQ command. When the E command is used on the last remaining buffer, MEDIT will exit to the operating system.

Note: As this manual went to press, MEDIT did not work quite as described above. Currently the E command distinguishes the initial buffer, buffer 0, from all others. If the E command is used on a non-zero buffer, the current buffer will be buffer 0. Buffer 0 is a special case. If the E or EQ commands are used on buffer 0 and there are some non-/READONLY buffers, MEDIT will ask for confirmation. If the user confirms, the other buffers will be EQ'ed.

\[Actually, MEDIT uses a clever scheme to avoid copying the file.\]
15.2. C - Copy command

The C (copy) command copies existing text to a given location in the current buffer. Lines may be copied from the current buffer or from another file or buffer.

15.2.1. Copying from the current file buffer

*Caddress,range[,inc]

The text in "range" is copied and inserted at address. Inc is used as the increment between successive line numbers for the inserted lines.

15.2.2. Copying from another file or buffer

Lines may be copied from another file or buffer using the command syntax

*Caddress=buffer,range[,inc]
*Caddress=buffer

If "buffer" is a file name then a temporary buffer will be created for that file. The buffer will be deleted after the copy is finished. If "buffer" is an existing buffer number then that buffer will be copied from and will not be deleted after the copy.

If the range of lines that are to be copied is known in advance, the first form above may be used. The specified range of lines from the file is copied to "address" in the file being edited. When the range of lines to be copied is not known in advance the second form may be used. This command puts MEDIT into "copy" mode. The prompt for commands is changed from "w" to "C" and any /READONLY command may be issued to examine the file being copied. When the lines to be copied have been located, the copy is completed by typing the E command. At this point, MEDIT will ask for the range of lines to copy and the copy will be performed as originally specified. For example

*C100=foo
Read: F00
C*<commands to locate the text>
C*E
Source Lines: 100/3:8750/4
[32 lines copied from F00 to 100/1]
*

The E command may optionally contain the range of source lines to be copied.

*E[Q][range]

The EQ command may be used while in copy mode if the user decides not to copy any lines.

15.3. T - Transfer command

The T (transfer) command transfers text from one location to another within the current buffer, or from one buffer to another. The syntax of the T command is the same as the syntax of the C command above.
Copying Text and Multiple File Editing

* Taddress, range[, inc]
* Taddress=buffer[, range[, inc]]

The lines in "range" are copied to "address" and then deleted from their original location. "inc" is used as the temporary increment for inserting lines as for the C command.

15.4. Buffer Examples

The following examples illustrate some of the uses of multiple buffers. In the first example, page 2 of file FOO is transferred to a new file named FOO2, and page 3 of FOO is transferred to a new file named FOO3. The other pages of FOO remain intact.

Example 1

@edit foo
Edit: FOO.
*b:foo2/create
Create: FOO2.
1#: Transfer page two of FOO (buffer 0) to FOO2
1$tfoo_0.1/2:*
[78 lines transferred from FOO to 100/1]
1#e
[F002. 1ok!]
*b:foo3/create
Create: FOO3.
1#: Transfer page three of FOO to FOO3
1$tfoo_0.2/3:*
[53 lines transferred from FOO to 100/1]
2#e
[F003. 1ok!]
*; Write out what is left as the new FOO
*e
[F00. 1ok!]
@

In the second example, the user copies selected lines from file BAD into file GOOD, and then deletes file BAD.
Example 2

```
@medit good
Edit: GOOD.
*b:bad
Edit: BAD.
1*ib
Active Buffers

  0      GOOD. (OUTPUT AS) GOOD.

  ==> 1      BAD. (OUTPUT AS) BAD.
```

```
1*ib0
Buffer 0
*; Set up copy from BAD (buffer 1) to GOOD (buffer 0)
*c300_1
CA: Use commands to locate lines to copy, and then
CA: 500:800
[4 lines copied from buffer 1 to 300/1]
*c_1
CA: Some more commands to locate lines to copy
CA: 1000:2000
[10 lines copied from buffer 1 to 800/1]
*b1
Buffer 1
1*ed
Really delete file? y
[BAD. deleted]
*e
[GOOD. 1ok!]
```
CHAPTER 16
PROGRAMMING NEW MEDIT COMMANDS

MEDIT provides several methods for defining new commands. Any user can create a file containing MEDIT commands and invoke it as an indirect command file. The commands may include conditionally executed statements and a rudimentary loop facility. For the user who can program in MAINSAIL, MEDIT provides the ability to invoke a MAINSAIL module (program) written by the user to perform a desired task that cannot easily be performed using existing editor commands. There is also a macro facility in alter mode that allows the user to define new alter mode commands as a sequence of existing commands, as described in Chapter 14.

16.1. - Invoke Indirect Command File

The @ indirect command file) command causes MEDIT to read line editing commands from the specified file instead of from the terminal. When the end of file is reached or an @ command with no arguments is executed, MEDIT will stop reading the file and return to the terminal for the next command.

```
*@[file[,range]]
```

or

```
*@
```

The range passed to the indirect command file is available within the command file as the symbolic range "IR". Thus "IR" may be used where a range is expected by any MEDIT command in the indirect command file.

If no file name is given following the "@" then MEDIT terminates the current indirect command file (if any) without printing a message. This feature is useful for early termination of an indirect command file using the O (on) command described below, or to prevent the message "[End of file]" from being printed.

The @ command may be used only in an indirect command file. It causes MEDIT to start reading the current command file from the beginning again, thus providing a rudimentary loop capability. See the O (on) command below for methods to exit from a loop created by the use of the @ command.

To perform alter mode commands in an indirect command file, use the /ALT CMD switch to set up the desired commands and then invoke the alter command on the desired line.

16.2. Automatic Indirect Command Files

MEDIT invokes certain command files automatically if they happen to exist. When MEDIT is first started, it looks for the file MEDIT.INI in the user's disk area. If a file with
that name is found, it is invoked as an indirect command file automatically. MEDIT.INI is especially useful for initializing switches and parameters to desired values for the duration of the edit. If the file MEDIT.BUF exists in the user’s disk area, it is invoked whenever a new buffer is created, including right after MEDIT.INI is read when MEDIT is first started. MEDIT.BUF is useful for file-dependent parameter initialization. For example, it is possible to set up parameters differently depending on the type of file being edited.

16.3. ; - Comments

MEDIT command lines that begin with a semicolon ";" are ignored. This feature may be used when talking to someone who is "linked" to the user’s terminal. It is also useful for inserting comment lines into indirect command files.

16.4. O - On command

The O (On) command performs a command or group of commands if a certain boolean condition is true at the time the On command is processed. The format of the command is

*Ob,c
or
*Ob;begin [id]
    c
    ...
  *;end [id]

where "b" is a boolean condition (see below), optionally preceded by a "-" to indicate 'not', "c" is any command including another On command, and "id" is an optional block name. The allowed boolean conditions include:

<table>
<thead>
<tr>
<th>Boolean condition</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOF</td>
<td>The current line is at the beginning of the file</td>
</tr>
<tr>
<td>DPY</td>
<td>The /DPY switch is set, i.e. the user has a supported video terminal</td>
</tr>
<tr>
<td>DPY=n</td>
<td>The terminal type is number n</td>
</tr>
<tr>
<td>EOF</td>
<td>The current line is at the end of the file</td>
</tr>
<tr>
<td>FOUND</td>
<td>The most recent F command did not fail</td>
</tr>
<tr>
<td>INITIALINDIRECT</td>
<td>This is the first time through the current indirect file, i.e. no @f command has been executed yet</td>
</tr>
<tr>
<td>MODE=E</td>
<td>C</td>
</tr>
<tr>
<td>NAMEOFFILE=&quot;xxxx&quot;</td>
<td>The name of the file being edited is xxxx</td>
</tr>
<tr>
<td>RANGE</td>
<td>The current line is within the range passed to an indirect file</td>
</tr>
<tr>
<td>SUBSTITUTED</td>
<td>The most recent S command did not fail</td>
</tr>
<tr>
<td>TYPEOFFILE=&quot;xxx&quot;</td>
<td>The extension of the current file is xxx</td>
</tr>
<tr>
<td>SPEED=n</td>
<td>The baud rate of the current terminal line is n</td>
</tr>
</tbody>
</table>
Programming New MEDIT Commands

16.5. Indirect Command File Examples

Most indirect command files are simply one or more commands stored in a file. For example:

```
/pmar:4
/lmar:17
/juef
/pmar:1
/lmar:1
```

Users will often want to set switches in their MEDIT.INI and MEDIT.BUF files, perhaps depending on the type of file being edited and other factors. For example,

```
otype="SAI";begin
  /identifier: "$!,"
  /comment: "Comment ",",""
:end
```

which defines the identifiers and comment format for SAIL program source files.

Using the indirect range ",R" is fairly simple. The following command file will insert a tab at the beginning of each line in the range passed to it:

```
$TAB:3
$TAB:5
$TAB:8
$TAB:11
```

If these commands were placed in a file named "TAB.MED" then they could be performed on lines 100:500 of a file by issuing the command:

```
*tab.med,100:500
```

A loop may be programmed by use of the @+ command. Using @+ is somewhat tricky and is not intended for the average user. The O command may be used to terminate the loop when a condition is met, and to perform some initial commands the first time the loop is executed. The "@" command (without arguments) may be used to terminate the command file at any point. The following general form illustrates how loops may be programmed in indirect command files. The commands in the initialization section are executed the first time through the loop. After the "@" (go to the next line) command is executed, the command file execution is terminated if the end of the indirect range (IR) has been reached, otherwise the command file is reexecuted (but the initialization section is skipped because it is no longer the first time through the file).
Programming New MEDIT Commands

; initial\begin initialization
  \1R
  ...
  ...
; end initialization
  ...
  ...
  +
  o-range, @
  ...
  ...
  @+

16.6. U - Invoke User Program Module

The U command invokes a previously written MAINSAIL module (program) that has been specially coded to interface to MEDIT. MEDIT will call the first interface procedure in the module, which must take one string argument.

*U[:modnam [,args]]

When the U command has been previously given, the name of the module may be left off on subsequent uses and it will default to the previous module. If the optional arguments are given, they will be passed to the user module as a string.

The details of interface of a user module to MEDIT have not yet been established. The file DPYMODMAI that is distributed with MEDIT contains a sample user module. Chapter V contains a description of currently available user modules.
CHAPTER 17
CUSTOMIZATION USING GLOBAL SWITCHES

The default actions taken by MEDIT may be modified by setting global switches in the program. Switches are set using the "/" command, and their current settings are examined using the "=" command. Switch names may be abbreviated to the shortest form that is unambiguous.

17.1. / - Set Switch command

The "/" command sets a global MEDIT switch. Global switches modify the actions taken by various MEDIT commands.

/*switch
 */switch=value

For example

/*AUTOCR

sets the /AUTOCR switch and

/*INCREMENT:5

sets the value of the /INCREMENT switch to 5.

17.2. = - Give Switch Value command

The "=" command causes MEDIT to type out the current setting of a global switch

==switch

17.3. Initial Switch Settings

Switches may be included on the command line that starts MEDIT, e.g.

@MEDIT/READONLY FO0.TXT

They may also be set by putting them in MEDIT.INI or MEDIT.BUF. When MEDIT is first started, it invokes MEDIT.INI as an indirect file. It then invokes MEDIT.BUF as an indirect file. MEDIT.BUF is also invoked every time a new buffer is created by the user. Thus users may customize the actions of MEDIT to their needs, and may have the program behave differently under different situations. For example, switches can be set according to the type of file being edited by using the On command in one of the initialization files (see section 18).
17.4. String Syntax

Some MEDIT switches take strings as arguments. String constant values are specified to MEDIT similarly to a MAINSAIL string constant expression, with the following syntax:

<string expr> ::= <string>
                ::= <string expr> & <string>
                ::=<string>

<string> ::= <primary>
          ::= [<decimal repeat count>]<primary>

<primary> ::= 'octal char code'
           ::= <decimal char code>
           ::= "<string constant>
           ::= <character>

17.5. Table of Global Switches

The following table lists all global switches along with their default values.

/ALTCMDS:"xxx"  (default = "") The commands to execute when altering a line. The user may set up ALTCMDS to perform some editing and then use the Find command with the "A" option to invoke those commands on certain lines. (Temporary switch—may disappear in a future release)

/AUTOCR

/COLUMNS

/AUTOOCR  (default) Turns off AUTOOCR.

=BIG

=COLUMN

=CHECKPOINT:"x","y"

=COLUMNS

=COMMENT:"x","y"

=CHECKPOINT:

=default = 5) The temporary files used by MEDIT to store
Customization Using Global Switches

the edit buffers will be checkpointed every n commands or n lines inserted. (may be changed to checkpoint every n characters typed by the user, in a future release)

/CREATE

Specifies that the file should be created (used only when beginning the edit or creating a new buffer).

/DPY

Says that the terminal being used is a supported video terminal capable of full screen editing. This switch is set automatically if MEDIT's terminal table is correct for a given site.

/NODPY

Says that the terminal being used is not capable of full screen editing.

/FLAG:c

(default = "", accent grave) Specifies the character used to enter control characters. See Appendix I to find out how to use "" to enter control characters. Certain control characters are virtually impossible to enter into the file without using the flag notation (e.g. \c on DEC machines).

/INCREMENT:n

(default = 100) The default increment between line numbers.

/IDENTIFIERS:"xxx"

(default = A-Z, a-z, 0-9) Adds new characters to the list of characters that may appear in identifiers. This switch affects the operation of the special pattern matching characters in searches and the W (word) prefix in the alter mode.

/NOIDENTIFIERS:"xxx"

Removes characters from the list of characters that may appear in identifiers.

/LENGTH:n

Number of lines per screen on the terminal. This switch should be set up correctly if MEDIT's terminal table is correct for a given site.

/LMARGIn:n

(default = 1) The left margin when justification text

/LPTDEVICE

The output device to which the L command sends output. The default is machine dependent (LPT: for the PDP10).

/LPTLENGTH:n

(default = 60) Number of lines per page on the line printer.

/MACDEF:"n":"b"[,m]

Defines an alter mode macro with name "n" and body "b". The third argument, m, specifies the mode in which the macro is available: if it is C then the macro will be available in alter command mode (default), and if it is / then the macro will be available in alter insert mode.
Customization Using Global Switches

/NAME
Prints the name of the file being edited. Currently, this switch cannot be set (except implicitly at the time editing on the file is first initiated).

/NUMBERS
(default) Line numbers should be output on the terminal along with the contents of the lines.

/NONUMBERS
Line numbers should not be output on the terminal.

/PLINES=n
(default = 16) The number of lines to print when the P<CR> command is used.

/PMARGIN=n
(default = 1) The column to indent the first line of a paragraph when justifying.

/READONLY
Specifies that a file is not going to be changed (used only in the initial EDIT command or when creating a new buffer).

/R
Abbreviation for /READONLY

/RECOVER
Asks MEDIT to look for its temporary files and try to recover editing that was going on before a crash (used only in the initial EDIT command).

/RMARGIN=n
(default = 70) The right margin when justifying.

/RUN:ppp
(default = last COMPILe/LOAD/EXECUTE command on the PDP10) The program to run when the G command is used.

/STRIP
(default on non-PDP10 systems) Strip off line numbers when writing the output file.

/NOSTRIP
(default on PDP10 systems) Do not strip off line numbers when writing output file.

/TERMINAL:name
Name of the terminal being used. This switch is set automatically if MEDIT's terminal table is correct for a given computer installation. If not, this switch can be set to specify an existing terminal driver module for the user's terminal.

/TRACE
Causes commands read from an indirect file or initialization file to be echoed on the terminal. This mode is helpful when debugging indirect command files.

/NOTRACE
Turns off tracing of indirect commands (default).
Customization Using Global Switches

/WIDTHn

Width of the terminal (number of characters). This switch is set automatically if MEDIT's terminal table is correct for a given site.

/WINDOW

Causes MEDIT to go into full screen alter mode when it starts up, after processing the automatic indirect command files.

/NOWINDOW

(default) Causes MEDIT to go into line editing mode when it starts up.

/::address

Sets the current position in the file (same as the "::" command in section 12.1). "::=" tells what the current position is and also gives the contents of the PUSH/POP stack of saved file positions.

=?

Prints the names of all MEDIT global switches.
TABLE OF ASCII CHARACTERS

APPENDIX I

TABLE OF ASCII CHARACTERS

This table contains information on the 128 ASCII characters, including their code numbers, the character they represent, the way to type the character when in /Flag mode, and the meaning of the character when it is included in the search string of the F or S commands.
<table>
<thead>
<tr>
<th>Octal</th>
<th>Decimal</th>
<th>Char</th>
<th>Name</th>
<th>Flag</th>
<th>Special Match Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>000</td>
<td>~</td>
<td>NUL</td>
<td>$S</td>
<td>(cannot be put into the file)</td>
</tr>
<tr>
<td>000</td>
<td>001</td>
<td>~A</td>
<td>SOH</td>
<td>$A</td>
<td>matches Alphabets (A-Z,a-z)</td>
</tr>
<tr>
<td>002</td>
<td>002</td>
<td>~B</td>
<td>STX</td>
<td>$B</td>
<td>matches Beginning of line</td>
</tr>
<tr>
<td>003</td>
<td>003</td>
<td>~C</td>
<td>ETX</td>
<td>$C</td>
<td>matches any Character</td>
</tr>
<tr>
<td>004</td>
<td>004</td>
<td>~D</td>
<td>EOT</td>
<td>$D</td>
<td>matches any Digit (0–9)</td>
</tr>
<tr>
<td>005</td>
<td>005</td>
<td>~E</td>
<td>ENQ</td>
<td>$E</td>
<td>matches End of line</td>
</tr>
<tr>
<td>006</td>
<td>006</td>
<td>~F</td>
<td>ACK</td>
<td>$F</td>
<td>matches Lower case (a-z)</td>
</tr>
<tr>
<td>007</td>
<td>007</td>
<td>~G</td>
<td>BEL</td>
<td>$G</td>
<td>Not next character</td>
</tr>
<tr>
<td>010</td>
<td>008</td>
<td>~H</td>
<td>BS</td>
<td>$H</td>
<td>matches One or more of</td>
</tr>
<tr>
<td>011</td>
<td>009</td>
<td>~I</td>
<td>HT</td>
<td>$I</td>
<td>any alphanumerics (A-Z,a-z,0–9)</td>
</tr>
<tr>
<td>012</td>
<td>010</td>
<td>~J</td>
<td>LF</td>
<td>$J</td>
<td>Quotes the next character</td>
</tr>
<tr>
<td>013</td>
<td>011</td>
<td>~K</td>
<td>VT</td>
<td>$K</td>
<td>substitutes nth matching String</td>
</tr>
<tr>
<td>014</td>
<td>012</td>
<td>~L</td>
<td>FF</td>
<td>$L</td>
<td>matches Tab or space</td>
</tr>
<tr>
<td>015</td>
<td>013</td>
<td>~M</td>
<td>CR</td>
<td>$M</td>
<td>matches any Upper case letter (A-Z)</td>
</tr>
<tr>
<td>016</td>
<td>014</td>
<td>~N</td>
<td>SO</td>
<td>$N</td>
<td>Inverse case next match string</td>
</tr>
<tr>
<td>017</td>
<td>015</td>
<td>~O</td>
<td>SI</td>
<td>$O</td>
<td>matches any Word delimiter</td>
</tr>
<tr>
<td>020</td>
<td>016</td>
<td>~P</td>
<td>DLE</td>
<td>$P</td>
<td>matches Zero or more of</td>
</tr>
<tr>
<td>021</td>
<td>017</td>
<td>~Q</td>
<td>DC1</td>
<td>$Q</td>
<td>Upper case next match string</td>
</tr>
<tr>
<td>022</td>
<td>018</td>
<td>~R</td>
<td>DC2</td>
<td>$R</td>
<td>substitutes next match string</td>
</tr>
<tr>
<td>023</td>
<td>019</td>
<td>~S</td>
<td>DC3</td>
<td>$S</td>
<td></td>
</tr>
<tr>
<td>024</td>
<td>020</td>
<td>~T</td>
<td>DC4</td>
<td>$T</td>
<td></td>
</tr>
<tr>
<td>025</td>
<td>021</td>
<td>~U</td>
<td>NAK</td>
<td>$U</td>
<td></td>
</tr>
<tr>
<td>026</td>
<td>022</td>
<td>~V</td>
<td>SYN</td>
<td>$V</td>
<td></td>
</tr>
<tr>
<td>027</td>
<td>023</td>
<td>~W</td>
<td>ETB</td>
<td>$W</td>
<td></td>
</tr>
<tr>
<td>030</td>
<td>024</td>
<td>~X</td>
<td>CAN</td>
<td>$X</td>
<td></td>
</tr>
<tr>
<td>031</td>
<td>025</td>
<td>~Y</td>
<td>EM</td>
<td>$Y</td>
<td></td>
</tr>
<tr>
<td>032</td>
<td>026</td>
<td>~Z</td>
<td>SUB</td>
<td>$Z</td>
<td></td>
</tr>
<tr>
<td>033</td>
<td>027</td>
<td>^A</td>
<td>ESC</td>
<td>^A</td>
<td></td>
</tr>
<tr>
<td>034</td>
<td>028</td>
<td>^B</td>
<td>FS</td>
<td>^B</td>
<td></td>
</tr>
<tr>
<td>035</td>
<td>029</td>
<td>^C</td>
<td>GS</td>
<td>^C</td>
<td></td>
</tr>
<tr>
<td>036</td>
<td>030</td>
<td>^D</td>
<td>RS</td>
<td>^D</td>
<td></td>
</tr>
<tr>
<td>037</td>
<td>031</td>
<td>^E</td>
<td>US</td>
<td>^E</td>
<td></td>
</tr>
<tr>
<td>040</td>
<td>032</td>
<td>^F</td>
<td>SPACE</td>
<td>^F</td>
<td></td>
</tr>
<tr>
<td>041</td>
<td>033</td>
<td>^G</td>
<td>SPACE</td>
<td>^G</td>
<td></td>
</tr>
<tr>
<td>042</td>
<td>034</td>
<td>^H</td>
<td>SPACE</td>
<td>^H</td>
<td></td>
</tr>
<tr>
<td>043</td>
<td>035</td>
<td>^I</td>
<td>SPACE</td>
<td>^I</td>
<td></td>
</tr>
<tr>
<td>044</td>
<td>036</td>
<td>^J</td>
<td>SPACE</td>
<td>^J</td>
<td></td>
</tr>
<tr>
<td>045</td>
<td>037</td>
<td>^K</td>
<td>SPACE</td>
<td>^K</td>
<td></td>
</tr>
<tr>
<td>046</td>
<td>038</td>
<td>^L</td>
<td>SPACE</td>
<td>^L</td>
<td></td>
</tr>
<tr>
<td>047</td>
<td>039</td>
<td>~M</td>
<td>SPACE</td>
<td>~M</td>
<td></td>
</tr>
<tr>
<td>050</td>
<td>040</td>
<td>~N</td>
<td>SPACE</td>
<td>~N</td>
<td></td>
</tr>
<tr>
<td>051</td>
<td>041</td>
<td>~O</td>
<td>SPACE</td>
<td>~O</td>
<td></td>
</tr>
<tr>
<td>052</td>
<td>042</td>
<td>~P</td>
<td>SPACE</td>
<td>~P</td>
<td></td>
</tr>
<tr>
<td>053</td>
<td>043</td>
<td>~Q</td>
<td>SPACE</td>
<td>~Q</td>
<td></td>
</tr>
<tr>
<td>054</td>
<td>044</td>
<td>~R</td>
<td>SPACE</td>
<td>~R</td>
<td></td>
</tr>
<tr>
<td>055</td>
<td>045</td>
<td>~S</td>
<td>SPACE</td>
<td>~S</td>
<td></td>
</tr>
<tr>
<td>056</td>
<td>046</td>
<td>~T</td>
<td>SPACE</td>
<td>~T</td>
<td></td>
</tr>
<tr>
<td>057</td>
<td>047</td>
<td>~U</td>
<td>SPACE</td>
<td>~U</td>
<td></td>
</tr>
<tr>
<td>060</td>
<td>048</td>
<td>~V</td>
<td>SPACE</td>
<td>~V</td>
<td></td>
</tr>
<tr>
<td>061</td>
<td>049</td>
<td>~W</td>
<td>SPACE</td>
<td>~W</td>
<td></td>
</tr>
<tr>
<td>062</td>
<td>050</td>
<td>~X</td>
<td>SPACE</td>
<td>~X</td>
<td></td>
</tr>
<tr>
<td>063</td>
<td>051</td>
<td>~Y</td>
<td>SPACE</td>
<td>~Y</td>
<td></td>
</tr>
<tr>
<td>064</td>
<td>052</td>
<td>~Z</td>
<td>SPACE</td>
<td>~Z</td>
<td></td>
</tr>
<tr>
<td>065</td>
<td>053</td>
<td>^A</td>
<td>SPACE</td>
<td>^A</td>
<td></td>
</tr>
<tr>
<td>066</td>
<td>054</td>
<td>^B</td>
<td>SPACE</td>
<td>^B</td>
<td></td>
</tr>
<tr>
<td>067</td>
<td>055</td>
<td>^C</td>
<td>SPACE</td>
<td>^C</td>
<td></td>
</tr>
<tr>
<td>070</td>
<td>056</td>
<td>^D</td>
<td>SPACE</td>
<td>^D</td>
<td></td>
</tr>
<tr>
<td>071</td>
<td>057</td>
<td>^E</td>
<td>SPACE</td>
<td>^E</td>
<td></td>
</tr>
<tr>
<td>072</td>
<td>058</td>
<td>^F</td>
<td>SPACE</td>
<td>^F</td>
<td></td>
</tr>
<tr>
<td>073</td>
<td>059</td>
<td>^G</td>
<td>SPACE</td>
<td>^G</td>
<td></td>
</tr>
<tr>
<td>074</td>
<td>060</td>
<td>^H</td>
<td>SPACE</td>
<td>^H</td>
<td></td>
</tr>
<tr>
<td>075</td>
<td>061</td>
<td>^I</td>
<td>SPACE</td>
<td>^I</td>
<td></td>
</tr>
<tr>
<td>076</td>
<td>062</td>
<td>^J</td>
<td>SPACE</td>
<td>^J</td>
<td></td>
</tr>
<tr>
<td>077</td>
<td>063</td>
<td>^K</td>
<td>SPACE</td>
<td>^K</td>
<td></td>
</tr>
</tbody>
</table>
## Table of ASCII Characters

<table>
<thead>
<tr>
<th>Octal</th>
<th>Decimal</th>
<th>Char</th>
<th>Name</th>
<th>Flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>084</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>085</td>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>102</td>
<td>086</td>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>103</td>
<td>087</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>104</td>
<td>088</td>
<td>D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>105</td>
<td>089</td>
<td>E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>106</td>
<td>090</td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>107</td>
<td>091</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>108</td>
<td>092</td>
<td>H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>109</td>
<td>093</td>
<td>I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>110</td>
<td>094</td>
<td>J</td>
<td></td>
<td></td>
</tr>
<tr>
<td>111</td>
<td>095</td>
<td>K</td>
<td></td>
<td></td>
</tr>
<tr>
<td>112</td>
<td>096</td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>113</td>
<td>097</td>
<td>M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>114</td>
<td>098</td>
<td>N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>115</td>
<td>099</td>
<td>O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>116</td>
<td>100</td>
<td>P</td>
<td></td>
<td></td>
</tr>
<tr>
<td>117</td>
<td>101</td>
<td>Q</td>
<td></td>
<td></td>
</tr>
<tr>
<td>118</td>
<td>102</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>119</td>
<td>103</td>
<td>S</td>
<td></td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>104</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>121</td>
<td>105</td>
<td>U</td>
<td></td>
<td></td>
</tr>
<tr>
<td>122</td>
<td>106</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>123</td>
<td>107</td>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>124</td>
<td>108</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>125</td>
<td>109</td>
<td>Y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>126</td>
<td>110</td>
<td>Z</td>
<td></td>
<td></td>
</tr>
<tr>
<td>127</td>
<td>111</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>112</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>129</td>
<td>113</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>130</td>
<td>114</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>131</td>
<td>115</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>132</td>
<td>116</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>133</td>
<td>117</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>134</td>
<td>118</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>135</td>
<td>119</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>136</td>
<td>120</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>137</td>
<td>121</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>138</td>
<td>122</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>139</td>
<td>123</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>140</td>
<td>124</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>141</td>
<td>125</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>142</td>
<td>126</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>143</td>
<td>127</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **CARET** (Upward Arrow on some terminals)
- **UNDERSCORE** (Backward Arrow on some terminals)
- **ACCENT GRAVE** (Underscore on some terminals)
MEDIT Command Summary
The line editing and alter mode commands available in MEDIT are summarized here for quick reference by experienced users. The table of line editing commands is output when the "?" command is used in line editing mode in MEDIT, and the table of alter mode commands is output when the "??" command is used in alter mode. Only the syntax of the commands is shown. Refer to the index for the pages on which the command is described in detail.

Line editing commands consist of one or two letters followed by arguments. No spaces need be typed between the command and its arguments, or between arguments. Many commands take a line address or a range of addresses as an argument.

Conventions for Command Syntax

The following conventions are used to describe the syntax of commands.

lower case words meta-symbols that stands for values that should be supplied by the user
[] encloses optional parts of commands
| denotes a choice
UPPER CASE indicates letters that should be typed as part of a command
s A few commands require the use of the <ESC> key (labeled "ESCAPE" or "ALTMODE"). Where <ESC> must be used, a "s" is shown in the syntax.
other symbols must appear in the command as indicated

Initial Command Line

@MEDIT[/switches] input-file[/switches] [output-file[/switches]]
@CREATE[/switches] file>
@EDIT[/switches] input-file [output-file]
MEDIT Line Editing Command Summary

A[range] [.W]
B[n] [bufname]
Caddress, range
Caddress=name[,range[,inc]]
Caddress=n[,range[,inc]]
Drange[,Y]
E[i] [Q] [S] [N] [K] [D] [file]
Fstrings[range] [opt]
G[i] [Q] [S] [N] [K] [file]
H[ELP] [topic]
I[address[,inc]]
Jrange
J[U] [L] [R] [C]range
Krange
Lrange[,N] [.S]
Maddress
N[P][inc][range[,start]]
Oc,command
P(range[,N] [.S]
Qrange[,inc][.Y]
Rstr&strs[range] [opt]
Taddress,range
U[modName] [args]
V[.L] [.U] [.V] range
W[i] [.S] [N] [.K] [file]
Xrange
Y
Z
/s[switch][value]
=switch
:address
- or <ESC>
+ or <LF>
@[file[,range]] [+]
MEDIT Command Summary

MEDIT Alter Mode Command Summary

A
[n]±A
[-][W]B
↑B
[-][n]Cc
[-][n]C
[-][n][W]D
[-][n]<rubout>
[-][n]↑D
E
[-]Fabc<ESC>
[-][n]↑abc<ESC><CR>
[n]labc<ESC>
[-][W]<TAB>
[n][W]J
<LF>
[-][n][W]Kc
l
[-]↑L
M
<CR>
[-][n]N
[n]↑N
P
↑P
Q
[-][n][W]Rstring<ESC>
[W]↑R
[-][n]Sc
[n]↑Str1<ESC>str2<ESC><CR>
[W]T
[n]U
↑U
[-][n][W]V
[-][n][W]<
[-][n][W]>
[n]X
[-]abc<ESC>
[-][n][W]SPACE
[-][n]<BACKSPACE>
.address
(
)
?
<ESC>
macro characters

Perform the previous command again
Attach next line(s) to current line
Go to beginning of line or window
Break line at current position
Change next character to “c”
Copy or unCopy next “n” lines
Delete next “n” characters or words
Delete previous “n” characters
Delete or unDelete lines(s)
End alter mode editing
Incremental find
Find a string
Insert characters
Go to the end of line or window
Justify the current line
Move to next line
Kill (delete) “n” characters or words
Print rest of line, go to beginning
Insert line below or above
Define a macro
End alter mode (single line mode only)
Go to beginning of next line
Renumber lines
Print the line
Insert a page mark
Quit alter mode editing and restore line
Replace next “n” characters
Refresh (retype) line or screen
Skip to nth occurrence of “c”
Substitute str2 for str1
Transpose next two characters or words
Go up to beginning of previous line
Undo editing on current line
inVert case of chars or words
Change chars or words to upper case
Change chars or words to lower case
extend line (insert at end)
Zap (delete) to end of line and insert
Advance “n” characters or words
Go back “n” characters
Go to line address or .PUSH or .POP
Move left to matching left paren
Move right to matching right paren
Give help on alter mode commands
Clear partially typed command
Execute macro command sequence

Invalid commands cause the terminal bell to ring.
APPENDIX III
MEDIT COMPARED TO OTHER EDITORS

Compared to Brookings/NIH SOS

Major differences between MEDIT and various versions of SOS exist in the following commands:

- **A** intraline editing has many more commands and includes a full screen editor
- **B** used to list, change, or delete buffers. To go to beginning of file, use .BF in MEDIT.
- **H** help command takes an optional topic name as an argument
- **O** new command to do conditional commands
- **U** new command to invoke a MAINSAIL user module
- ***** new command to type a message

MEDIT reads the indirect command file MEDIT.INI at the start of each edit rather than examining SWITCH.INI for switch settings. MEDIT.INI may contain switch settings (one per line) or other MEDIT commands (possibly with conditional commands).

The table of control character representations has been changed from the SOS standard to make it easy to remember. The old standard contains such obscure representations as "#" for control-C and "=" for <ESC>. Furthermore, because the single quote "'" is used as the default /C128 character it is necessary to type two single quotes to insert one while in /C128 mode. Also, the special match characters for the F and S commands are completely non-mnemonic, e.g. "[]" means "zero or more of". The default /Flag character in MEDIT is accent grave "\", a character which is not normally used in text. Control characters are entered via accent grave followed by the character, e.g. "C" is control-C and "[" is <ESC>. All of the "F" and "S" MEDIT special match characters are mnemonic. For example, "*O" means "one or more of", "*Z" means "zero or more of", "*C" means any character, "*A" means any alphabetic character, etc. Refer to Appendix I for a list of the /Flag representations of control characters and their special match definitions.

MEDIT supports passing ranges to indirect files, an extension of the Brookings SOS "symbolic ranges". Also, the command "@" by itself in an indirect command file causes the file to terminate without the usual "End of command file" message. This feature is helpful for frequently used command files and when conditionally terminating an indirect file early.

---

2) /Flag mode cannot be used to enter mixed case from an upper-case-only terminal in MEDIT. However, considering the advantage of having mnemonics and the increasing availability of full ASCII terminals, this change should pose no problems for the overwhelming majority of users.
MEDIT Compared to Other Editors

Some of the SOS switches have been replaced or removed. Additional global switches have been added for increased flexibility. The available switches are described in chapter 17.

MEDIT has retained the original default range definition for the F command, i.e. "+1:EF", rather than the Brookings SOS definition "+EF". A similar difference exists in the intraline editing S command when finding a character that is the first in the line. MEDIT and DEC-20 EDIT do not find the first character in the line as does Brookings SOS.

A few of the more obscure Brookings intraline editing commands are different in MEDIT. Brookings SOS uses "T" and "U" in alter mode as word delete and word replace, and it uses "W" as word advance. MEDIT uses "W" to set up the repeat count so that the next alter command is repeated enough times to process the next word in the line. Thus "WD" deletes a word, "W" advances a word, "WR" replaces a word, "WW" inverts the case of a word, etc. MEDIT uses "U" to "go up a line", and "T" to transpose the next two characters in the line. It has many new alter mode commands oriented toward video terminals and full screen editing.

Compared to TOPS20 EDIT

Two major incompatibilities exist between MEDIT and the standard editor for the DECsystem-20. The first is the "B" command: in MEDIT, it creates a new editing buffer, while in TOPS20 edit it writes out the file. Use the "W" command in MEDIT to write out the file. The second is the definition of the flag character (the C128 character in TOPS20 EDIT documentation) and the representation of the control characters using it: in MEDIT, the accent grave (backward quote) character is used as the flag character and the control characters are represented mnemonically, while in TOPS20 EDIT, the single quote is used and the control character representations are obscure abbreviations.

MEDIT has the following improvements over TOPS20 EDIT:

- Full screen editing - A
- Multiple buffers - B
- Conditional commands - O
- Symbolic ranges - BF, EF, IR
- Range passing and loops for indirect command files - IR
- Text justification commands - JW, JL, JR, JC
- Case inversion commands - VI, VU, VV
- Additional special match characters for F, S
- Find decide mode
- Improved substitute decide mode
- Mnemonic control characters and match characters
- El command is parameterized
- Ability to invoke user-written MAINSAIL modules - U

Compared to TVEDIT

MEDIT's alter mode provides capabilities similar to the Stanford TVEDIT video editor. However, the method of command and text entry is different. In TVEDIT, all normal characters typed are interpreted as text to be written over the current text in the file, and all commands must be typed with the "edit key" held down or preceded by <ESC>. In MEDIT, commands are normal characters. Text is inserted or changed using the "I" or "C" commands. <ESC> is used to end insertion or change mode. Thus, no special keyboard is needed and the user need not stray from the "home row" of the keyboard while editing.
MEDIT Compared to Other Editors

Both MEDIT and TVEDIT provide macros. MEDIT allows the user to redefine any key on the terminal to execute any built in command while TVEDIT restricts macros to control characters.

MEDIT’s line editing mode provides powerful commands for global changes to the file. TVEDIT does not provide as much support for global changes (such as text search and substitution with patterns, multiple buffers, etc.). All of MEDIT’s line editing commands and many alter commands may be used effectively on all terminals. TVEDIT cannot be used effectively except on video terminals with the necessary hardware functions. Also, TVEDIT does not provide a linkage to user-written program modules.

Compared to EMACS

The EMACS video editor is really an open-ended collection of TECO macros that provide many video text editing functions. Since it is TECO based, it has the power of a Turing machine and can be programmed by the user. While many man-hours have gone into writing libraries of TECO macros for EMACS, the work has been distributed among many users, each with a different style of programming and user interaction. MEDIT, on the other hand, was designed as a unified program with great thought given to consistency, style, and comprehensibility by non-programmers as well as “hackers”.

Unlike EMACS, MEDIT works very well on all terminal types (including hard copy terminals and glass tty’s) because it provides a comprehensive set of line editing commands and “sticky” line numbers to which they can refer. It is available on more computers (at least potentially) since it is written in a machine independent language. New commands can be programmed in MAINSAIL, an Algol-like language, rather than in the arcane TECO language. Dynamic linking of MAINSAIL modules is provided so that the desired modules may be brought in on demand.

MEDIT’s file management allows editing to be recovered after a system crash without the overhead of saving the entire file on disk every few minutes the way EMACS does. Furthermore, MEDIT does not require the entire file to be read into memory. Thus files larger than the address space of the machine may be edited, and editing large files does not cause degradation of system performance due to thrashing or swapping.

Compared to DEC TECO

Ordinary TECO cannot be compared to MEDIT in a meaningful way. It lacks many of the powerful functions of MEDIT and requires the user to program macros before it can be used effectively. In fact, it should probably be viewed as a programming language rather than as an editor. MEDIT provides many more useful primitives than TECO, and much effort has gone into designing these commands to be easily used.
APPENDIX IV
TEMPORARY FILE IMPLEMENTATION

MEDIT maintains two special temporary files in the user's directory while it is editing. These files contain a string space and pointer space into the file(s) being edited so that random access I/O is possible. The pointer space file contains a two-way linked list of records describing lines of the file being edited. Initially, the records all refer to text in the original input file. Whenever a line is changed, the new text is written to the string space file and the text pointer for that line updated to point to the correct line in the string space file. Thus, the original input file is treated like a read-only memory and is never changed during the editing. Whenever new lines are added to the edit buffer, new records are created in the pointer space file at the end, and inserted into the two-way linked list. When lines are deleted, the records for the lines before and after the deleted lines are linked. Thus, the string space file does not contain ordered lines. Rather, the pointer space file contains the information necessary to determine the correct order of the lines.

A record with no predecessor link is used as the beginningOfFile record, and a record with no successor link is used as the endOfFile record. Thus an empty buffer with no lines or page marks at all contains a beginningOfFile record linked to an endOfFile record. Page marks are represented as records with no text or line numbers associated with them.

When more than one buffer is being edited, the same string space and pointer space files are shared. The beginning of the pointer space actually contains special buffer records which point to the beginningOfFile and endOfFile records for each buffer. No garbage collection of the temporary files is performed, except when a W command is issued and there is only one buffer. In this case, the text pointers in all the records are updated to point to the newly created file, which becomes the new input file, and the string space file can be reclaimed. If the size of the pointer space file ever becomes a problem, it is possible to maintain a "free list" of unused pointers and reuse them, but the overhead involved in maintaining the free list and the resultant non-localized records corresponding to adjacent lines in the file were deemed sufficiently prohibitive to use such an approach.

The linked list organization of the temporary files has many advantages. For example, when moving through the file to locate a particular line number, only the pointer file need be read. The record for the desired line number contains the location of the text of that line, which may then be accessed directly. Random access I/O speeds up movement from place to place in large files and permits added flexibility in commands. For example, moving back and forth between the beginning and end of the file buffer is very fast, as it simply involves reading the location of the beginningOfFile or endOfFile record for the buffer, and then reading in that record from the pointer space. No file output is ever done to reposition within the file, unlike schemes which recopy the file as they move through it (e.g. SOS). Unlike such schemes, and also unlike schemes which keep
Temporary File Implementation

the entire file in random access memory, editing can be recovered easily after a system
crash because the temporary files contain all the information necessary to reconstruct the
editing done so far. The W command is used only when the user is satisfied with the
current state of the edit, rather than to save intermediate results in case the system
 crashes. Thus the overhead of writing a large amount of data to disk at frequent intervals
is avoided. Unlike schemes which modify the original input file, the user can backtrack to
the original state of the file in the event of an emergency.

The temporary files are created in the current "default" file directory for the user,
usually his own directory, and are deleted at the end of a normal edit. Because the output
file must be constructed from the linked list of records anyway, there is no point in
creating the temporary files in the file directory of the file being edited, especially since
the user may not have rename or delete privileges for new files created in that directory.
The highly machine-dependent kludgery of SOS's temporary file management is thus
avoided via MEDIT's scheme.

In typical editing situations, the temporary files created by MEDIT require fewer disk
pages than SOS's temporary files. The savings is quite dramatic when editing large files.
For example, a 400 page input file consisting of 10,000 lines of 80 characters each
causes MEDIT to create a pointer-space file with 10,000 pointers. The resultant 40,000
word file is about 80 pages long, considerably shorter than the two 600 page (remember
line numbers!) files that SOS creates. EMACS and TVEDIT cannot edit a file of this size at
all on the DECSYSTEM-20.
APPENDIX V
USER MODULE LIBRARY

This appendix describes the user modules that can be invoked using the line editing "U" command. Some of these modules perform functions that are machine dependent, and are thus available only for the noted implementations of MEDIT.

**U:GO[.program]**
In the TOPS10, TOPS20, and TENEX implementations, this module chains to the named program, replacing the current invocation of MEDIT. This module is invoked by the G command and should not be invoked directly using the U command. The program supplied to the GO module will be that set by the /RUN switch. On DEC systems, if no program name has been specified, the module will repeat the most recent COMPILE, LOAD, DEBUG, or EXECUTE command issued at monitor level. This feature is useful when debugging programs written in a compiled language.

Note that the SAIL compiler's editor interface is also supported by MEDIT so typing "E" to the SAIL compiler's error handler will cause MEDIT to be invoked and position automatically to the line about which the compiler complained.

**U:JFNS[.bits]**
In the TOPS20 implementation, appends the name of the file being edited or created to the end of the current line of the file. Bits are the JFNS bits documented in the monitor calls manual, in octal.

**U:ODTIM[.bits]**
In the TOPS20 implementation, appends the current date and/or time to the end of the current line of the file. Bits are the JFNS bits documented in the monitor calls manual, in octal.

**U:PUUSH[,program args]**
In the TOPS20 implementation, this module runs the named program in a fork below MEDIT. When the program starts, it will behave as if it was invoked at the EXEC level with a command line containing args. When the program exits, control returns to MEDIT. If no program name is given, the EXEC is invoked below MEDIT.
APPENDIX VI

TERMINAL-DEPENDENT INFORMATION

This appendix contains information relevant to particular terminal types on which
MEDIT is implemented. MEDIT currently supports the following terminals for full screen
editing:

- Visual 200
- DEC VT52
- Heath (Zenith) 19
- Datamedia 2500
- Delta Data 7000 (alias NIH 7000)
- Infoton 400

In addition, less idea video terminals are supported for single line ALTER mode editing using
backspacing and overwriting with spaces, under the generic name "CLASS". All other
terminals are supported under the generic name "HRDCPY", and implement ALTER mode in
the old fashioned SDS way by not showing the text to the right of the cursor. Support
for additional terminals can and will be provided as they evolve. It is relatively easy for a
programmer familiar with the terminal to be supported and one of the above terminals to
add the required support by using an existing support module as a guide.

17.6. Visual 200 Information

The Visual 200 (V200) is supported fully by MEDIT. In addition, when the user
enters ALTER mode, MEDIT redefines the numeric keypad to perform the following
functions:
Terminal-Dependent Information

Key        Function
7          Move to previous window (WU)
8          Move to previous line (U)
9          Move to next window (WN)
4          Move back one character (<BACKSPACE>)
5          Find (∗F)
6          Move forward one character (<SPACE>)
1          Move back one word (-W<SPACE>)
2          Move down one line (N)
3          Move forward one word (W<SPACE>)
0          Delete one character (D)
.          Delete one word (WD)
,          Delete one line (∗D)
-          Insert one line (∗I)
ENTER      Undelete one line (∗A)

Due to the way the special keypad is set up, the "?" command in ALTER mode (give help command) must be entered using two question marks, i.e. "??". Note that it is not necessary to use two question marks to enter a "?" into the file, only when typing it as a command.

Some models of the V200 have special function keys labeled F0, F1, etc. These keys transmit fixed sequences of characters, some of which are valid MEDIT commands and some of which are not. The definition of these keys is left to the individual user who may set them up using a /MACDEF switch in MEDIT.INI.

MEDIT makes certain assumptions about the switch settings on the back of the V200 terminal. If these switch settings are not correct, the editor will not handle the screen properly. These switch settings are consistent with all conventional programs that operate on the DECsystem-10 and DECsystem-20. Switches whose setting is unimportant to MEDIT are not mentioned, but the operating system under which MEDIT runs may require certain settings (e.g. BAUD RATE and PARITY).

AUTO NEW LINE  OFF
AUTO LF/CR     OFF
SCROLL/PAGE    SCROLL
HALF/FULL DUPL  FULL DUPLEX
EMULATION MODE  VISUAL 200

17.7. DEC VT52 Information

The VT52 is supported by MEDIT even though it does not provide insert/delete line and character. Using the VT52 for full screen editing should be tolerable at 9600 baud, but at slower speeds it may be slightly annoying to see the screen repainted whenever a line or character is inserted.
17.8. Heath (Zenith) 19 Information

The Heath 19 (H19) is supported fully, similarly to the Visual 200. The special keypad keys are set up as follows in ALTER mode:

<table>
<thead>
<tr>
<th>Key</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>Up a line (U)</td>
</tr>
<tr>
<td>2</td>
<td>Down a line (N)</td>
</tr>
<tr>
<td>6</td>
<td>Forward a character (&lt;SPACE&gt;)</td>
</tr>
<tr>
<td>4</td>
<td>Back a character (&lt;BACKSPACE&gt;)</td>
</tr>
<tr>
<td>3</td>
<td>Forward a word (W&lt;SPACE&gt;)</td>
</tr>
<tr>
<td>1</td>
<td>Back a word (-W&lt;SPACE&gt;)</td>
</tr>
<tr>
<td>9</td>
<td>Next window (WN)</td>
</tr>
<tr>
<td>7</td>
<td>Previous window (WU)</td>
</tr>
<tr>
<td>5</td>
<td>Find (±F)</td>
</tr>
<tr>
<td>0</td>
<td>Delete character (D)</td>
</tr>
<tr>
<td>.</td>
<td>Delete word (WD)</td>
</tr>
<tr>
<td>ENTER</td>
<td>Delete line (±D)</td>
</tr>
</tbody>
</table>

As for the Visual 200, two question marks "??" must be entered to get help, rather than just one.

For MEDIT to drive the H19 properly, it must be in its native mode rather than the ANSI mode, and it must not insert free carriage returns or linefeeds.

17.9. Datamedia 2500 Information

The Datamedia 2500 (DM2500) is supported fully.

17.10. Delta Data 7000 (alias NIH 7000) Information

The NIH 7000 (N7000) is supported fully, but as of this writing the code has not been debugged on an actual N7000. Eventually, special keypad features of the NIH7000 will also be supported. There may also be some restrictions on the modes which the terminal must be in, but these will likely be set by MEDIT automatically.

17.11. Infoton 400 Information

The Infoton 400 is supported fully. Its numeric keypad and some of its special orange keys are set up as follows:
### Terminal-Dependent Information

<table>
<thead>
<tr>
<th>Key</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>Previous line</td>
</tr>
<tr>
<td>2</td>
<td>Next line</td>
</tr>
<tr>
<td>6</td>
<td>Forward character</td>
</tr>
<tr>
<td>1</td>
<td>Back a word</td>
</tr>
<tr>
<td>3</td>
<td>Forward word</td>
</tr>
<tr>
<td>9</td>
<td>Next window</td>
</tr>
<tr>
<td>7</td>
<td>Previous window</td>
</tr>
<tr>
<td>5</td>
<td>Find</td>
</tr>
<tr>
<td>?</td>
<td>Insert</td>
</tr>
<tr>
<td>?</td>
<td>Delete char</td>
</tr>
<tr>
<td>?</td>
<td>Delete word</td>
</tr>
<tr>
<td>?</td>
<td>Delete line</td>
</tr>
<tr>
<td>?</td>
<td>Undelete Line</td>
</tr>
</tbody>
</table>

Unfortunately, the special function keys (F0, F1, ...) are not usable because they do not transmit uniquely recognizable sequences of characters.
INDEX

+ Command 36
- Command 36
. Command 40
/ Command 67
//address 71
? 71
/ALTCMDs 66
/AUTOOCR 68
/BRIG 69
/Checkpoint 68
/COLUMN 68
/COMMENT 68
/CREATE 69
/COPY 69
/FLAG 69
/IDENTIFIERS 69
/INCREMENT 69
/LENGTH 69
/LMARGIN 69
/LPTDEVICE 69
/LPTLENGTH 69
/MACDEF 69
/NAME 69
/NODAUTOOCR 69
/NODPY 69
/NOCIDENTIFIERS 69
/NONUMBERS 70
/NOSTRIP 70
/NOTRACE 70
/NOWINDOW 71
/NUMBERS 70
/PAGES 70
/PMARGIN 70
/READONLY 70
/RECOVER 70
/RMARGIN 70
/RUN 70
/STRIP 70
/Terminal 70
/TRACE 70
/WIDTH 70
/VIEW 70
/<ESC> 6, 12
/<LF> 36
= Command 67
@ files 63
Aborting an edit 37
Addresses of lines 30
Alter command 48
Alter Mode 48

ASCII Character Table 72
Automatic Indirect Command Files 63
B command 59
BF 31
Buffers 59

Command Files 63
Command Summary 76
Comments 64
Copy command 60
Copying text 59
Creating files 5, 28
Current line 11

Decide mode 43
Default parameters 67
Delete command 33

Editing existing files 7, 28
Editing mode 6
EF 9, 11, 31
End-edit command 37
Exiting from MEDIT 10, 37

Find command 12, 42
Full Screen Editing 21, 24

Give Switch Value command 67
Global Switches 67
Go command 38

Help command 17

Indirect command files 18, 63
Initial switch settings 67
Input mode 6
Insert command 33

JC 19
JK 18
Join command 18, 25
JR 18
JU 19
Justify 26
Justify Center 25
Justify Left 25
Justify Right 25
Justify Without padding 25
JWW 18

Kill-page-mark command 41

Line editing 8
Line Numbers 30
List command 36
Locating Text 12
Index

Margins 25
Mark command 40
MEDIT.BUF 39, 56, 64, 65, 67
MEDIT.INI 23, 28, 34, 39, 56, 57, 63, 64, 65, 67, 79
Minus command 12

Number command 16, 17, 41

On command 64
Option file 26

Page Numbers 30
Paragraph 26
Parameters 67
Patterns 45
Plus command 12
POP 40, 52, 78
Print command 35
Programming 63
PUSH 40, 52, 78, 84

R 70
Ranges 30
Renumbering lines 16, 17
Replace command 34
Replacement strings 46

Saving an edit 10
Search strings 45
Searching for text 42
Set Switch command 67
Special matching characters 45
Storing the file 10
Substitute command 13, 44
Summary of Commands 78
Switches 67

Text Justification 25
Transfer command 60

User Modules 66

V commands 36
Video Editing 48
Video Terminals 21

Write command 38
Writing the file 10

Xtend Command 15