V12

TM

# MAINSAIL®

MAINEDIT™ User's Guide

# MAINEDIT™ User's Guide

24 March 1989

xidak™

## Table of Contents

1. Basic Concepts . . . . . 1
1.1. Version . . . . . 1
1.2. Changes to Commands . . . . . 1
1.3. Using This Manual . . . . . 1
1.4. Notation Conventions . . . . . 2
1.4.1. File Names . . . . . 2
1.4.2. Special Keys . . . . . 2
1.4.3. Dialogues . . . . . 3
1.4.4. Command Descriptions . . . . . 3
1.5. Files, Buffers, and Windows . . . . . 3
1.6. Front Ends and Back Ends . . . . . 4
1.7. Establishing the Default Front End . . . . . 4
1.8. Invoking MAINEDIT . . . . . 5
1.8.1. File to Edit . . . . . 5
1.8.2. Display Module and Baud Rate . . . . . 6
1.8.3. New File . . . . . 7
1.9. MAINEDIT with Command Line Arguments . . . . . 7
1.10. Screen Format . . . . . 8
1.10.1. Message Line . . . . . 8
1.10.2. Status Line . . . . . 9
1.10.3. Page Marks . . . . . 10
1.10.4. Other Special Characters . . . . . 10
1.10.5. Cursor . . . . . 10
1.11. Tabs . . . . . 10
1.12. Finishing the Edit Session . . . . . 11

2. The "eparms" File . . . . . 12
2.1. "EDITORPARMSFILE" . . . . . 12
2.2. "CONTEXT" . . . . . 12
2.3. "DISPLAYMODULE" . . . . . 13
2.4. "UPDATEDISPLAYMODULE" . . . . . 13
2.5. "DONOTPROMPTFORDISPLAYMODULE" . . . . . 13
2.6. "BAUDRATE", "UPDATEBAUDRATE", and "DONOTPROMPTFORBAUDRATE" . . . . . 14
2.7. "MACRO" and "NAMEDMACRO" . . . . . 14
2.8. "WINDOWWIDTH" . . . . . 15
2.9. "DONOTUPDATEEPARMS" . . . . . 15
2.10. Changing an "eparms" File . . . . . 16
2.11. "FRONTEND" and "BACKEND" . . . . . 16

3. Switching among Front Ends . . . . . 17

I. MAINED User's Guide . . . . . . . . . . . . . . . . . . . . . . . . 19

4. MAINED Basics . . . . . . . . . . . . . . . . . . . . . . . . 20
4.1. Notation Conventions . . . . . . . . . . . . . . . . . . . . 20
4.1.1. Case . . . . . . . . . . . . . . . . . . . . . . . . 20
4.2. Starting and Ending a MAINED Session . . . . . . . . . . . . 20
4.2.1. Initial Page and Line . . . . . . . . . . . . . . . . . . 20
4.2.2. Finishing the Edit Session . . . . . . . . . . . . . . . . 21
4.3. Modes . . . . . . . . . . . . . . . . . . . . . . . . 21
4.3.1. Command Mode . . . . . . . . . . . . . . . . . . . . 21
4.3.2. Insert Mode . . . . . . . . . . . . . . . . . . . . 22
4.3.3. Overstrike Mode . . . . . . . . . . . . . . . . . . . 22
4.3.4. Escape Mode . . . . . . . . . . . . . . . . . . . . 22
4.3.5. Returning to Command Mode . . . . . . . . . . . . . . 22
4.4. Basic Editing Commands . . . . . . . . . . . . . . . . . 22
4.4.1. Moving the Cursor . . . . . . . . . . . . . . . . . . 23
4.4.2. Windowing . . . . . . . . . . . . . . . . . . . . 23
4.4.3. Deleting . . . . . . . . . . . . . . . . . . . . . 24
4.5. Current Character, Word, Line, and Page . . . . . . . . . . 24
4.6. MAINED Window Appearance . . . . . . . . . . . . . . . 24
4.6.1. Margins . . . . . . . . . . . . . . . . . . . . . 25
4.6.2. Buffer End . . . . . . . . . . . . . . . . . . . . 25
4.7. Pages and Page Marks . . . . . . . . . . . . . . . . . 25
4.8. Buffer Names . . . . . . . . . . . . . . . . . . . . 26
4.9. MAINED Special Keys . . . . . . . . . . . . . . . . . 26

5. Command Syntax . . . . . . . . . . . . . . . . . . . . 28
5.1. Direction ("+" or "-") . . . . . . . . . . . . . . . . . 28
5.2. Count (n) . . . . . . . . . . . . . . . . . . . . . 28
5.3. Emphasis ("Q") . . . . . . . . . . . . . . . . . . . 29
5.4. Basic Command . . . . . . . . . . . . . . . . . . . 29
5.5. Object . . . . . . . . . . . . . . . . . . . . . . 29

6. Cursor Movement and Windowing . . . . . . . . . . . . . . 30
6.1. Basic Cursor Movement . . . . . . . . . . . . . . . . . 30
6.2. Go to Specified Page and Line . . . . . . . . . . . . . . 30
6.3. Go to Specified Character Position . . . . . . . . . . . . 31
6.4. Skip to Character or Line . . . . . . . . . . . . . . . . 32
6.5. Set and Jump to Mark . . . . . . . . . . . . . . . . . 33
6.6. Text Search . . . . . . . . . . . . . . . . . . . . 33
6.7. Scroll Window . . . . . . . . . . . . . . . . . . . 35

7. Overstriking . . . . . . . . . . . . . . . . . . . . . 36
7.1. Overstrike Mode . . . . . . . . . . . . . . . . . . . 36

7.2. Special Keys in Overstrike Mode . . . . . . . . . . . . . . . . . . . . 36
7.3. Overstriking a Sequence of Characters . . . . . . . . . . . . . . . . . . 36

8. Inserting . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 38
8.1. Insert Mode . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 38
8.2. Special Keys in Insert Mode . . . . . . . . . . . . . . . . . . . . . . 38
8.3. Inserting Lines or Characters . . . . . . . . . . . . . . . . . . . . . 38
8.4. Inserting the Contents of a Buffer or File . . . . . . . . . . . . . . . 39
8.5. Inserting Page Marks . . . . . . . . . . . . . . . . . . . . . . . . . 39
8.6. Inserting Characters by Code . . . . . . . . . . . . . . . . . . . . . 40

9. Deleting and Recalling Text . . . . . . . . . . . . . . . . . . . . . . 42
9.1. Character, Word, and Line Delete Buffers . . . . . . . . . . . . . . . 42
9.2. Page Buffers . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 42
9.3. Delete . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 42
9.4. Zap . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43
9.5. Recalling Text . . . . . . . . . . . . . . . . . . . . . . . . . . . . 44

10. Other Text Modifying Commands . . . . . . . . . . . . . . . . . . . . 46
10.1. Center . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 46
10.2. Copy . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 46
10.3. Move . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 47
10.4. Break . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 48
10.5. Join . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 48
10.6. Convert to Lower or Upper Case . . . . . . . . . . . . . . . . . . . 49
10.7. Filling and Justification . . . . . . . . . . . . . . . . . . . . . . 50

11. Windows . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 55
11.1. Changing Window Size . . . . . . . . . . . . . . . . . . . . . . . . 55
11.2. Anchoring Windows . . . . . . . . . . . . . . . . . . . . . . . . . . 56

12. Files and Buffers . . . . . . . . . . . . . . . . . . . . . . . . . . . 59
12.1. Bringing a File into the Editor . . . . . . . . . . . . . . . . . . . 59
12.2. Moving the Cursor among Buffers . . . . . . . . . . . . . . . . . . . 59
12.3. Modifiers of the ".F" and ".B" Commands . . . . . . . . . . . . . . . 61
12.4. Deleting Buffers . . . . . . . . . . . . . . . . . . . . . . . . . . . 62
12.5. Changing Buffer and File Names . . . . . . . . . . . . . . . . . . . . 62
12.6. Displaying Information about Buffers and Files . . . . . . . . . . . . 63
12.7. Specifying Front and Back Ends for a Buffer; Multiple Buffers into the Same Data . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 64

13. Command and View Front Ends . . . . . . . . . . . . . . . . . . . . . . 66

14. Saving and Finishing . . . . . . . . . . . . . . . . . . . . . . . . . . 67

15. Miscellaneous Commands . . . . . . . . . . . . . . . . . . . . . . . . 69
15.1. <abort> . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 69

15.2. Again . . . . . . . . . . . . . . . . . . . . . . 69
15.3. Undo . . . . . . . . . . . . . . . . . . . . . . 70
15.4. Status Report . . . . . . . . . . . . . . . . . . . . 71
15.5. Suppress Output . . . . . . . . . . . . . . . . . . . 71
15.6. Refresh Screen . . . . . . . . . . . . . . . . . . . 72
15.7. Setting Tab Stops . . . . . . . . . . . . . . . . . . 72
15.8. Changing the Frequency of Save Reminders . . . . . . . . . . 73

16. Macros . . . . . . . . . . . . . . . . . . . . . . 74
16.1. Defining a Macro . . . . . . . . . . . . . . . . . . 74
16.2. Invoking a Macro . . . . . . . . . . . . . . . . . . 75
16.3. Recursive Macros . . . . . . . . . . . . . . . . . . 75
16.4. Macros and Editor Modes . . . . . . . . . . . . . . . 77
16.5. Special Macro "INITIALIZE" . . . . . . . . . . . . . . 77

17. Arithmetic Functions . . . . . . . . . . . . . . . . . 79

18. Setting and Clearing Options . . . . . . . . . . . . . . 82
18.1. stickyTabs Option . . . . . . . . . . . . . . . . . . 83
18.2. readOnly Option . . . . . . . . . . . . . . . . . . 84
18.3. Border Options . . . . . . . . . . . . . . . . . . . 84
18.4. wordWrap Option . . . . . . . . . . . . . . . . . . 84
18.5. proportionalWindowsMode Option . . . . . . . . . . . . 85
18.6. insertModeDefault and overstrikeModeDefault Options . . . . . . . . 85
18.7. Case Options . . . . . . . . . . . . . . . . . . . . 85

19. Using MAINED to Invoke Other MAINSAIL Modules . . . . . . . . . . 86
19.1. Executing a MAINSAIL Module . . . . . . . . . . . . . 86
19.2. Escaping to Caller . . . . . . . . . . . . . . . . . . 87
19.3. Sample Invocation of the MAINSAIL Compiler from MAINED . . . . . 88
19.4. Simultaneous File Access Caveat . . . . . . . . . . . . . 93

II. MAINVI User's Guide . . . . . . . . . . . . . . . . . 103

20. MAINVI Introduction . . . . . . . . . . . . . . . . . 104

21. Differences between MAINVI and vi . . . . . . . . . . . . 105
21.1. Initialization . . . . . . . . . . . . . . . . . . . 105
21.2. Status Line . . . . . . . . . . . . . . . . . . . . 105
21.3. Tabs . . . . . . . . . . . . . . . . . . . . . . 105
21.4. Lines Too Long to Fit on the Screen . . . . . . . . . . . . 106
21.5. Display of Control Characters . . . . . . . . . . . . . . 106
21.6. Page Marks (CTRL-L in a File) . . . . . . . . . . . . . 106
21.7. Deletions during Input Mode . . . . . . . . . . . . . . 106
21.8. CTRL-L Command . . . . . . . . . . . . . . . . . . 107

21.9. Shell Commands . . . . . . . . . . . . . . . . . . . . . 107
21.10. Delete Buffers and Text Recovery . . . . . . . . . . . . . 107
21.11. Macros . . . . . . . . . . . . . . . . . . . . . . . . . 107
21.12. Ignored Modes and Variables . . . . . . . . . . . . . . . . 107
21.13. tabstop . . . . . . . . . . . . . . . . . . . . . . . . 108
21.14. Undoing Commands . . . . . . . . . . . . . . . . . . . . 108
21.15. Marks . . . . . . . . . . . . . . . . . . . . . . . . . 108
21.16. Quoting Input Characters . . . . . . . . . . . . . . . . . 109
21.17. Tags . . . . . . . . . . . . . . . . . . . . . . . . . . 109
21.18. "=" (Indent for LISP) . . . . . . . . . . . . . . . . . . 109
21.19. Crash Facilities . . . . . . . . . . . . . . . . . . . . 109
21.20. Other Unimplemented Commands . . . . . . . . . . . . . . . 109
21.21. File Argument List . . . . . . . . . . . . . . . . . . . 109
21.22. Reading a File . . . . . . . . . . . . . . . . . . . . . 109
21.23. Autoindent Space and CTRL-D in Input Mode . . . . . . . . . 110
21.24. Warnings about Switching among Buffers . . . . . . . . . . 110
21.25. The Bell . . . . . . . . . . . . . . . . . . . . . . . . 110
21.26. Error Messages . . . . . . . . . . . . . . . . . . . . . 110
21.27. Printing the Current Line in ex Mode . . . . . . . . . . . 110

22. Commands and Variables Added to MAINVI . . . . . . . . . . . . 111
22.1. Commands . . . . . . . . . . . . . . . . . . . . . . . . 111
22.1.1. "g": Moving to a Page Mark . . . . . . . . . . . . . . . 111
22.1.2. :buffer . . . . . . . . . . . . . . . . . . . . . . . . 111
22.1.3. :hredo, :hundo . . . . . . . . . . . . . . . . . . . . 112
22.1.4. :invoke . . . . . . . . . . . . . . . . . . . . . . . . 112
22.1.5. :leftcolumn . . . . . . . . . . . . . . . . . . . . . . 113
22.1.6. :lset, :set . . . . . . . . . . . . . . . . . . . . . . 113
22.1.7. :unabbreviateall, :unmapall . . . . . . . . . . . . . . 113
22.2. Variables . . . . . . . . . . . . . . . . . . . . . . . . 114
22.2.1. leftBorderChar, rightBorderChar (Numeric) . . . . . . . . 114
22.2.2. width (Numeric) . . . . . . . . . . . . . . . . . . . . 114
22.2.3. pageAndLinePrompt . . . . . . . . . . . . . . . . . . . 114
22.2.4. replaceWithoutPrompt . . . . . . . . . . . . . . . . . 114

23. Known MAINVI Problems . . . . . . . . . . . . . . . . . . . . 115

III. MEDT User's Guide . . . . . . . . . . . . . . . . . . . . . 117

24. MEDT Introduction . . . . . . . . . . . . . . . . . . . . . . 118

25. Differences between MEDT and EDT . . . . . . . . . . . . . . . 119
25.1. Terminals . . . . . . . . . . . . . . . . . . . . . . . . 119
25.2. Initialization Files . . . . . . . . . . . . . . . . . . . 119
25.3. EDT Line Mode . . . . . . . . . . . . . . . . . . . . . . 119

25.4. Initial Mode . . . . . . . . . . . . . . . . . . . . . . . 120
25.5. Display of Tabs and Control Characters . . . . . . . . . . . . . . . . 121
25.6. Word Entity and Tabs . . . . . . . . . . . . . . . . . . . . 121
25.7. Searches and Tabs . . . . . . . . . . . . . . . . . . . . . 121
25.8. End-of-Line Character . . . . . . . . . . . . . . . . . . . . 121
25.9. Message and Status Lines . . . . . . . . . . . . . . . . . . . 121
25.10. Borders and the Display of End of Buffer . . . . . . . . . . . . . . . 122
25.11. Keypad in Nokeypad Mode . . . . . . . . . . . . . . . . . . 122
25.12. Select Range . . . . . . . . . . . . . . . . . . . . . . 122
25.13. CTRL-C and CTRL-Z . . . . . . . . . . . . . . . . . . . . 122
25.14. Main Buffer Name . . . . . . . . . . . . . . . . . . . . . 122
25.15. Buffer Name Formats . . . . . . . . . . . . . . . . . . . . 122
25.16. Select Range . . . . . . . . . . . . . . . . . . . . . . 123
25.17. Sequence Numbers . . . . . . . . . . . . . . . . . . . . . 123
25.18. Specifying Line Numbers in Line Mode Commands . . . . . . . . . . 123
25.19. "EXIT" and File Names . . . . . . . . . . . . . . . . . . . 124
25.20. "WRITE" and File Names . . . . . . . . . . . . . . . . . . . 124
25.21. "CLEAR" . . . . . . . . . . . . . . . . . . . . . . . 124
25.22. Ignored "SET/SHOW" Options . . . . . . . . . . . . . . . . . 124
25.23. "SET SEARCH" . . . . . . . . . . . . . . . . . . . . . . 125
25.24. "SET COMMAND" . . . . . . . . . . . . . . . . . . . . . 125
25.25. "SET LINES" . . . . . . . . . . . . . . . . . . . . . . 125
25.26. "SHOW BUFFER" and "SHOW FILES" . . . . . . . . . . . . . . 125
25.27. "." at End of Key Definition . . . . . . . . . . . . . . . . . 125
25.28. "DEFK" Limitations . . . . . . . . . . . . . . . . . . . . 126
25.29. Leaving Escape Mode . . . . . . . . . . . . . . . . . . . . 126

26. MEDT Extensions . . . . . . . . . . . . . . . . . . . . . . . 127
26.1. "(NO)LEFTBORDER", "(NO)RIGHTBORDER" . . . . . . . . . . . . 127
26.2. Invoking a Program . . . . . . . . . . . . . . . . . . . . 127
26.3. Switching Buffers . . . . . . . . . . . . . . . . . . . . . 127
26.4. Switching Files . . . . . . . . . . . . . . . . . . . . . . 128
26.5. Keypad Mode Key Definitions . . . . . . . . . . . . . . . . . 128
26.6. Key Numbering and Naming . . . . . . . . . . . . . . . . . 128
26.7. Front Ends and Views . . . . . . . . . . . . . . . . . . . . 130

## Appendices

A. MAINED Command Summary . . . . . . . . . . . . . . . . . . . . . . . 94

B. The DATMGR Back End . . . . . . . . . . . . . . . . . . . . . . . 131

C. Display Modules . . . . . . . . . . . . . . . . . . . . . . . 133
C.1. Display Modules AM48 and AM60 . . . . . . . . . . . . . . . . 134
C.2. Apollo Display Modules . . . . . . . . . . . . . . . . . . . . 135
C.3. Display Module AT386 . . . . . . . . . . . . . . . . . . . . 137
C.4. Display Module BIGSUN . . . . . . . . . . . . . . . . . . . . 138
C.5. Display Module D400 . . . . . . . . . . . . . . . . . . . . 141
C.6. Display Modules D460 and D460C . . . . . . . . . . . . . . . . 142
C.7. Display Module DATAME . . . . . . . . . . . . . . . . . . . . 143
C.8. Display Module EWY100 . . . . . . . . . . . . . . . . . . . . 144
C.9. Display Module HEATH . . . . . . . . . . . . . . . . . . . . 145
C.10. Display Modules HP300H and HPTERM . . . . . . . . . . . . . . 147
C.11. Display Module LINDPY . . . . . . . . . . . . . . . . . . . . 148
C.12. Display Modules SUN, SUN3, and SUN46 . . . . . . . . . . . . 150
C.13. Display Modules TELEVI and TVI950 . . . . . . . . . . . . . . 151
C.14. Display Module TRMCAP . . . . . . . . . . . . . . . . . . . . 152
C.15. Display Module VIS550 . . . . . . . . . . . . . . . . . . . . 153
C.16. Display Modules VT100, VT102, and VT102M . . . . . . . . . . . 154
C.17. Display Modules WY43, WY50, WY5043, and WY75 . . . . . . . . . 155

## List of Examples

1.4.3-1. Sample Prompt and User Response . . . . . . . . . . . . . . . . . . . . 3
1.8.1-1. Answering the "File to Edit" Prompt . . . . . . . . . . . . . . . . . . . 5
1.8.2-1. Display Module and Baud Rate Prompts . . . . . . . . . . . . . . . . . 7
2.10-1. An Initial "eparms" File . . . . . . . . . . . . . . . . . . . . . . . . 16
4.6.1-1. Text beyond the Margin . . . . . . . . . . . . . . . . . . . . . . . 25
5.2-1. Examples of the Use of Counts . . . . . . . . . . . . . . . . . . . . 28
6.3-1. Use of the "V" Command . . . . . . . . . . . . . . . . . . . . . . . 32
8.6-2. Four Ways to Insert an ASCII Delete Character . . . . . . . . . . . . . . 41
10.7-2. ".J" Command, Example 1 . . . . . . . . . . . . . . . . . . . . . . 52
10.7-3. ".J" Command, Example 2 . . . . . . . . . . . . . . . . . . . . . . 53
10.7-4. ".J" Command, Example 3 . . . . . . . . . . . . . . . . . . . . . . 54
11-1. A Screen with Two Windows . . . . . . . . . . . . . . . . . . . . . . 55
11.1-2. Left-Right Scrolling . . . . . . . . . . . . . . . . . . . . . . . . . 57
12.3-1. Sample Use of ".B" . . . . . . . . . . . . . . . . . . . . . . . . . 61
12.6-1. Sample Output from the "Q=" Command . . . . . . . . . . . . . . . . 64
16.3-1. Example of a Recursive Macro Definition . . . . . . . . . . . . . . . . 76
16.5-1. Setting Tabs to Be Every Fourth Column . . . . . . . . . . . . . . . . 78
17-2. Valid Numeric Forms for Arithmetic Commands . . . . . . . . . . . . . 81
18.1-1. How the stickyTabs Option Affects the Insertion of Tab Characters . . . . 83
19.3-1. Initial Screen Configuration . . . . . . . . . . . . . . . . . . . . . 88
19.3-2. After Issuing the Command "QEcompil<eol>" . . . . . . . . . . . . . . 89
19.3-3. After Typing "poem<eol>" . . . . . . . . . . . . . . . . . . . . . . 89
19.3-4. After Typing "<ecm>QY1G" in CMDLOG . . . . . . . . . . . . . . . . 90
19.3-5. Examining the File "poet" with ".Fpoet<eol><eol>" . . . . . . . . . . . 90
19.3-6. After Returning to COMPIL with the "E" Command . . . . . . . . . . . 91
19.3-7. Terminating the Compilation with "a c<eol>" . . . . . . . . . . . . . . 91
19.3-8. Compiling the Correct File . . . . . . . . . . . . . . . . . . . . . . 92
19.3-9. After Ending the Compiler Session . . . . . . . . . . . . . . . . . . . 92
B-1. Sample Buffer Using Back End DATMGR . . . . . . . . . . . . . . . . . 131

## List of Figures

1.10-1. MAINEDIT Screen Format . . . . . . . . . . . . . . . . . . . . . . . 9
10.7-1. Commands to Fill and Justify Text . . . . . . . . . . . . . . . . . . . 50
11.2-1. Commands Used to Anchor and Unanchor Windows . . . . . . . . . . . 58
12.7-1. How to Get Multiple Buffers into the Same Data . . . . . . . . . . . . 65
18-2. Options . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 82
C.9-1. Sample Heath Switch Settings . . . . . . . . . . . . . . . . . . . . . 146

## List of Tables

1.4.2-1. Key Name Abbreviations . . . . . . . . . . . . . . . . . . . . . . 2
4.4.1-1. Basic Cursor Movement Commands . . . . . . . . . . . . . . . . . 23
4.4.2-1. Basic Scrolling Commands . . . . . . . . . . . . . . . . . . . . 23
4.4.3-1. Basic Commands for Deleting Text . . . . . . . . . . . . . . . . 24
4.9-1. Key Name Abbreviations . . . . . . . . . . . . . . . . . . . . . . 26
6.2-1. Page- and Line-Relative Motion Commands . . . . . . . . . . . . . 30
6.1-1. Basic Cursor Positioning Commands . . . . . . . . . . . . . . . . 31
6.4-1. Commands to Skip to a Character or Line . . . . . . . . . . . . . 32
6.5-1. Commands to Control and Use the Mark . . . . . . . . . . . . . . 33
6.6-1. Text Search Commands . . . . . . . . . . . . . . . . . . . . . . . 33
6.7-1. Commands for Scrolling Windows . . . . . . . . . . . . . . . . . 35
7.2-1. Effect of Special Keys in Overstrike Mode . . . . . . . . . . . . 36
7.3-1. Commands to Overstrike Text . . . . . . . . . . . . . . . . . . . 37
8.1-1. Commands to Enter Insert Mode . . . . . . . . . . . . . . . . . . 38
8.2-1. Effect of Special Keys in Insert Mode . . . . . . . . . . . . . . 39
8.3-1. Commands to Insert Characters or Lines . . . . . . . . . . . . . 39
8.4-1. Commands to Insert Buffers or Files . . . . . . . . . . . . . . . 40
8.6-1. The Four Radices Understood by the "'" Command . . . . . . . . . 40
9.3-1. Text Deletion Commmands . . . . . . . . . . . . . . . . . . . . . 43
9.4-1. The Zap Commands . . . . . . . . . . . . . . . . . . . . . . . . . 44
9.5-1. Text Recalling (Undeletion) Commands . . . . . . . . . . . . . . 45
10.1-1. Commands to Center Text on a Line . . . . . . . . . . . . . . . 46
10.2-1. Commands to Copy Text . . . . . . . . . . . . . . . . . . . . . 46
10.3-1. Text Rearranging Commands . . . . . . . . . . . . . . . . . . . 47
10.4-1. Commands to Break a Line . . . . . . . . . . . . . . . . . . . . 49
10.5-1. Commands to Join Two Lines . . . . . . . . . . . . . . . . . . . 49
10.6-1. Commands to Convert to Lower/Upper Case . . . . . . . . . . . . 49
11.1-1. Window Management Commands . . . . . . . . . . . . . . . . . . 56
12.1-1. File Selection Commands . . . . . . . . . . . . . . . . . . . . 60
12.2-1. Buffer Selection Commands . . . . . . . . . . . . . . . . . . . 60
12.5-1. Commands to Change Buffer and File Names . . . . . . . . . . . . 63
14-1. Commands to Save Files and/or Exit from MAINEDIT . . . . . . . . 67
15.2-1. Forms of the Again Command . . . . . . . . . . . . . . . . . . . 70
15.3-1. Undoing Previous Commands on the Current Line . . . . . . . . . 70
15.4-1. Fields Displayed by the "=" Command . . . . . . . . . . . . . . 71
15.6-1. Commands to Refresh Part or All of the Screen . . . . . . . . . 72
16.4-1. Commands Used to Manipulate the Mode . . . . . . . . . . . . . . 77
17-1. Commands to Perform Arithmetic . . . . . . . . . . . . . . . . . 79
18-1. Editor Option Commands . . . . . . . . . . . . . . . . . . . . . . 82
19.2-1. Commands to Invoke Modules . . . . . . . . . . . . . . . . . . . 87
25.2-1. Redirecting "edtini.edt" . . . . . . . . . . . . . . . . . . . . 120
C-1. Names of the Arrow Key Codes and Default Macros . . . . . . . . . 133

C.2-1. Color Map as Preset by Aegis . . . . . . . . . . . . . . . . . . . . . 136
C.11-1. LINDPY Special Sequences . . . . . . . . . . . . . . . . . . . . . 149

# 1. Basic Concepts

MAINEDIT is a versatile full-screen editor for general text preparation and review. MAINEDIT uses the MAINSAIL portable software technology to provide the same capabilities across a broad range of computing systems and terminals. It can be used to invoke other MAINSAIL modules without losing the context of an editing session. It is an integral part of the MAINSAIL "display executive", which provides a powerful environment for program development and execution.

MAINEDIT provides three different interfaces: MAINED, MAINVI, and MEDT. MAINED is XIDAK's own editor, and provides full access to all the capabilities of MAINEDIT. MAINVI emulates vi, the popular UNIX editor. MEDT is an emulation of the text editor "EDT" included with Digital Equipment Corporation's VAX/VMS operating system.

## 1.1. Version

This version of the "MAINEDIT User's Guide" is current as of Version 12.10 of MAINSAIL. It incorporates the "MAINEDIT Version 5.10 Release Note" of October, 1982; the "MAINEDIT Version 7.4 Release Note" and the "Display Modules Version 7.4 Release Note" of May, 1983; the "MAINEDIT Release Note, Version 8" of January, 1984; the "MAINEDIT Release Note, Version 9" of February, 1985; the "MAINEDIT Release Note, Version 10" of March, 1986; the "MAINEDIT Release Note, Version 11" of July, 1987, and the special document "MAINVI and MEDT: MAINSAIL VI and EDT Front Ends".

## 1.2. Changes to Commands

XIDAK reserves the right to create MAINEDIT commands for internal use only. Such commands are not documented here and are subject to change and/or removal without notice. MAINEDIT commands and the command sequences used to invoke MAINEDIT may change from release to release of MAINSAIL.

## 1.3. Using This Manual

This manual is divided into four parts: the MAINEDIT part, which tells how to start an editing session independent of whether MAINED, MAINVI, or MEDT is chosen, and one part each for MAINED, MAINVI, and MEDT commands.

This manual is designed to introduce MAINEDIT to a new user. Some familiarity with computers and computerized text editing is assumed. You should be comfortable with the concepts of files and directories, and know how to log on and off your computer system.

## 1.4. Notation Conventions

### 1.4.1. File Names

Some of the file name examples used in this manual may not be considered legal by the operating system of your computer. Because of the wide variation in operating systems, a comprehensive description of legal file names cannot be given here. Check with the manager of your computer system or read the user's guide for the system to find the file name syntax rules.

### 1.4.2. Special Keys

Throughout the manual, the notation <keyName> means to press the indicated key. For example, "<sp>" means press the space bar. Table 1.4.2-1 lists the key name abbreviations used in this manual, and their meanings. The actual keys to press are terminal-dependent.

Many terminals have keypads and other special keys not listed in Table 1.4.2-1. MAINED, MAINVI, and MEDT may map some of them to often-used functions such as cursor movement. See Appendix C for the mapping of such keys for your terminal.

| <key name> | Meaning |
|---|---|
| <del> | DELete (or rubout) |
| <eol> | End Of Line (marked RETURN or ENTER on most keyboards) |
| <tab> | horizontal TAB |
| <lf> | LineFeed |
| <sp> | SPace bar |
| <bs> | Back Space |
| <ecm> | Enter Command Mode |
| <abort> | ABORT command |
| <esc> | ESCape key |

Table 1.4.2-1. Key Name Abbreviations

"CTRL-" prefixing a letter means press the CONTROL (or CTRL) key while typing the letter. For example, "CTRL-C" means to hold down the CONTROL key and type "c" or "C".

### 1.4.3. Dialogues

In any illustration of a dialogue between a user and MAINEDIT, user input is underlined. Thus, in Example 1.4.3-1, MAINEDIT prompts with "File (foo.txt):". The user responds with "demo-file.txt<eol>". The "<eol>" means that input is terminated by pressing the <eol> key.

```
File (foo.txt): demo-file.txt<eol>
```

Example 1.4.3-1. Sample Prompt and User Response

### 1.4.4. Command Descriptions

In the command descriptions, a lowercase "n" indicates any nonnegative number. For example, "nK" can represent "6K" or "23K" or even "0K". Omitted "n" is equivalent to "1", unless stated otherwise. Thus, "K" is the same as "1K".

A list of options in square brackets and separated by vertical bars, such as "[C|W|L|P]", means to select exactly one of the options. For example, "D[C|W|L|P]" represents the four different MAINED commands "DC", "DW", "DL", and "DP" (Delete Character, Word, Line, and Page, respectively).

## 1.5. Files, Buffers, and Windows

MAINEDIT maintains one (or sometimes more than one) workspace called a "buffer" for each file edited during a MAINEDIT session. When a file is edited, a buffer is created and the contents of the file incrementally copied into it. The buffer name is formed from the file name by replacing every lowercase letter in the file name with the corresponding uppercase letter, so that the buffer name is entirely in upper case. The buffer may be examined and altered during the edit session, but the original file remains unaltered until its associated buffer is explicitly written. It is not necessary for a buffer to have an association with a file; however, if the contents of the buffer are to be saved at the end of a MAINEDIT session, such an association must be established.

Many MAINVI and MEDT commands work slightly differently from the corresponding vi and EDT commands because vi and EDT do not have the same type of buffer as MAINEDIT. The MAINVI and MEDT portions of this document describe the differences in detail.

The buffer name "CMDLOG" is special; see Chapter 19.

Buffers are displayed on the screen in areas called "windows". At any given moment, a buffer may be either displayed or invisible. Invisible buffers can be made visible (i.e., windows are created for them) using the MAINED commands described in Chapter 12. MAINVI and MEDT do not have complete sets of buffer and window manipulation commands of their own; however, they each have a command that allows the MAINED commands to be given, as described in Chapter 3.

## 1.6. Front Ends and Back Ends

MAINEDIT is really a software package on which different "front ends" (ways of interpreting commands) and "back ends" (ways of displaying information) may be layered. The initial default front end is "MAINED", and the initial default back end is "TXTMGR". "MAINVI" and "MEDT" are also front ends; MAINEDIT can be set up to change the default front end to one of these, or a front end can be specified when a buffer is created. The TXTMGR back end, which is for editing ordinary text files, is what is usually assumed in this manual, except in Appendix B, which describes the "DATMGR" back end. If you are using MAINEDIT in the usual fashion, you do not need to worry about different front ends and back ends after you have set up the default correctly. Occasionally during a MAINEDIT session you may see a prompt asking you for a front end or a back end; in most cases, you will want to accept the offered default.

## 1.7. Establishing the Default Front End

If you are using MAINVI or MEDT, then before you use MAINEDIT for the first time, you should create a file named "eparms" in your home directory (you can also create on your current directory instead, if you want to have more than one "eparms" file). If you are using MAINVI, the "eparms" file should contain the line:

```
FRONTEND MAINVI
```

If using MEDT, it should contain:

```
FRONTEND MEDT
```

If using MAINED, you do not need to create an "eparms" file, since in the absence of a "FRONTEND" directive, MAINEDIT chooses MAINED to be the default front end. However, you may wish to create an "eparms" anyway, as it can store much useful editor information (see Chapter 2).

## 1.8. Invoking MAINEDIT

To begin using MAINEDIT, invoke MAINSAIL and execute the module "EDIT". MAINEDIT can also be set up so that it can be invoked directly from your host operating system. If you do not know how to invoke MAINSAIL, ask your MAINSAIL representative for assistance, or consult the appropriate operating-system-specific MAINSAIL user's guide.

Invoking MAINEDIT is quite different from invoking vi or EDT. This requires some getting used to if you are familiar with vi or EDT; however, once you are in the editor environment, you will find that the MAINVI and MEDT commands are pretty familiar.

### 1.8.1. File to Edit

MAINEDIT asks for the name of the file you want to edit. This can be an existing file that you want to examine or modify, or a new file that you want to create. For example, if you want to create a new file called "foo.txt", respond to MAINEDIT as shown:

```
File to edit: foo.txt<eol>
```

MAINEDIT normally remembers the names of your most recently edited files. If MAINEDIT has this information when you begin editing, it offers you in parentheses the name of the last file that you edited. This is the "default" file. You may accept this default by typing <eol>, or type in a different name, as shown in Example 1.8.1-1.

```
To edit "old-file.txt":

    File to edit (old-file.txt): <eol>

To edit "foo.txt":

    File to edit (old-file.txt): foo.txt<eol>
```

Example 1.8.1-1. Answering the "File to Edit" Prompt

The name of the file you last edited is normally stored in a file called "eparms"; see Chapter 2. If MAINEDIT does not prompt with a default file name (either because there is no "eparms" file or the "eparms" file does not contain a default file name), you must type a file name to the "File to edit" prompt.

### 1.8.2. Display Module and Baud Rate

Since MAINEDIT can be used with many terminals, it must know what type of terminal you are using. A "display module" exists for each terminal supported by MAINEDIT. Your "eparms" file may contain the name of the display module for the terminal that you use most often. If this information is not available to MAINEDIT (i.e., is not present in the "eparms" file, or the "eparms" file does not exist), it asks you which display module to use. See Appendix C for descriptions of the display modules available as of the date of this writing.

If MAINEDIT does prompt for a display module, respond to the query with the name of the display module for your terminal, terminated with a carriage return. For example, if you are using a VT100 terminal:

```
                    Display module: vt100<eol>
```

In order to work correctly, many display terminals require extra characters ("padding") to be sent following certain terminal commands. The number of extra characters following each command depends on the speed ("baud rate") at which the terminal is operating. The baud rate you are using must be available to MAINEDIT so that the correct padding can be performed. When insufficient padding is supplied, many terminals lose characters and/or beep.

The baud rate is approximately ten times the number of characters transmitted per second to the terminal; e.g., 9600 baud is roughly 960 characters per second. The most common baud rates are 150, 300, 1200, 4800, 9600, and 19200. MAINEDIT usually assumes a default baud rate of 1200 baud (if applicable) and does not prompt for the baud rate. The default baud rate may be altered by modifying the "eparms" file as described in Section 2.6.

If you have defaults for display module or baud rate and are temporarily using a terminal or baud rate other than the default, you can override the default by typing a comma after the file name (and before the <eol>) when you respond to the "File to edit" question. This tells MAINEDIT to prompt for the display module and baud rate instead of using the defaults. Type the name of the display module (or just <eol> to use the default display module) and the baud rate (or just <eol> to accept the default baud rate). See Example 1.8.2-1.

At this point MAINEDIT clears your display screen. If your screen is not cleared, MAINEDIT is probably using the wrong display module for your terminal. If the screen is cleared but characters are subsequently lost from the display, the baud rate you have specified may be incorrect. If either problem occurs, you must exit to the operating system or the MAINEX asterisk prompt and restart the editor, this time specifying the correct display module and baud rate. If you don't know what baud rate you're using, try accepting the default; if the screen does not update properly, try again with a higher baud rate.

```
To edit "old-file.txt", using HEATH instead of VT100,
running at 9600 baud:

    File to edit (old-file.txt): ,<eol>
    Display module (vt100): heath<eol>
    Baud rate (1200): 9600<eol>

To edit "foo.txt", using HEATH instead of VT100 at the
default baud rate:

    File to edit (old-file.txt): foo.txt,<eol>
    Display module (vt100): heath<eol>
    Baud rate (0): <eol>
```

Example 1.8.2-1. Display Module and Baud Rate Prompts

### 1.8.3. New File

If MAINEDIT cannot open a file with the name that you typed, it asks whether you want a new file created with this name. Respond with "Y" (or "y"; no trailing <eol>) to create and edit a new file:

```
New file foo.txt (Yes or No)? Y
```

Type "N" (or "n") to indicate that you really did not mean to create a new file (for example, if you misspelled the file name). MAINEDIT prompts for another file name:

```
File: correct-name.txt<eol>
```

This dialogue continues until either a file is found with the name you specify, or you agree that you want a new file created.

## 1.9. MAINEDIT with Command Line Arguments

MAINEDIT can be invoked with arguments on the MAINEX, MAINEDIT, or MAINDEBUG command line according to the syntax:

```
edit {fileName {displayModule {baudRate {frontEnd {backEnd}}}}}
```

fileName specifies the initial file name. MAINEDIT prompts for the display module if none is specified on the command line or in the "eparms" file. If specified, frontEnd and backEnd override the default front end for the first buffer created; normally, MAINEDIT does not prompt for the front end and back end. MAINEDIT can be made to prompt for the front end and back end if they are not specified on the command line and fileName ends with two commas.

For example, the following command line causes MAINEDIT to prompt for the display module and baud rate and also the front and back end:

```
edit foo,,
```

The following example prompts for the front and back end:

```
edit foo,, bigsun 0
```

The following example does not prompt for anything, and accepts the default front and back ends:

```
edit foo bigsun 0
```

The following example uses MAINVI as the initial front end and prompts for the back end:

```
edit foo,, bigsun 0 mainvi
```

The following example uses MAINVI for the initial front end and TXTMGR for the initial back end and does not prompt for anything:

```
edit foo bigsun 0 mainvi txtmgr
```

## 1.10. Screen Format

MAINEDIT now fills the display screen with text from the file you are editing. If the file is new, the effect is the same as if the file contained a single blank line. See Figure 1.10-1, which shows a default MAINED screen format (MAINVI and MEDT formats are slightly different, and do not have the colon borders).

### 1.10.1. Message Line

The message line, which is at the top of the screen (which is not where vi and EDT put it, but is where MAINVI and MEDT put it) is used for error and warning messages, prompting, and user responses. When reading long lines from the message row, MAINEDIT clears the line and redisplays the last ten characters (left-justified) before echoing the next input character if the

```
message line->
status line--> ---P.2---L.3----C----FOO.TXT--------------
               :This is some of the text contained in the:
               :file "foo.txt". The first line of the    :
               :screen happens to be line three of page  :
               :two of the file.                         :
page mark----> PAGE 3 -----                        --- PAGE
               :At this point, page 3 starts.            :
               :                                         :
               :This is the last line of "foo.txt".      :
buffer end---> E
               E
               E
```

Figure 1.10-1. MAINEDIT Screen Format

echoed character would have been beyond the end of the message row. Control characters (and other non-printing characters) echoed on the message line appear as asterisks (the "*" character).

## 1.10.2. Status Line

The status line shows information about the current window. There is one status line per window. Initially there is only one window on the screen; multiple windows and the MAINED commands for manipulating them are described in Chapter 11.

The information displayed in the status line for MAINED includes the page and line number of the top line of the window, the mode, and the buffer name. "P.2 L.89", for example, means that the first line in the window is the eighty-ninth line of the second page of the buffer (pages and lines are numbered starting with one). The next character ("C", "E", "I", or "O") indicates the current mode (Command, Escape, Insert, or Overstrike). Upon entry, command mode is in effect. Modes are explained in Section 4.3.

Status lines for front ends other than MAINED may be slightly different, and are described in the front-end-specific portions of this document.

When you first invoke the editor, the buffer name displayed is the name as the file you specified upon entering the editor converted to upper case.

### 1.10.3. Page Marks

The line in Figure 1.10-1 that reads:

```
            PAGE 3 -----                                 --- PAGE
```

is called a page mark.

A page is a body of text bounded by end-of-page characters or by the beginning or end of the file. The end-of-page character is system-dependent (it is the MAINSAIL "eop" character). On most ASCII systems this character is a CTRL-L (or formfeed). An end-of-page character, when printed by a line printer, usually causes the paper to be advanced to a new page. When displayed by MAINEDIT, it appears as a page mark.

### 1.10.4. Other Special Characters

Non-displayable and other special characters (such as control or escape characters) that occur in a file are displayed on most terminals as an asterisk ("*"). The actual value of such a character can be determined by positioning the cursor at the character and issuing the "=" command. See Section 15.4.

### 1.10.5. Cursor

The cursor indicates where editing takes place. The form of the cursor depends on your terminal; it is commonly a solid or blinking underbar or box. When you wish to insert or delete text at a particular place, you must (for most commands in most front ends) move the cursor to that point.

## 1.11. Tabs

When MAINEDIT reads from a file into a buffer, it pads each tab character with enough additional tabs to fill out to the next tab stop. This allows MAINEDIT to keep one character in its internal data structure for each character position on the screen. For the purposes of reading and writing files, a tab stop is defined at every eighth column. During the edit session, a tab in a buffer is treated as if it were a space (for exceptions, see Sections 6.6 and 18.1).

Whenever a buffer is written to a file, each sequence of tabs that extends to a tab stop is replaced with a single tab. The net result is that tabs that have not been moved or altered during the edit session are preserved in the output file.

## 1.12. Finishing the Edit Session

When you are ready to end your edit session, type the appropriate exit command for the front end (e.g., "QF" in MAINED, ":wq" in MAINVI, "quit" in MEDT). If you have made unsaved changes to any buffer, MAINEDIT prompts in the message line to ask if you want the buffer saved:

```
                    Write buffer FOO.BAR (Yes No)?
```

If you type "Y" or "y" in response, MAINEDIT writes the buffer into a file. If you type "N" or "n", MAINEDIT does not write the buffer. This means that none of your changes since you last saved the buffer is recorded, so be careful that this is what you really want.

The name of the output file is normally the same as the name of the input file (you can change the name to be used for the output file by using the MAINED "Q.F" command described in Chapter 12). If you are editing a file that has version numbers, a new version is created for the output file. Otherwise, you are prompted whether to delete the existing file, so that the new file can replace it. If you say no ("N"), then you are prompted for a new file name to be used for the output file.

Before returning control to the operating system or MAINEX prompt, MAINEDIT blanks the screen.

# 2. The "eparms" File

When MAINEDIT is invoked, it tries to open a file named "eparms" (the name must be lowercase on case-sensitive file systems). If it succeeds, it uses this file to access and store information about the environment in which you normally use MAINEDIT. If it fails, it looks for a file named "eparms" on your home directory (as returned by the MAINSAIL system procedure $homeDirectory). Each line in "eparms" starts with a keyword.

If an "eparms" file does not exist at the start of the MAINEDIT session, MAINEDIT does not create a new one. Users typically create an "eparms" file just once, since "eparms" is automatically updated by MAINEDIT thereafter.

## 2.1. "EDITORPARMSFILE"

The "EDITORPARMSFILE" keyword is used to specify the file name to be used when the "eparms" file is updated. It is usually satisfactory to allow the "eparms" on your current or home directory to be updated, so this keyword is rarely needed.

## 2.2. "CONTEXT"

When "eparms" is updated, MAINEDIT records the final page and line of the cursor position for each of the files you edited during the current session as the "context" for that file. Whenever you edit a file, MAINEDIT checks "eparms" to see if context information is recorded for the file. If so, it offers the saved page and line as a starting point (on the front ends that support this option), since it is common to want to continue editing from the most recently edited point.

MAINEDIT maintains context for the thirty most recently edited files. Sample "CONTEXT" entries are:

```
CONTEXT mainedit-manual.txt 3 55
CONTEXT w:editor-notes.txt 1 7
```

There is no need to change these entries, since they are maintained by MAINEDIT.

## 2.3. "DISPLAYMODULE"

To display text properly, MAINEDIT must know the characteristics of your terminal. For each terminal MAINEDIT supports, there is a "display module" that handles the specifics of the terminal interaction.

The "DISPLAYMODULE" entry in your "eparms" file supplies MAINEDIT with a default display module. For example:

```
DISPLAYMODULE vt100
```

means that the default is the VT100 display module.

If the "DISPLAYMODULE" keyword is present, MAINEDIT uses the specified display module without prompting for one, unless a comma is typed after the file name specified to the initial "File to edit:" prompt. If this keyword is not present, MAINEDIT always prompts for the display module name.

## 2.4. "UPDATEDISPLAYMODULE"

If this keyword is present, the "DISPLAYMODULE" entry is updated to be the display module in use whenever "eparms" is rewritten. Otherwise, the "DISPLAYMODULE" entry (if any) is unchanged.

For example, if, when MAINEDIT is invoked, the "eparms" file contains:

```
DISPLAYMODULE vt100
UPDATEDISPLAYMODULE
```

and you override the default (VT100) display module because you are using a Heath terminal and need the HEATH display module, then when MAINEDIT updates "eparms", it changes the "DISPLAYMODULE" line to be:

```
DISPLAYMODULE heath
```

so that the next time MAINEDIT is invoked, the default display module is HEATH.

## 2.5. "DONOTPROMPTFORDISPLAYMODULE"

Certain XIDAK utilites allow you to switch from line-oriented to display mode (see, for example, the description of the "@" command in the "MAINDEBUG User's Guide"). When the switch occurs, MAINSAIL needs to know what display module to use. Normally, when such a switch occurs, you are prompted for the name of the display module regardless of whether a "DISPLAYMODULE" entry appears in your "eparms" file. If "eparms" contains both the "DISPLAYMODULE" and "DONOTPROMPTFORDISPLAYMODULE" keywords, however, then when MAINSAIL switches to display mode, it uses the default display module without prompting.

The "DONOTPROMPTFORDISPLAYMODULE" keyword is ignored at the initial "File to edit:" prompt; that is, if a comma terminates the initial file name, you are prompted for the display module name whether or not the "DONOTPROMPTFORDISPLAYMODULE" keyword is present in "eparms".

If both the "DISPLAYMODULE" and "DONOTPROMPTFORDISPLAYMODULE" keywords are present in "eparms", it is not possible to override the default display module when switching to display mode from a program.

## 2.6. "BAUDRATE", "UPDATEBAUDRATE", and "DONOTPROMPTFORBAUDRATE"

These keywords can be used to specify how MAINEDIT determines the line speed of your terminal. On some terminals, padding is required to perform certain terminal functions properly, and that padding is dependent on the baud rate of the terminal line. If MAINEDIT appears to be losing characters or otherwise improperly updating the screen when it is first invoked, it is possible that you do not have the baud rate set correctly.

The "BAUDRATE", "UPDATEBAUDRATE", and "DONOTPROMPTFORBAUDRATE" keywords function analagously to the "DISPLAYMODULE", "UPDATEDISPLAYMODULE", and "DONOTPROMPTFORDISPLAYMODULE" keywords, except that there is always a default baud rate (1200 baud, if applicable). Therefore, MAINEDIT does not prompt for the baud rate after the initial "File to edit:" prompt unless a comma terminates the initial file name, even if there is no "BAUDRATE" keyword present in the "eparms" file.

## 2.7. "MACRO" and "NAMEDMACRO"

MAINEDIT stores editor macro definitions using these keywords. A macro is a user-defined sequence of commands. Macros are defined and invoked with different commands under the different MAINEDIT front ends. When you define a macro from the keyboard, MAINEDIT stores the definition in "eparms" when "eparms" is updated. Thus, a permanent library of macro definitions is maintained across edit sessions. The type of the terminal on which each macro was originally defined and the front end for which it was defined are stored with the macro, since the codes transmitted by different keys make different macros appropriate on different terminals, and a macro appropriate to one front end is probably not appropriate to another.

An example of a "MACRO" entry is:

```
MACRO(HEATH,MAINED) 30     <27> j q > . ( b
```

On an ASCII system using a Heath terminal this means that the character with code 30 (CTRL-^) is defined to be the sequence "<esc> J Q > . ( B". Blanks are ignored in the command sequence following the "MACRO" keyword.

An example of a "NAMEDMACRO" entry is:

```
NAMEDMACRO(DATAMEDIA,MAINED) FOO
<13> i F O O <27> <27>
```

The named macro "FOO" is defined for a Datamedia terminal as "<eol> i F O O <esc> <esc>". "MACRO" and "NAMEDMACRO" entries are maintained by MAINEDIT, so there is no need for you to understand or modify them.

## 2.8. "WINDOWWIDTH"

The "WINDOWWIDTH" parameter specifies the default location of the right margin whenever a new window is created. For example,

```
WINDOWWIDTH 72
```

gives you 72 columns in which to enter text. If this entry is not present, MAINEDIT sets margins to the maximum width for the terminal you are using; see the information for your display module in Appendix C.

You may also change the window width of the current buffer at any time with the MAINED "QnX" command, as described in Chapter 11; some other front ends may also provide commands that affect the window width. This does not affect the value stored in "eparms"; you must explicitly modify "eparms" to change the default window width.

## 2.9. "DONOTUPDATEEPARMS"

This keyword tells MAINEDIT not to update the "eparms" file. It is useful in the event that the editor is embedded in an application for which a fixed set of "eparms" parameters is desired.

If this keyword is present, your contexts and macros cannot be updated.

## 2.10. Changing an "eparms" File

The "eparms" file is a convenience to help record information that otherwise must be entered each time you use MAINEDIT. It is not necessary to have an "eparms" file, but you may well find it more convenient to have one. MAINEDIT does not automatically create "eparms", so you must first do so yourself, either by copying one from someone else, or using MAINEDIT to create a file called "eparms". To edit an existing "eparms", change the file name associated with the buffer that contains your edited "eparms" (in MAINED, use the "Q.F" command) before exiting MAINEDIT; otherwise, MAINEDIT overwrites the edited version of "eparms" when it exits. Then rename your edited file to be "eparms"; it is used the next time MAINEDIT is invoked. It is most convenient to put "eparms" on your home directory, so that MAINEDIT finds it regardless of the directory to which you are connected when you invoke MAINEDIT.

An example of an initial "eparms" file is shown in Example 2.10-1. The file automatically acquires additional commands (e.g., "CONTEXT" and "MACRO" commands) as it is updated by MAINEDIT.

```
DISPLAYMODULE vt100
WINDOWWIDTH 72
```

Example 2.10-1. An Initial "eparms" File

## 2.11. "FRONTEND" and "BACKEND"

The "FRONTEND" keyword specifies a default front end; if absent, MAINED is used. The "BACKEND" keyword specifies a default back end; if absent, TXTMGR (the usual back end) is used.

# 3. Switching among Front Ends

It is often convenient to give MAINED commands from MAINVI or MEDT, since MAINED supplies some concepts not found in the original vi or EDT, and MAINVI and MEDT have no commands for dealing with such concepts.

A buffer generally has one front end that displays the text of the buffer (this is called the "view" front end) and front end that processes the user's commands (the "command" front end). These two front ends are usually the same but may be different. The view front end governs the "look" of the editor; for example, MAINED usually indicates a left and right border and page marks are displayed on a separate line with the page number, while MAINVI usually has no borders and page marks are displayed on a separate line as an asterisk. The command front end interprets keystrokes and determines how they affect the edited text; it is the command front commands that are described in the documentation for each front end.

Both MAINVI and MEDT provide commands for switching the command or view front end to another front end, either for the duration of a single series of commands or until another command is given to switch front ends. A front end may also be killed if it is not currently in use and will not be subsequently used. These commands are mentioned elsewhere in this manual, but are repeated here for reference:

(Note: MEDT commands are line mode commands)

| Command | MAINVI | MEDT | MAINED |
|---|---|---|---|
| Switch to command front end s until another command is given to switch command front end; if cmds present, execute, then switch back to current command front end | :swf s {cmds} | SWF s {cmds} | Q..Bs {cmds} |
| Switch both command and view front ends to s until another command is given to switch; if cmds present, execute, then switch back to current command and view front ends | :swfv s {cmds} | SWFV s {cmds} | +Q..Bs {cmds} |
| Switch to MAINED command and view front ends; if cmds present, execute, then switch back to current command and view front ends | :swm {cmds} | SWM {cmds} | not applicable |
| Switch view front end to s until another command is given to switch view front end | :swv s | SWV s | Q..Vs |
| Kill front end s | :killf s | KILLF s | -Q..Bs |
| Show buffer status information including front end information | not provided | not provided | +Q= |

# MAINED User's Guide

# 4. MAINED Basics

MAINED was the first front end written for MAINEDIT and has the most complete set of commands for manipulating MAINEDIT's special facilities (such as buffers, windows, and executing modules). For this reason, users of some of the other front ends should familiarize themselves somewhat with this description of MAINED.

## 4.1. Notation Conventions

### 4.1.1. Case

MAINED does not distinguish between upper and lower case in command characters. Though most commands in this manual are shown in upper case, you may type them in upper or lower case.

## 4.2. Starting and Ending a MAINED Session

### 4.2.1. Initial Page and Line

If you are editing an existing file, MAINED asks for the page and line number at which to start editing. If you are editing a new file, you must start at the first line of the first page, so MAINED does not ask for the initial page and line.

In the "Initial page.line" prompt, the numbers in parentheses indicate the page and line number that MAINED uses by default. This is the point last edited in the file, as recorded in "eparms". Thus, if you last edited the file at page 3, line 45, the prompt offers "(3.45)":

```
            Initial page.line for old-file.txt (3.45):
```

Typing <eol> at this point accepts the offered default value. If you wish to start at another point you can type it in:

```
            Initial page.line for foo.txt (6.8): 2.3<eol>
```

The above response tells MAINED to place the cursor at the third line of the second page of the file "foo.txt". This can save time if you have a particular destination in mind. A response of "n" is equivalent to "n.1"; ".n" is equivalent to "1.n". A line number greater than the number of the last line on the specified page selects the last line on the page; a page number greater than the number of the last page in the file selects the last line in the file.

### 4.2.2. Finishing the Edit Session

When you are ready to end your edit session, type "QF". If you have made changes to the buffer, MAINED prompts in the message line to ask if you want the buffer saved:

```
Write buffer FOO.BAR (Yes No)?
```

If you type "Y" or "y" in response, MAINED writes the buffer into a file. If you type "N" or "n", MAINED does not write the buffer. This means that none of your changes since you last saved the buffer is recorded, so be careful that this is what you really want.

The name of the output file is normally the same as the name of the input file (you can change the name to be used for the output file by using the "Q.F" command described in Chapter 12). If your operating system supports file versions, a new version is created for the output file. Otherwise, you are prompted whether to delete the existing file, so that the new file can replace it. If you say no ("N"), then you are prompted for a new file name to be used for the output file.

Before returning control to the operating system or MAINEX prompt after a "QF" command, MAINED blanks the screen.

## 4.3. Modes

In MAINED your keystrokes either:

1. Form command characters or answer prompts. This mode is called "command mode".

2. Are inserted into the buffer. This is "insert mode".

3. Are overstruck into the buffer. This is "overstrike mode".

4. Are interpreted by a MAINSAIL module other than the editor. This is "escape mode".

### 4.3.1. Command Mode

When MAINED is first invoked, command mode is in effect. Any input is interpreted as commands, except for responses to prompts. The status line at the top of the buffer indicates command mode with a "C".

### 4.3.2. Insert Mode

The command "I" (typed while in command mode) puts MAINED into insert mode. The "I" command changes the letter in the status line to "I". In insert mode, characters are inserted into the text in the buffer at the cursor. Any characters to the right of the cursor on the current line are moved to the right. Typing <eol> in insert mode terminates the current line and begins a new one below; any text to the right of the cursor is moved to the beginning of the new line.

### 4.3.3. Overstrike Mode

The command "O" (typed while in command mode) puts MAINED into overstrike mode. The "O" command changes the letter in the status line to "O". In overstrike mode, characters at the cursor are overwritten (replaced). Typing <eol> in overstrike mode moves the cursor to the beginning of the next line without altering the existing line.

### 4.3.4. Escape Mode

In escape mode, keystrokes are intercepted and interpreted by some program other than the editor. The effect of various keys depends on the program. Refer to Chapter 19 for more information on escape mode.

### 4.3.5. Returning to Command Mode

A special non-printing key is needed to leave insert or overstrike mode. This key, called <ecm> (Enter Command Mode), puts MAINED into command mode from either insert or overstrike mode. <ecm> typed while in command mode aborts the current command (if in the midst of entering a multi-character command) and remains in command mode. <ecm> typed in escape mode is interpreted by the program that is intercepting keystrokes, and may or may not enter command mode.

The actual key to press for <ecm> is terminal-dependent. See the description of your display module in Appendix C to determine the <ecm> key for your terminal.

## 4.4. Basic Editing Commands

The following sections briefly describe a few commands to allow you to move the cursor, scroll the window, and delete objects. Together with insert and overstrike mode, these are enough to allow simple editing.

### 4.4.1. Moving the Cursor

Six commands for moving the cursor are shown in Table 4.4.1-1. These commands can be modified by first typing the number of columns or words by which you want to move. Thus, "5<" moves the cursor left by 5 columns.

| Command | Move Cursor: |
|---|---|
| < | left |
| > | right |
| ^ | up |
| \ | down |
| ( | left one word |
| ) | right one word |

Table 4.4.1-1. Basic Cursor Movement Commands

The commands shown in Table 4.4.1-1 work for any terminal. On many terminals, there are keys marked with arrows pointing left, right, up, and down. These keys are mapped by default to perform the commands ">", "<", "^", and "\", respectively. Refer to the keypad mappings for supported terminals given in Appendix C.

### 4.4.2. Windowing

The text in the current window can be scrolled up or down with the commands shown in Table 4.4.2-1.

| Command | Scroll: |
|---|---|
| W | up 4/5 of a screen |
| -W | down 4/5 of a screen |
| nW | up n lines (n must be present) |
| -nW | down n lines (n must be present) |

Table 4.4.2-1. Basic Scrolling Commands

### 4.4.3. Deleting

Text can be deleted by means of the commands shown in Table 4.4.3-1. Any of these commands can be modified with a count to indicate how many objects to delete. For example, "5K" deletes five characters.

| Command | Delete Current: |
|---|---|
| K or DC | character |
| DW | word |
| DL | line |
| DP | page |

Table 4.4.3-1. Basic Commands for Deleting Text

## 4.5. Current Character, Word, Line, and Page

Most MAINED commands deal with the "current" character, word, line, or page.

The current character is the one at which the cursor is positioned. If the cursor is beyond the end of the line, then there is no current character.

The current word is the word that contains the current character. A word is any sequence of visible characters, separated from other words by blank space (i.e., space or tab characters) or by the end of a line. If the cursor is not under a visible character, then the current word is the blank space from the cursor up to the next word to the right, if there is a word to the right of the cursor; otherwise there is no current word.

The current line is the one that contains the cursor. The cursor can be anywhere on the line.

The current page is the one that contains the cursor. The cursor can be anywhere on the page, including the initial page mark (which is considered to be line zero of the page).

## 4.6. MAINED Window Appearance

MAINED windows by default show special characters (colons, asterisks, and E's) in the margins. This display is suppressed by default in some of the other front ends, although the it can be disabled in MAINED or reenabled with special commands in the other front ends.

### 4.6.1. Margins

The columns of colons on the left and right sides of the screen mark the left and right margins, respectively. There is no limit to the length of a line of text, though you can see at any one time only what can be displayed on your screen. If a line of text extends beyond a margin, an asterisk ("*") is used in place of the colon as a margin marker. See Example 4.6.1-1.

```
:This line contains text beyond the right margin, b*
:The text is not visible on the screen, but is     :
:"there" nevertheless.                             :
```

Example 4.6.1-1. Text beyond the Margin

The display of the colons may be disabled, as described in Chapter 18.

### 4.6.2. Buffer End

The E's at the lower left margin indicate "end-of-buffer". That is, there is no more text beyond the last colon printed. In this case the right margin is blank.

## 4.7. Pages and Page Marks

Type "P" to insert an end-of-page character immediately ahead of the current line. MAINED indicates the end-of-page on your screen with a "page mark", which shows the number of the page. The first page in a buffer is number one, and it starts with the beginning of the file instead of a page mark. The last page is terminated by the end of the file instead of a page mark. See Figure 1.10-1.

To delete a page mark, place the cursor anywhere on the line containing the page mark, and type "DL" (Delete Line). Insertion or deletion of a page mark automatically updates page numbers in any page marks visible on the screen. The "G" (Go) command moves the cursor to a specified page and line.

## 4.8. Buffer Names

Buffer names may be abbreviated in most commands that prompt for buffer names (the exceptions are the ".K" command and its variants, which require the full name of the buffer to be killed). A buffer name abbreviation is any substring of a buffer name.

If the string typed in response to a prompt for a buffer matches the name of some buffer exactly, that buffer is selected. If the string typed is a substring of exactly one buffer name, that buffer is selected. If the string typed is a substring of more than one buffer name, one of the buffers is selected (no guarantee is given on which one).

To force the creation of a new buffer with a name that is the substring of an existing buffer name, first create a buffer the name of which is not a substring of any existing buffer name, then rename the buffer with the "Q.B" command.

## 4.9. MAINED Special Keys

The keys marked with an asterisk in Table 4.9-1 are user-redefinable as macro ID's. In all command descriptions in this manual, the keys are assumed not to have been redefined; commands using these keys behave differently if they have been redefined. See Chapter 16 for an explanation of how to redefine these keys.

| | <key name> | Meaning |
|---|---|---|
| | `<del>` | DELete (or rubout) |
| * | `<eol>` | End Of Line (marked RETURN or ENTER on most keyboards) |
| * | `<tab>` | horizontal TAB |
| * | `<lf>` | LineFeed |
| | `<sp>` | SPace bar |
| * | `<bs>` | Back Space |
| | `<ecm>` | Enter Command Mode |
| | `<abort>` | ABORT command |
| | `<esc>` | ESCape key |

Table 4.9-1. Key Name Abbreviations

The remainder of the MAINED portion of this manual documents all MAINED commands, and is required reading for full use of MAINED's features. All the MAINED commands and their uses, as well as the handling of buffers, files, and windows, are discussed in detail.

Appendix A is a command summary for quick reference; the commands are summarized there in alphabetic order.

# 5. Command Syntax

The basic syntax of MAINED commands is:

```
          <modifiers>   <basic command>    <object>
```

The three types of modifiers are direction, count, and emphasis. They can be specified in any order, as long as they precede <basic command>. The object must follow <basic command>. Not all basic commands need or allow the use of modifiers or objects.

## 5.1. Direction ("+" or "-")

For many commands, "+" (plus) indicates "toward the end of the buffer", and "-" (minus) indicates "toward the beginning of the buffer". "+" is the default in such commands.

Some commands, such as ".B", "F", "X", and "Y", use the "+" or "-" modifiers in their own special ways. See the descriptions of those commands for details. In some commands, both "+" and "-" may be used.

## 5.2. Count (n)

This modifier (where n is a nonnegative integer) is used in two similar ways. As a count, it indicates the number of times a command is to be repeated. As a quantifier, it specifies an exact target. See Example 5.2-1.

| Command | Action |
|---|---|
| < | Move the cursor left one column |
| n< | Move the cursor left n columns |
| G | Go to next page |
| nG | Go to page n |

Example 5.2-1. Examples of the Use of Counts

The effects of quantifiers are included in the description of each command.

## 5.3. Emphasis ("Q")

The "Q" modifier generally indicates emphasis, or "do the following command in a big way". Thus, it often plays the role of an infinite count, though it can also indicate a special case of the command. For example, with the "D" (Delete) command:

```
"3DW"     Delete three words
"QDW"     Delete all remaining words on the current line
```

Command descriptions often describe the "Q" modifier as giving a command an effect over "all characters", "all words", or "all lines". Such character and word ranges are limited to the current line, and such line ranges are limited to the current page (although they may be partially outside the current window).

## 5.4. Basic Command

The basic command is the verb part of the command string; it specifies the action to be taken. The basic command consists usually of a single character, or sometimes one or more dots followed by a character. For example, "K" means "Kill a character", whereas ".K" means "Kill buffers". The "." has no consistent meaning across commands.

It is important to remember that the basic command separates the modifiers from the object. If a basic command consists of more than one character, the characters must be contiguous; i.e., they may not be separated by modifiers. For example, ".3C" is not the same as "3.C".

## 5.5. Object

Many basic commands can act on either characters, words, lines, or pages, and thus require that the target object be specified. The summaries for these commands give a list of possible objects, such as "[C|W|L|P]". This means to follow the basic command with exactly one of the options inside the square brackets.

For example, when using the "C" command, type "CC" to copy characters, "CW" to copy words, "CL" to copy lines or "CP" to copy pages.

# 6. Cursor Movement and Windowing

## 6.1. Basic Cursor Movement

Since all text insertion and modification takes place at the cursor location, it is important to be able to move the cursor quickly through your buffer. The commands in Table 6.1-1 are used for simple cursor positioning.

## 6.2. Go to Specified Page and Line

The variations of the "G" (Go) command are used to go to a specified page and line. In Table 6.2-1, "p" and "l" are nonnegative integers. The "G" command may be aborted with the <abort> key.

| Command | Go to: |
|---|---|
| G | first line of next page |
| -G | first line of previous page |
| p.lG | page p, line l |
| .G | first line of current page |
| .lG | line l of current page |
| pG | first line of page p |
| +nG | first line of current page + n |
| -nG | first line of current page - n |

Table 6.2-1. Page- and Line-Relative Motion Commands

If the target page does not exist, the cursor is moved to the last line of the buffer. If the target line does not exist, the cursor is moved to the last line of the target page.

Use the emphasis modifier ("Q") with the "G" command to set a mark at the current position before going to the new position. For example, "Q5G" sets the mark to the current position and then goes to page 5. This is equivalent to typing ".@" before the "G" command; see Section 6.5.

| Command | Move cursor: |
|---|---|
| n<, n<bs>, n<del> | left n columns |
| Q<, Q<bs>, Q<del> | to left margin |
| n>, n<sp> | right n columns |
| Q>, Q<sp> | to right margin |
| n( | left n words |
| Q( | to first visible character on current line |
| n.( | to space after nth previous word |
| n) | right n words |
| Q) | to space after last character on current line |
| n.) | to space after nth next word |
| n<tab> | to nth next tab stop |
| -n<tab> | to nth previous tab stop |
| n^ | up n rows |
| Q^ | to first row of window |
| n\, n<lf> | down n rows |
| Q\, Q<lf> | to last row of window |
| n<eol> | to left margin of nth next line |
| -n<eol> | to left margin of nth previous line |
| nX | to column n from left margin |
| -nX | to window width - n + 1 |
| nY | n rows from top of window |
| -nY | n rows from bottom of window |
| n.Y | to nth next window on screen |
| -n.Y | to nth previous window on screen |
| Q.Y | to bottommost window on screen |
| -Q.Y | to topmost window on screen |

Table 6.1-1. Basic Cursor Positioning Commands

## 6.3. Go to Specified Character Position

The "V" command is used to go to a specified character position in the current buffer. Character position "n" is the location of the nth character in the buffer, counting from 1. In response to the "V" command, MAINED prompts for the target character position, giving the current position as the default target position. To stay at the current position type just <eol> in response to the prompt. See Example 6.3-1. The "V" command may be aborted with the <abort> key.

```
Position (27832): <eol>          (stay at current position)

Position (27832): 12345<eol>     (go to position 12345)
```

Example 6.3-1. Use of the "V" Command

## 6.4. Skip to Character or Line

The Skip command moves the cursor either to an occurrence of a specified character on the current line, or to an occurrence of a line (in the current window) of which the first visible character is the specified character. In Table 6.4-1, x represents any character.

If the "+" modifier is specified, the skip command skips to the first character (or line) that is not (or does not begin with) the specified character.

| Command | Skip to: |
|---|---|
| nSx | nth next occurrence of x on current line |
| -nSx | nth previous occurrence of x on current line |
| +Sx | next occurrence of non-x on current line |
| -+Sx | previous occurrence of non-x on current line |
| QnSx | nth next line with x as first visible character (x may not be <sp>) |
| -QnSx | nth previous line with x as first visible character (x may not be <sp>) |
| +QSx | next line without x as first character |
| -+QSx | previous line without x as first character |
| QnS<sp> | nth next all-blank line |
| -QnS<sp> | nth previous all-blank line |

Table 6.4-1. Commands to Skip to a Character or Line

If the target is not found, the cursor is moved to the beginning (for "-") or end of the line for the non-"Q" forms, and the beginning (for "-") or end of the window for the "Q" forms.

## 6.5. Set and Jump to Mark

Each buffer contains a single "mark" that acts as a place holder. Use of the mark allows you to move the cursor temporarily to another point in the buffer, then return quickly to the original location. The commands are shown in Table 6.5-1.

| Command | Action: |
|---|---|
| .@ | Set the mark to the current location |
| @ | Jump to the marked location |
| Q@ | Set the mark to the current location, then jump to previously marked location |

Table 6.5-1. Commands to Control and Use the Mark

"Q@" makes it easy to bounce back and forth between two places in the buffer. The "QG" command (see Section 6.2) sets the mark to the current location before going to the target page and line.

The mark remembers the page number, the relative line number on the page, and the relative character number on the line. If text prior to the marked location is changed, then jumping to a mark may not position the cursor at exactly the same point in the text as the one at which the mark was set.

## 6.6. Text Search

The "T" command is the text search command. It prompts for the search string, then moves the cursor to the first occurrence of the search string, if found. Table 6.6-1 lists the basic search commands. In the table, "s" represents the search string that you type in response to the prompt.

| Command | Search right and: |
|---|---|
| Ts<eol> | all lines down for s |
| nTs<eol> | n-1 lines down for s (n required) |
| .Ts<eol> | all lines down for identifier s |
| n.Ts<eol> | n-1 lines down for identifier s (n required) |

Table 6.6-1. Text Search Commands

The "T" command searches for any sequence of characters that matches the target string, even if it occurs in the midst of a word. ".T", however, searches for an "identifier"; i.e., the target cannot be bordered by an alphanumeric character. For example, target string "at" specified to the "T" command is found in "match", "Attach", "watch", or any other string containing the consecutive letters "at" in any upper-lower case mixture. The ".T" command, on the other hand, does not find the target "at" in any of these words, but does find it in ". At", "at;", or "AT?". Tabs in the search string match target tabs, while blanks in the search string match both tabs and blanks.

Use the "-" modifier to search left and towards the beginning of the file. Use the "Q" modifier to specify multiple target strings, separating the targets with <eol>. Terminate the last target with an extra <eol>. The search then finds the first occurrence of any of the target strings.

Use the "+" modifier to wrap around the beginning or end of buffer during the search. The "+" and "-" qualifiers can be used in the same search command. The "+" modifier is useful when you want to find the target string anywhere in the file regardless of the current location. If this modifier is not used, the search is terminated by the beginning or end of the buffer.

The "QQ" qualifier causes each target line to be displayed on the message row, preceded by a page and line number. This is called a "line search". MAINED waits for a one-character response:

- <eol>: go to the indicated target line and terminate the search.
- A: search for the next match.
- Q: abort the search, but do not terminate any macro being defined or invoked.
- <abort>: abort the search and any macros being defined or invoked.
- 0 (i.e., the digit "zero"): redisplay the message row
- <digit in the range 1 through 9>: display the digit-th segment of the target line (so that wide lines can be examined). Each segment is 4/5 the width of the screen.

Upper and lower case distinctions are ignored in the search and target strings. If MAINED does not find some target "FOO", the cursor is not moved, and MAINED beeps and gives the message:

```
                    Did not find "FOO"
```

The prompt for the search string gives as the default the string(s) last searched for. Type just <eol> to use the default:

```
     Search string ("FOO"): <eol>     (search for "FOO")
```

## 6.7. Scroll Window

Scrolling the text in the window to show portions of the text that are not currently visible is called "windowing". The variations of the "W" (Window) command allow you to scroll the text in the current window in either direction, and by varying amounts. See Table 6.7-1.

| Command | Scroll: |
|---|---|
| W | up 4/5 of a window |
| -W | down 4/5 of a window |
| nW | up n lines (n required) |
| QW | up all lines of the window |
| -nW | down n lines (n required) |
| -QW | down all lines of the window |
| n.W | current line to nth line from top of window |
| -n.W | current line to nth line from bottom of window |

Table 6.7-1. Commands for Scrolling Windows

# 7. Overstriking

## 7.1. Overstrike Mode

When MAINED is in overstrike mode, text input from the keyboard is written over the text currently in the buffer. With each keystroke, the character under the cursor is replaced with the new character typed, and the cursor is moved one column to the right. If a character is overstruck beyond the last character on the current line, tab characters are inserted out to the current character position.

To enter overstrike mode from command mode, type "O". To return from overstrike mode to command mode, type <ecm>.

## 7.2. Special Keys in Overstrike Mode

The commands shown in Table 7.2-1 can be used in overstrike mode to move the cursor; the <tab> key also replaces the text it moves over with tab characters. All of these commands leave you in overstrike mode.

| Command | Action |
|---|---|
| <bs>, <del> | Move cursor left 1 column, except if at left margin, move to end of previous line |
| <tab> | Overstrike tabs to next tab stop |
| <lf> | Move cursor down 1 row |
| <eol> | Move cursor to left margin of next line |

Table 7.2-1. Effect of Special Keys in Overstrike Mode

## 7.3. Overstriking a Sequence of Characters

When modified by a count or emphasis, the "O" command overstrikes a string of characters with a specified character. In Table 7.3-1, "x" is any character.

| Command | Overstrike: |
|---|---|
| nOCx | n x's (n required) |
| QOCx | x's to right margin |

Table 7.3-1. Commands to Overstrike Text

After each command in Table 7.3-1 is executed, the mode remains command mode and the cursor is positioned one column to the right of the last overstruck character. For example, to overstrike eight characters at and to the right of the cursor with the character "-", type "8OC-".

# 8. Inserting

## 8.1. Insert Mode

When MAINED is in insert mode, text input from the keyboard is inserted into the buffer. With each keystroke, the character typed is inserted at the current cursor position and the cursor is moved one column to the right. Any characters to the right are shifted one column to the right to make room for the new text. If a character is inserted beyond the last character on the current line, tab characters are inserted out to the current character position.

Table 8.1-1 shows how to enter insert mode from command mode. Type <ecm> to return from insert mode to command mode.

| Command | Action |
|---|---|
| `I` | `enter insert mode` |
| `.I` | `insert a line above current line, enter insert mode` |
| `n.I` | `same as .I, except indent new line n columns` |

Table 8.1-1. Commands to Enter Insert Mode

## 8.2. Special Keys in Insert Mode

The commands in Table 8.2-1 have the effects shown. All of these commands leave you in insert mode.

## 8.3. Inserting Lines or Characters

When modified by a count or emphasis, the "I" command inserts new lines, or inserts a character a specified number of times. These commands do not leave you in insert mode. In Table 8.3-1, "x" is any character.

For example, to insert eight blanks at the cursor, type "8IC<sp>".

| Command | Action |
|---|---|
| `<bs>` | Move cursor left 1 column |
| `<del>` | Delete the character to the left of the cursor, then move cursor left 1 column, except if at left margin, join current line to end of previous line (like "-QJ" command) |
| `<tab>` | Insert tabs to next tab stop |
| `<lf>` | Move cursor down 1 row |
| `<eol>` | Break the line at the cursor into two lines, and put the cursor at the start of the second line (like "QB" command) |

Table 8.2-1. Effect of Special Keys in Insert Mode

| Command | Insert: |
|---|---|
| nIL | n lines above current line, cursor to first inserted line (n is required) |
| QIL | lines above current line to end of window, cursor to first inserted line |
| nICx | n x's (n is required) |
| QICx | x's to right margin |

Table 8.3-1. Commands to Insert Characters or Lines

## 8.4. Inserting the Contents of a Buffer or File

The "1IB" command prompts for the name of a buffer and inserts the contents of the specified buffer in the current buffer above the current line. The "1IF" command prompts for the name of a file and inserts the contents of the specified file in the current buffer above the current line. See Table 8.4-1.

## 8.5. Inserting Page Marks

Type "P" to insert an end-of-page character immediately ahead of the current line in the buffer.

| Command | Insert: |
|---|---|
| 1IB | a buffer (name is asked) |
| 1IF | a file (name is asked) |

Table 8.4-1. Commands to Insert Buffers or Files

MAINEDIT displays the end-of-page character on your screen with a "page mark", as shown in Figure 1.10-1. To delete the end-of-page character, place the cursor anywhere on the line containing the page mark, and issue the "DL" (Delete Line) command.

## 8.6. Inserting Characters by Code

Associated with each character (printing and non-printing) is a unique nonnegative integer, called a "character code". MAINED provides a command for inserting a character by specifying its character code, thereby allowing you to insert a non-printing character such as a bell or escape.

The actual character code set used is dependent on your host operating system. For example, the character code for a blank on an ASCII host is 32, while that for a blank on an EBCDIC host is 64.

To insert a character by its code, type a single quote ("'"). MAINED prompts for the numeric character code in the message line at the top of the screen. Type the value of the character you wish to insert, followed by <eol>. The value can be entered in any of four radices, as shown in Table 8.6-1. For example, to insert an ASCII <del> character, type a single quote ("'"), and respond to the prompt in one of the ways shown in Example 8.6-2.

| Format | Description |
|---|---|
| n... | decimal digits n... |
| Bn... | binary digits n... |
| On... | octal digits n... |
| Hn... | hexadecimal digits n... |

Table 8.6-1. The Four Radices Understood by the "'" Command

```
Char code: 127<eol>                (decimal)
Char code: B1111111<eol>           (binary)
Char code: 0177<eol>               (octal)
Char code: H7F<eol>                (hexadecimal)
```

Example 8.6-2. Four Ways to Insert an ASCII Delete Character

# 9. Deleting and Recalling Text

Three commands, "D" (Delete), "K" (Kill), and "Z" (Zap), simultaneously remove text from your buffer and store it into the delete buffer implied by the command. Two additional commands, ".D" and ".Z", copy text into a delete buffer without removing it from the edit buffer. "..D", "..K", and "..Z" delete the text without storing it into a delete buffer. Text can be recalled from a delete buffer with the "R" (Recall) command.

## 9.1. Character, Word, and Line Delete Buffers

When text is deleted with the "D", "K", or "Z" commands, it is stored into what is called a "delete buffer". This text can then be retrieved using the "R" commands described in Section 9.5. The delete buffers act like pushdown stacks in that text is recovered in "last in, first out" order.

Separate delete buffers are used to store deleted characters, words, and lines. Each delete buffer can hold up to 300 objects. When a delete buffer is full, the least recently deleted objects are discarded.

Character, word, and line delete buffers cannot be displayed in a MAINEDIT window or edited with regular MAINEDIT commands.

## 9.2. Page Buffers

Each time a page is deleted with a "DP" (Delete Page) command, it is inserted into a page delete buffer for possible recall. The buffer is named "<DELETEDPAGES>", and may be edited like a normal buffer. The "DP" command in <DELETEDPAGES> immediately reinserts the deleted page at the end of the buffer; "..DP" must be used to delete a page from <DELETEDPAGES>.

## 9.3. Delete

The delete commands shown in Table 9.3-1 remove text from the edit buffer and copy it into a delete buffer.

The "QDL" command deletes up to the next page mark or the end of the current window, whichever comes first; the "-QDL" command deletes back to the previous page mark or the start of the current window.

| Command | Delete: |
|---|---|
| nK | n characters at and after |
| QK | all characters to the end of the line |
| -nK | n characters before |
| -QK | all characters to the start of the line |
| nD[C\|W\|L\|P] | n objects at and after |
| QD[C\|W\|L\|P] | all objects at and after |
| -nD[C\|W\|L\|P] | n objects before |
| -QD[C\|W\|L\|P] | all objects before |

Table 9.3-1. Text Deletion Commmands

The ".D" command copies text to a delete buffer without removing it from the edit buffer. It can be modified in the same way as shown for the "D" command in Table 9.3-1.

The "..D" command deletes text without copying it to the delete buffer. It can be modified in the same way as shown for the "D" command in Table 9.3-1.

"K" is a short form equivalent to "DC" (but ".K" means "Kill buffers", quite different from ".DC"). "..K" is equivalent to "..DC".

Text is moved or copied from one place to another by combining the "D" (Delete) and "K" (Kill) commands with the "R" (Recall) command (sometimes called "cut and paste").

To copy text, use ".D" to copy the desired objects into the delete buffer, which leaves the edit buffer unaltered. Then move the cursor to the destination location and use "R" (Recall) to insert the text. To move text instead of copying it, use the "D" command instead of ".D".

Copying and moving text to immediately adjacent parts of the buffer may also be accomplished with the "C" and "M" commands, as described in Sections 10.2 and 10.3.

The "..DP" form of the delete page command, which does not insert the deleted page into <DELETEDPAGES>, is considerably more efficient than "DP", and so should be used when the page is not going to be recovered.

## 9.4. Zap

The "Z" (Zap) command behaves like the "S" (Skip) command, except that the characters or lines that are be skipped by "S" are deleted by "Z". See Table 9.4-1.

| Command | Delete to: |
|---|---|
| nZx | nth next occurrence of x on current line |
| -nZx | nth previous occurrence of x on current line |
| +Zx | next occurrence of non-x on current line |
| -+Zx | previous occurrence of non-x on current line |
| QnZx | nth next line with x as first visible character (x may not be <sp>) |
| -QnZx | nth previous line with x as first visible char (x may not be <sp>) |
| +QZx | next line without x as first character |
| -+QZ | previous line without x as first character |
| QnZ<sp> | nth next all-blank line |
| -QnZ<sp> | nth previous all-blank line |

Table 9.4-1. The Zap Commands

Zapped characters and lines are stored in the character and line delete buffers, respectively. Use the ".Z" command to copy text into a delete buffer without removing it from the edit buffer. The ".Z" command can be modified in the same way as the "Z" command. The "..Z" command deletes without copying text into the delete buffer.

"QZ<sp>" is a convenient way to delete the current "paragraph", assuming paragraphs are separated by blank lines.

## 9.5. Recalling Text

Use the "R" (Recall) command to remove text from a delete buffer and insert it at the current cursor location. The recall command is always followed by an object (character, word, line, or page) that specifies the delete buffer from which the text is to be copied or moved.

In each case the recall command inserts the recalled text before the current object. Thus, recalled words, lines, and pages are inserted before the current word, line, or page, respectively, not into the middle of it.

Since character recall does not insert blanks that were not deleted, characters can be recalled into the middle of words. Word recall inserts spaces, if necessary, to separate the inserted text from the surrounding text; multiple blanks are eliminated.

When the "R" command is given, the number of objects may be specified in any of the three ways shown in Table 9.5-1.

| Command | Recall and insert: |
|---|---|
| R[C\|W\|L] | most recently stored group of objects |
| nR[C\|W\|L\|P] | most recently stored n objects |
| QR[C\|W\|L\|P] | all stored objects to delete-buffer mark |

Table 9.5-1. Text Recalling (Undeletion) Commands

If no count modifier is specified, then the most recently deleted "group" of objects is recalled. A "group" of objects has been deleted with a single delete command. For example, "3DW" deletes and stores three words from the edit buffer and stores them into the word delete buffer as a single group. A subsequent "RW" command recovers all three words at once.

If the count modifier is present, then the number of objects specified by the count is recovered, regardless of how the objects are broken into groups. "1RW" after a "3DW" command recalls only the last word deleted by the "3DW" command, leaving the other two words in the group in the word delete buffer.

If the count modifier specifies more objects than are present in the delete buffer, only as many objects as are present are recalled. If there are no objects in the appropriate delete buffer, a recall command has no effect.

The emphasis modifier "Q" is used to recall all objects up to the last delete-buffer mark. A delete-buffer mark is set with the commands ".M[C|W|L]". A message appears on the message line to let you know the mark is set. If no marks have been set, "QR[C|W|L]" recalls everything in the delete buffer.

Group and delete-buffer mark divisions are not maintained for page delete buffers. An "RP" command with no count recovers the last deleted page. "QRP" recovers all pages in all page delete buffers.

The commands ".R[C|W|L|P]" may be modified in the same way as the "R[C|W|L|P]" commands. ".R" inserts objects into the edit buffer without removing them from the delete buffer, so that the same object (or group of objects) may be recalled any number of times.

# 10. Other Text Modifying Commands

## 10.1. Center

The ".C" (Center) command centers the visible part of the current line of text within the current line. That is, the amount of white space between the left margin and the leftmost visible character on the line is made to be (to the best approximation possible) the same as the amount of white space between the right margin and the rightmost visible character on the line. See Table 10.1-1.

| Command | Center: |
|---|---|
| n.C | n lines (but not beyond end of window) |
| Q.C | all lines to end of window or page |

Table 10.1-1. Commands to Center Text on a Line

## 10.2. Copy

The "C" (Copy) command duplicates characters, words, lines, or pages. See Table 10.2-1.

| Command | Copy: |
|---|---|
| nC[C\|W\|L\|P] | n objects at and after |
| QC[C\|W\|L\|P] | all objects at and after |
| -nC[C\|W\|L\|P] | n objects before |
| -QC[C\|W\|L\|P] | all objects before |

Table 10.2-1. Commands to Copy Text

For example, suppose the cursor is under the word "orange" in the following sentence:

```
The bright orange balloon rose upwards.
           ^
```

Typing "CW" (Copy Word) copies the word "orange" and positions the cursor at the beginning of the second occurrence of the copied object:

```
        The bright orange orange balloon rose upwards.
                          ^
```

The count modifier specifies the number of objects to be copied, not the number of times an object is to be copied. Using the same sample sentence, the "2CW" command copies "orange balloon":

```
     The bright orange balloon orange balloon rose upwards.
                               ^
```

To copy the objects prior to the current object, modify the copy command with "-". Again using the "orange balloon" sentence as an example, the "-CW" command copies the word "bright" and positions the cursor at the beginning of the first occurrence of the copied object:

```
        The bright bright orange balloon rose upwards.
            ^
```

## 10.3. Move

The "M" (Move) command rearranges text by changing the order of characters, words, lines, or pages. See Table 10.3-1. The "M" commands without count or emphasis can be thought of as "swaps"; e.g., "MW" swaps the next two words.

| Command | Move current object: |
|---|---|
| nM[C\|W\|L\|P] | n further |
| QM[C\|W\|L\|P] | to end (of line, line, window, or buffer, respectively) |
| -nM[C\|W\|L\|P] | n earlier |
| -QM[C\|W\|L\|P] | to start (of line, line, window, or buffer, respectively) |

Table 10.3-1. Text Rearranging Commands

For example, suppose the cursor is positioned at the "n" in the word "orange" in the following sentence:

```
        The bright orange balloon caught their attention.
                          ^
```

The "MW" (Move Word) command moves the word "orange" one word to the right, positioning the cursor at the beginning of the object beyond the one moved:

```
        The bright balloon orange caught their attention.
                       ^
```

Typing "MC" (Move Character) moves the character at the cursor one position to the right, and positions the cursor at the character beyond the one moved:

```
        The bright oragne balloon caught their attention.
                          ^
```

The "-" modifier changes the direction of the move to be towards the left, or towards the beginning of the buffer. "-MW" applied to the original sentence results in:

```
        The orange bright balloon caught their attention.
                       ^
```

"-M" forms leave the cursor at the beginning of the object that was originally located immediately before the moved object.

## 10.4. Break

The "B" (Break) command breaks a line at the current cursor position into two lines. Modifiers are used to specify where to position the first visible character on the second line, and where to leave the cursor. See Table 10.4-1.

When a count n is specified, the first visible character on the second line is positioned to column n + 1. The "spaces" added by the "nB" forms are actually tab characters. They may be replaced with spaces when the file is written out in accordance with the rules in Section 1.11.

When the "-" modifier is specified, the cursor is positioned after the last character on the first line. Otherwise, it is positioned on the first visible character of the second line.

## 10.5. Join

The "J" (Join) command joins two lines. Modifiers are used to specify which lines are joined, and how many spaces are to separate the two joined pieces of the new line. See Table 10.5-1.

The spaces inserted by the "nJ" forms are space characters, not tab characters.

| Command | Break line at cursor, then: |
|---|---|
| B | remove spaces, cursor to second line |
| nB | indent n spaces, cursor to second line |
| QB | leave original spaces, cursor to second line |
| -B | remove spaces, cursor to end of first line |
| -nB | indent n spaces, cursor to end of first line |
| -QB | leave original spaces, cursor to end of 1st line |

Table 10.4-1. Commands to Break a Line

| Command | Join: |
|---|---|
| nJ | next to current line, n separating spaces |
| QJ | next to current line, leave original spaces |
| -nJ | current to previous line, n separating spaces |
| -QJ | current to previous line, leave original spaces |

Table 10.5-1. Commands to Join Two Lines

## 10.6. Convert to Lower or Upper Case

The "L" (Lower case) and "U" (Upper case) commands convert alphabetic text to lower or upper case, respectively. They have no effect on non-alphabetic characters. See Table 10.6-1. These commands position the cursor at the beginning of the object beyond the last one converted.

| Command | Convert: |
|---|---|
| nL[C\|W\|L] | n objects to lower case |
| QL[C\|W\|L] | all objects to lower case |
| nU[C\|W\|L] | n objects to upper case |
| QU[C\|W\|L] | all objects to upper case |

Table 10.6-1. Commands to Convert to Lower/Upper Case

## 10.7. Filling and Justification

The filling and justification (".J") commands are shown in Figure 10.7-1. They provide a way to fill paragraphs and optionally justify the right margin. Filling a paragraph eliminates excess blank space within and at the end of lines; justifying aligns a margin, so that the entire column at the margin contains printing characters.

| Command | Effect |
|---|---|
| `.J` | Fill current paragraph to right margin of window |
| `n.J` | Fill n lines |
| `+.J` | Fill and justify to right margin of window |
| `-.J` | Fill starting at cursor column |
| `Q.J` | Fill all remaining paragraphs in buffer |
| `.mJ` | Fill to right margin in column m |
| `nQ.J` | Fill next n paragraphs |

(All modifiers may be combined; i.e., nQ+-.m.J means fill next n paragraphs from cursor column, justifying to column m).

Figure 10.7-1. Commands to Fill and Justify Text

The ".J" command with no modifiers fills the current paragraph. The current paragraph starts with the first line at or beyond the current line that contains a visible character in the current column (column 1 if the minus modifier ("-") is not specified) and extends either to the first line that contains an invisible character in the current column or to a page mark, whichever comes first. With this form of the command, it does not matter where the cursor is on the line. Multiple spaces in a filled paragraph are changed to a single space.

If a count (n) is specified, then only the first n lines of the paragraph are filled. If n is larger than the number of lines in the current paragraph, only the current paragraph is filled.

If n is not specified, the emphasis modifier ("Q") means fill all remaining paragraphs in the buffer; if n is specified, it means fill the next n paragraphs.

The plus modifier ("+") means fill with an even right margin ("justify"). Spaces are scattered in the filled text to make the right margin of all but (possibly) the last line even.

The ".m" modifier means fill with the right margin set to m (the displayed right margin of the window remains unchanged). If ".m" is not specified, the right margin of the window is used.

The minus modifier ("-") means fill between the current column and the right margin only. This form of the command is useful when filling indented text. When using this form of the fill command, make sure that the cursor is positioned in the column at which filling is to begin; all characters to the left of this column are ignored and remain unaltered. If, after filling, new line(s) are created, the text to the left of the current column on the existing filled lines is examined. If this text is identical on all filled lines, then that text is inserted to the left of the current column on every new line added as a result of the fill. Otherwise, blanks are added to the left of the current column on new line(s). This seems to give desirable results in most cases.

Some examples of the ".J" command are shown in Examples 10.7-2, 10.7-3, and 10.7-4.

```
A window that originally looks like this:

    :                                        :
    :The ".m" modifier means fill with the ri*
     ^
    :actual right margin remains unchanged). *
    :current right margin is used.           :
    :                                        :

looks like this after ".J":

    :                                        :
    :The ".m" modifier means fill with the   :
    :right margin set to m (the actual right :
    :margin remains unchanged). If ".m" is   :
    :not specified, the current right margin :
    :is used.                                :
    :                                        :


and looks like this after "+.J":

    :                                        :
    :The ".m"  modifier  means fill  with the:
    :right  margin set to m (the actual right:
    :margin remains  unchanged).  If  ".m" is:
    :not specified, the current right  margin:
    :is used.                                :
    :                                        :

The position of the cursor on the first line does not
matter; i.e., it can be positioned anywhere on the first
line.
```

Example 10.7-2. ".J" Command, Example 1

```
A window that originally looks like this:

    :                                         :
    :trunc(x)     x is a real value; the resul*
                  ^
    :             (The fractional part is disc*
    :             trunc(3.7) = 3 and trunc(-3.*
    :                                         :
    :round(x)     x is a real value; the resul*
                  ^
    :             integer. round(x) means for *
    :             for x < 0 trunc(x - 0.5)    *
    :                                         :

looks like this after moving the cursor to each position
indicated by "^" and typing "-.J":

    :                                         :
    :trunc(x)     x is a real value; the      :
    :             result is its whole part.   :
    :             (The fractional part is     :
    :             discarded. Hence trunc(3.7) :
    :             = 3 and trunc(-3.7) = -3)   :
    :                                         :
    :                                         :
    :round(x)     x is a real value; the      :
    :             result is the rounded       :
    :             integer. round(x) means for :
    :             x >= 0 trunc(x + 0.5), and  :
    :             for x < 0 trunc(x - 0.5)    :
    :                                         :

The cursor must be positioned as shown.
```

Example 10.7-3. ".J" Command, Example 2

```
A window that originally looks like this:

    :                                         :
    :# Window bp got bigger (it may have just*
       ^
    :# are pushed down (possibly disappearing*
    :# ctrlBit is set; otherwise they are ove*
    :# pushed up (possibly disappearing) if t*
    :# set; otherwise they are overlayed.    *
    :                                         :

looks like this after "-.J":

    :                                         :
    :# Window bp got bigger (it may have just:
    :# been created). Lower windows are      :
    :# pushed down (possibly disappearing) if:
    :# the pushWindowsBelow ctrlBit is set;  :
    :# otherwise they are overlayed. Higher  :
    :# windows are pushed up (possibly       :
    :# disappearing) if the pushWindowsAbove :
    :# ctrlBit is set; otherwise they are    :
    :# overlayed.                            :
    :                                         :

and looks like this after "+-.J":

    :                                         :
    :# Window bp got bigger (it may have just:
    :# been    created).   Lower  windows  are:
    :# pushed down (possibly disappearing) if:
    :# the pushWindowsBelow ctrlBit  is  set;:
    :# otherwise  they  are overlayed. Higher:
    :# windows   are  pushed   up   (possibly:
    :# disappearing) if  the pushWindowsAbove:
    :# ctrlBit  is set;  otherwise  they  are:
    :# overlayed.                            :
    :                                         :

The newly created lines start with a pound sign and a
blank, since that text started all of the original lines.
The cursor must be positioned as shown.
```

Example 10.7-4. ".J" Command, Example 3

# 11. Windows

A window is a section of the screen used to display the contents of a particular buffer. Just as you can have more than one edit buffer during an editing session, you can have more than one window on the screen. Example 11-1 shows a screen with two windows.

```
status line
for top window  -> ---P.36--L.3---C---FOO----------------
                   :                                    :
                   :This top window shows part of the   :
                   :buffer named "FOO".  Editing commands:
                   :are executed in this window.        :
                   :The cursor is here: _               :
status line for    :                                    :
bottom window   -> ---P.2---L.47------BAR----------------
                   :                                    :
                   :In the bottom window is text from the:
                   :buffer "BAR".  This window is       :
                   :currently inactive, since the cursor :
                   :is in the other window.             :
```

Example 11-1. A Screen with Two Windows

The status line at the top of each window reports the name of the buffer and the page and line number of the top line of the window. In addition, the status line of the window in which the cursor is positioned contains a letter indicating the mode, e.g., "C". See Chapter 1 for more information on the status line.

The window in which the cursor is positioned is the "current window". To move the cursor to a buffer that is not in the current window, use the ".B" command, described in Chapter 12, or the ".Y" command, described in Chapter 6.

## 11.1. Changing Window Size

Window sizes are limited only by the size of your terminal screen. The "QX" and "QY" commands change the width and height of the current window. See Table 11.1-1.

| Command | Action |
|---|---|
| QnX | Set right margin to column n |
| -QnX | Set left margin at column n |
| -Q0X | Set left margin to line origin |
| QnY | Set window height to n rows |
| Q0Y | Kill current window (prompts for new buffer) |
| QY | Make current window as large as possible |
| Q+Y | Expand current window to bottom of screen |
| Q+nY | Expand current window by n rows |
| -QY | Synonym for Q0Y |
| -QnY | Shrink current window by n rows |

Table 11.1-1. Window Management Commands

In addition to changing the size of a window, the "Q+{n}Y", "Q-{n}Y", and "Q{n}Y" commands associate a default size with a window. The window's default size is used whenever it is made visible (unless overridden by the count modifier (n) of the command used to make the window visible).

All forms of the "Y" command that change a window's size are ignored when in proportionalWindowsMode (except "QY", which sets the current window to the maximum size). proportionalWindowsMode is discussed in Chapter 18.

The "-QnX" command provides left-right scrolling of a window. For example, if columns one through 72 are currently visible on the screen, then "-Q50X" makes columns 50 through 121 visible. An asterisk is displayed instead of a colon on the left margin of each line that contains characters whenever the leftmost column displayed is not column one. See Example 11.1-2.

## 11.2. Anchoring Windows

When a window is anchored, it remains stationary on the screen until it is unanchored. The size of an anchored window does not change unless explicitly changed with the "{-}{+}{n}QY" command. Unanchored windows, by contrast, may be moved or changed in size when a new window is created.

MAINED commands that manipulate windows, e.g., "B", "F", "Y", etc., use only the unanchored portion of the screen when bringing an unanchored window into view and when changing the size of an unanchored window (MAINED ensures that there are at least two unanchored lines on the screen). In other words, anchoring a window effectively reduces the size of the screen available for unanchored window manipulation.

```
A window that originally looks like this:

  ---P.4---L.6---C---XYZ---------------
  :This buffer has some text that is in*
  :because it runs beyond the right mar*
  :The text can be viewed by scrolling *
  :screen to the left.                 :
  :                                    :
  :When the screen is scrolled left, no*
  :lines have an asterisk in the leftmo*
  :column instead of a colon.          :

looks like this after a "-Q10X" command is
issued:

  ---P.4---L.6---C---XYZ---------------
  *er has some text that is invisible  :
  *t runs beyond the right margin.     :
  *can be viewed by scrolling the      :
  * the left.                          :
  :                                    :
  *screen is scrolled left, non-blank  :
  *e an asterisk in the leftmost       :
  *stead of a colon.                   :
```

Example 11.1-2. Left-Right Scrolling

One or more windows can be anchored at the top of the screen and/or at the bottom of the screen. When a window is anchored at the top of the screen, it is anchored immediately below the bottommost anchored top window (at the top if there are no windows anchored at the top). When a window is anchored at the bottom of the screen, it is anchored immediately above the topmost anchored bottom window (at the bottom if there are no windows anchored at the bottom).

The cursor is moved to an anchored window with normal MAINED commands, e.g., ".Y" and ".B". Once a window is anchored, it remains anchored until it is explicitly unanchored or until it is killed. An anchored window is independent of the buffer in it. Even when the buffer in an anchored window is changed, e.g., with the ".B" or ".F" commands, the window remains anchored. However, if the buffer in a window is killed, the window is also killed, and hence, unanchored.

Figure 11.2-1 shows the commands used to anchor and unanchor windows.

| Command | Action |
|---|---|
| .A | anchor current window |
| +.A | anchor at bottom |
| -.A | anchor at top |
| n.A | anchor, change size to n rows |
| +n.A | anchor at bottom, change size to n rows |
| -n.A | anchor at top, change size to n rows |
| ..A | unanchor current window |
| Q..A | unanchor all windows |
| Q+..A | unanchor all windows at bottom of screen |
| Q-..A | unanchor all windows at top of screen |

Figure 11.2-1. Commands Used to Anchor and Unanchor Windows

The ".A" command anchors the current window. If no modifiers are specified and the window is immediately above the topmost anchored bottom window (or at the bottom of the screen if there are no windows anchored at the bottom), then it is anchored at the bottom. Otherwise, it is anchored at the top. If the minus modifier ("-") is specified, the window is anchored at the top of the screen. If the plus modifier ("+") is specified, the window is anchored at the bottom of the screen.

The anchored window is adjusted to be n rows (its size remains unchanged if n is not specified). Any number of windows can be anchored, as long as there are at least two unanchored lines on the screen.

The "..A" command unanchors the current window. If the window is anchored at the top of the screen and there are other windows below it that are also anchored at the top, then the window is repositioned immediately below the bottommost anchored top window. Similarly, if the window is anchored at the bottom of the screen and there are other windows above it that are also anchored at the bottom of the screen, the window is repositioned immediately above the topmost anchored bottom window. The emphasis modifier ("Q") means unanchor all windows; "Q-" means unanchor all windows anchored at the top of the screen; "Q+" means unanchor all windows anchored at the bottom of the screen.

# 12. Files and Buffers

During a MAINEDIT session, you may want to edit more than one file. MAINEDIT provides for this by maintaining a "buffer" for each file you use in an edit session. When a new file is brought in for editing, MAINEDIT creates a new buffer and copies the file into the buffer (as needed; it does not necessarily read it all at once). As you edit the buffer, the original copy of the file is not affected. The buffer automatically grows and shrinks as you edit. The file is updated only when explicitly saved. The original file is replaced, or a new version created, as described in Chapter 14.

MAINED provides a complete set of commands to manipulate files and buffers; you may want to use some of these commands from other front ends as well as from MAINED, since some of the front ends are based on editors that do not have as flexible a notion of file and buffer as MAINED, and therefore do not have all the commands that MAINED does.

## 12.1. Bringing a File into the Editor

The ".F" (File) command moves the cursor into a buffer associated with the file of the specified name. If a buffer associated with the file does not exist, it is created.

When any of the commands in Table 12.1-1 is issued, MAINEDIT prompts for the file name "s" on the message line at the top of the screen. The subsequent dialogue is the same as that described for the initial file in Chapter 1, except that if you type the name of a file for which a buffer already exists, MAINEDIT prompts to confirm that you want to use the existing buffer. The ".B" command is usually used to move into an existing buffer; the ".F" command is intended primarily for creating new buffers.

## 12.2. Moving the Cursor among Buffers

The ".B" (Buffer) command is used to move the cursor to a specified buffer.

When any of the commands in Table 12.2-1 is issued, MAINEDIT prompts for the buffer name "s" on the message line at the top of the screen. If the buffer exists, MAINEDIT positions the cursor there; if not, MAINEDIT inquires whether you wish to create a new buffer.

If the buffer you want to edit is already visible in a window on the screen, the cursor is moved to that window and is positioned at the last place you edited in that buffer.

| Command | Action |
|---|---|
| .Fs | edit file s; use current window if not on screen |
| n.Fs | same as ".Fs", except n-row window (n required) |
| +.Fs | edit file s; insert new window at bottom if not on screen |
| +n.Fs | same as "+.Fs", except n-row window (n required) |
| -.Fs | edit file s; insert new window at top if not on screen |
| -n.Fs | same as "-.Fs", except n-row window (n required) |
| --{n}.Fs | overlay (n-row) window at top |
| ++{n}.Fs | overlay (n-row) window at bottom |
| ..{n}Fs | edit s, making window 1/mth of screen, where m is the number of windows; but no window is allowed to be smaller than n lines |
| Q.Fs | change file name of current buffer to s |

Table 12.1-1. File Selection Commands

| Command | Action |
|---|---|
| .Bs | edit buffer s; use current window if not on screen |
| n.Bs | same as ".Bs", except n-row window (n required) |
| +.Bs | edit buffer s; insert new window at bottom if not on screen |
| +n.Bs | same as "+.Bs", except n-row window (n required) |
| -.Bs | edit file s; insert new window at top if not on screen |
| -n.Bs | same as "-.Bs", except n-row window (n required) |
| --{n}.Bs | overlay (n-row) window at top |
| ++{n}.Bs | overlay (n-row) window at bottom |
| ..{n}Bs | edit s, making window 1/mth of screen, where m is the number of windows; but no window is allowed to be smaller than n lines |
| Q.Bs | change buffer name of current buffer to s |

Table 12.2-1. Buffer Selection Commands

## 12.3. Modifiers of the ".F" and ".B" Commands

The plus modifier ("+") means insert the window at the bottom if it is not already on the screen (all windows above are pushed up). If n is specifed the new window is n rows; otherwise it occupies the bottom half of the screen.

The minus minus modifier ("--") means overlay the window at the top if it is not already on the screen. If n is specified, the new window is n rows; otherwise it occupies the top half the screen.

The plus plus modifier ("++") means overlay the window at the bottom if it is not already on the screen. If n is specified, the new window is n rows; otherwise it occupies the bottom half of the screen.

The dot dot modifier ("..") means make all windows on the screen the same size (proportional windows). A new window is created if it is not already on the screen. If there are m windows on the screen, then the size of each window is adjusted so that the window occupies 1/mth of the screen. n is the minimum number of rows (excluding the status line) in a window (1 if n is unspecified). n is sticky, i.e., it remains in effect until changed by a subsequent "..{n}B" or "..{n}F" command. When a new window is created, some existing windows may be killed to ensure that all remaining windows are at least n rows. The "..{n}B" and "..{n}F" commands do not affect the setting of the proportionalWindowsMode option discussed in Chapter 18.

When the ".B" commands prompt for the buffer name, they offer as a default the name of the buffer in which the cursor was located before it was moved to the current buffer, if any. See Example 12.3-1. The default makes it simple to bounce back and forth between two buffers.

```
Buffer (MEMO): <eol>            (to go to buffer "MEMO")

Buffer (MEMO): OUTLINE<eol>     (to go to buffer "OUTLINE")
```

Example 12.3-1. Sample Use of ".B"

The ".B" command accepts a string that is a substring of an existing buffer name, e.g., "OUT" or "LINE" would be sufficient to specify the buffer "OUTLINE". If you give an answer that is a substring of more than one existing buffer name, one of the buffers is arbitrarily chosen by MAINEDIT. If the name you specify is not a substring of an existing buffer, MAINEDIT asks:

```
New buffer OUTLINE (Yes No)?
```

Type "Y" (or "y") to create a new buffer. If you do so, a buffer containing a single blank line is created and the cursor moved to its window.

If you type "N" (or "n") to the "New buffer" prompt, the original "Buffer:" prompt is repeated. You may wish to use the <abort> key to discontinue the dialogue if you decide you do not want to change buffers after all; see Section 15.1.

The ".Y" command (see Chapter 6) provides a simple means of moving among buffers that are visible on the screen.

## 12.4. Deleting Buffers

You are encouraged to delete a buffer if you are finished with it, since this frees the memory space used by the buffer. The ".K" (Kill buffers) command causes MAINED to prompt for each buffer asking whether you want to kill it. Each prompt is as follows:

```
Kill buffer MEMO (Yes No)?
```

Type "Y" (or "y") to delete the buffer, "N" (or "n") to retain it. If you delete a buffer that has been changed since it was last saved, MAINEDIT prompts:

```
Write buffer MEMO (Yes No)?
```

Type "Y" (or "y") to save the text in this buffer on a file (if you do not do so, your changes to the buffer are lost). When a buffer in a window on the screen is deleted, it disappears from the screen. If the current buffer is killed, MAINEDIT prompts for a new buffer to which to move the cursor (unless all the buffers have been killed, in which case it prompts for a file to edit).

The form "Q.K" is used to kill just one buffer. This command prompts for the name of the buffer to kill, the default being the current buffer.

## 12.5. Changing Buffer and File Names

Buffer and file names are changed with the "Q.Bs" and "Q.Fs" commands. See Table 12.5-1.

MAINED prompts for "s" on the message line. For "Q.B", MAINED offers the file name associated with the current buffer, converted to upper case, as the default buffer name. Type the new buffer name, or just <eol> to accept the default:

```
New buffer name (BAR): FOO<eol>
    (to change the current buffer's name to FOO)
```

| Command | Action |
|---|---|
| Q.Bs<eol> | Change buffer name of current buffer to "s" |
| Q.B<eol> | Change buffer name to current file name in upper case |
| Q.Fs<eol> | Change file name of current buffer to "s" |

Table 12.5-1. Commands to Change Buffer and File Names

```
        New buffer name (BAR): <eol>
             (to change the current buffer's name to BAR)
```

The file name associated with a buffer is the default name of the file into which the text of the buffer is copied when the buffer is saved. The file name associated with a buffer is independent of the buffer name, but by default the buffer name is the same as its associated file name converted to upper case.

## 12.6. Displaying Information about Buffers and Files

The "Q=" command displays a list of the current buffers and their status. The list is displayed in a buffer named "CMDLOG" (ComManD LOG). If CMDLOG does not exist or is invisible, it is created or brought into view as described in Chapter 19.

Example 12.6-1 shows the format of the information displayed by the "Q=" command. The buffer name of each edit buffer is displayed in the "BUFFER" column. The "FILE" column shows the name of the file associated with the buffer, if any. The "STATUS" may contain three different characters:

- D means that the buffer is dirty (has changes that have not been saved).
- S means that the buffer has been saved during the current edit session (and hence was once dirty).
- A means that the buffer is anchored.

```
STATUS BUFFER                       FILE
------ ------                       ----
D      CMDLOG
D A    EDTMGR.MSS                   edtmgr.mss
 S     FOO                          foo
D A    EDTMGR-UPGRADE.TXT           edtmgr-upgrade.txt
       EDIT.MSL                     edit.msl
       MAINED.MSL                   mained.msl
```

Example 12.6-1. Sample Output from the "Q=" Command

## 12.7. Specifying Front and Back Ends for a Buffer; Multiple Buffers into the Same Data

The ".B" and ".F" commands allow you to specify a front and/or back end different from the default. The ".B" command can also be used to create multiple buffers into the same data (so that more than one window on the screen can be examining the same buffer or file at a time).

When the ".B" and ".F" commands are typed, the user is prompted for a buffer or file name, respectively. If the specified name is followed by a comma, then MAINEDIT prompts for the names of the front and back ends to be used for the buffer. Defaults are offered in parentheses. The currently supported back ends are DATMGR and TXTMGR.

In addition to the standard back ends, MAINEDIT recognizes the special form "=foo". In this case, MAINEDIT assumes that "foo" is the name of a buffer (partial matches are recognized) and it shares that buffer's back end as the back end of the new buffer. This means that the new buffer shares the same data as the buffer "FOO". The response "=<eol>" is equivalent to "=foo<eol>" if "foo" is the current buffer.

When changes are made to data shared by more than one buffer, the changes are reflected in all buffers that share the data (there is only one copy of the data). In addition, all windows in which the affected data are visible are updated. There is no limit on the number of buffers that can share the same back end.

Figure 12.7-1 shows how to create a new buffer "FOO2" that shares the same data as an existing buffer "FOO".

During MAINEDIT's initial dialogue, the user is prompted for the name of a file to edit. If the last character of the response is a comma, then MAINEDIT strips off the comma and prompts for the display module and baud rate. When MAINEDIT creates the initial buffer, it once again

```
-.B                 Create a new buffer (at top of screen)
foo2,<eol>          Call it foo2, enter subcommand mode
Y                   Yes to "New buffer" prompt
<eol>               Use default front end
=foo<eol>           Use foo's back end for foo2
```

Figure 12.7-1. How to Get Multiple Buffers into the Same Data

examines the last of the remaining characters. If it is a comma, MAINEDIT strips it off and prompts for the names of the front and back ends to be used for the buffer. Thus, to specify a front or back end different from the default for the initial buffer, type two commas after the file name.

# 13. Command and View Front Ends

The MAINED command and view front end commands are summarized in Chapter 3. They are repeated here for convenience:

| Command | Effect |
|---|---|
| Q..Bs {cmds} | Change command front end |
| +Q..Bs {cmds} | Change command and view front ends |
| Q..Vs | Change view front end |
| -Q..Bs | Kill front end |
| +Q= | Show status including front ends |

# 14. Saving and Finishing

The "F" (File) commands allow you to save changes made to buffers and/or exit from MAINEDIT. See Table 14-1.

| Command | Prompt to save altered buffers, then: |
|---|---|
| F | return to current location |
| QF | If a program invoked with "QE" is running, raise the exception $abortProgramExcpt; otherwise, exit from MAINEDIT, continuation not allowed |
| -QF | exit from MAINEDIT and save context for continuation, if continuation possible |

Table 14-1. Commands to Save Files and/or Exit from MAINEDIT

When you type one of the "F" commands, MAINEDIT prompts on the message line for each buffer (except CMDLOG, if it exists; see Chapter 19) that you have altered since it was last saved (or marked as unaltered):

```
        Write buffer FOO.BAR (Yes No Mark-as-unaltered)?
```

If you type "Y" or "y" in response, MAINEDIT attempts to write the buffer to its associated file. If you type "N" or "n", MAINEDIT does not write the changes to the file. If you type "M" or "m" (for "Mark-as-unaltered"), MAINEDIT forgets that you have made any changes to the buffer. This option is not offered for "QF".

Use of any of the "F" commands updates the "eparms" file.

When MAINEDIT attempts to write a buffer to its associated file on an operating system without file version numbers, it first prompts whether to overwrite the existing file. If you choose not to do so, you may specify a new file name to which the buffer is to be written. The new file name becomes the file name associated with the buffer. If, for any reason, MAINEDIT cannot open for output a file that it is trying to write, it issues an error message to that effect, and offers you the option of typing <eol> to select a new file name, or <abort> to abort the current command.

"QF" blanks the screen upon exit.

"-QF", on systems where it is supported, exits directly from MAINEDIT to the operating system without blanking the screen. On operating systems that permit it, the state of the MAINEDIT session is maintained, so that if an operating-system-dependent command is used to restart the MAINEDIT program, the edit session continues from the point at which it was suspended. When you continue MAINEDIT in this way, the screen is refreshed and the cursor positioned where it was when the "-QF" command was given.

# 15. Miscellaneous Commands

## 15.1. <abort>

The <abort> key is used to terminate a multi-character command before type-in of the command characters is complete. This includes commands that prompt for responses; i.e., the <abort> key may be typed to a prompt.

On some operating systems, the <abort> key may also be used to terminate:

- A recursive macro that is executing.
- A search that is executing.
- A "V" or "G" command that is executing.

When the <abort> key is used, MAINED rings the terminal bell. If you were in insert or overstrike mode, you are returned to command mode. For example, if you accidentally type a command such as ".F" (to edit another file), you can type the <abort> key in response to the prompt in order to terminate the command.

When <abort> is used to terminate a macro or other executing command, there may be some delay before the command is actually terminated.

The actual key (or sequence of keys) used for <abort> is terminal-dependent. Refer to the information in Appendix C on your display module to determine the <abort> key for your terminal.

<ecm> also terminates multi-character command type-in and prompt response.

## 15.2. Again

Use the "A" (Again) command to repeat the most recently executed command. Modifiers to the "A" command allow you to repeat the previous command exactly as it was typed (including any modifiers) or change the direction and/or count. See Table 15.2-1.

The "-" direction is towards beginning of file, and the "+" direction is towards the end of file.

| Command | Do last command again, with: |
|---|---|
| nA | count = n, original direction |
| QA | original count and direction |
| +nA | count = n, "+" modifier |
| Q+A | original count, "+" modifier |
| -nA | count = n, "-" modifier |
| -QA | original count, "-" modifier |

Table 15.2-1. Forms of the Again Command

"A" alone is especially useful for commands like "S" (Skip). For example, to get to a particular instance of the character "," in a line of text, you might type "3S," (skip to the third comma in the line). If the comma you want turns out to be the fourth in the line, typing "A" advances you to the next comma, since the omitted count for the "A" command defaults to 1; the effect is as if "1S," had been typed.

As another example, suppose you type "27K" to delete 27 characters. If the next command is "QA", another 27 characters are deleted. Typing "5A" deletes only five characters; typing "A" alone deletes one character. "-A" deletes to the left.

## 15.3. Undo

The "H" ("Hoops!") undoes the last change to the current buffer. A "change" is everything altered since the last command typed. For example, if you type a macro that makes many modifications, "H" undoes them all. Also, changes that affect contiguous text on the same line are treated as a single change; e.g., when "KKK" deletes three characters, all three are restored by a single "H" command. Some changes may be undone with "H", then "redone" with "-H" if you did not really want to undo them. See Table 15.3-1.

| Command | Action |
|---|---|
| nH | undo previous n changes |
| QH | undo all changes on current line |
| -nH | redo next n changes |
| -QH | redo all changes on current line |

Table 15.3-1. Undoing Previous Commands on the Current Line

There is currently a limit of 300 "undos" per buffer; i.e., only the most recent 300 changes can be undone.

The "H" command does not work in CMDLOG.

## 15.4. Status Report

The "=" command displays a brief status report in the message line at the top of the screen. The report looks something like:

```
        B=MANUAL F=mainedit-manual.txt P.L=41.48 X=1 C=-1
```

The fields are described in Table 15.4-1. Another form of the "=" command is described in Chapter 13.

| Field Name | Meaning |
|---|---|
| B | name of current Buffer |
| F | name of current File (shown only if buffer name is not equal to file name, case not distinguished) |
| P.L | current Page and Line |
| M | current left Margin (shown only if scrolled left) |
| X | current column (X-coordinate) |
| C | decimal code for the current Character (-1 means there is no current character; the cursor is beyond the end of the current line) |

Table 15.4-1. Fields Displayed by the "=" Command

## 15.5. Suppress Output

Many operating systems have a command to "suppress terminal output". This is usually a control key, referred to here as the "<suppress-output> key". If you press the <suppress-output> key, MAINEDIT continues to function internally as usual, but the screen is not updated. Typically, screen update remains inhibited until you press the the <suppress-output> key again.

Suppressing terminal output is useful if the screen is filling up with text that you do not really want to see (e.g., you are immediately going to "G" (Go) to a different page). It can, however, have the effect of confusing MAINEDIT's screen update algorithm, so that the display is incorrect until the screen is refreshed (with the "QN" command).

## 15.6. Refresh Screen

The "N" command refreshes (rewrites) text on the screen. This is useful if the screen becomes garbled. For example, some operating systems write messages to the user without regard to what may already be on the screen. The <suppress-output> key can also cause the screen to be updated incompletely. The "N" command can be modified with a direction, count, or emphasis, as shown in Table 15.6-1.

| Command | Refresh: |
|---|---|
| "0N" | current status line |
| "nN" | n lines at and below in current window |
| "-nN" | n lines above in current window |
| "QN" | the entire screen |
| "Q.N" | the current window |

Table 15.6-1. Commands to Refresh Part or All of the Screen

## 15.7. Setting Tab Stops

The "Q<tab>" command allows you to specify the columns at which the <tab> command stops. Tab stops are specified as an increasing sequence of positive integers separated by spaces and terminated by <eol>. The integers specify successive 1-origin columns for tab stops. A tab stop is always assumed at column 1. Additional tab stops beyond the last explicitly specified tab stop are defined at increments of n columns, where n is the difference between the last two numbers in the tab stop sequence. For example, the sequence "5 17 23 40<eol>" puts tab stops at columns 1, 5, 17, 23, and 40, as well as at 57 (40 + 17), 74 (57 + 17), and so forth, since 40 - 23 = 17. As another example, the sequence "5<eol>" puts tab stops at every fourth column (5 - 1 = 4). Tab stops are initially set at every eighth column, starting at column one, i.e., the setting specified by the command "Q<tab>9<eol>".

Typing <eol> in response to the "Q<tab>" prompt means to use the current line in the edit buffer as the tab sequence. The current line should contain a sequence of integers as defined above.

"?<eol>" in response to the tab stop prompt displays the sequence for the current tab stop setting. This sequence is also stored in the line buffer, and hence can be recalled with the "RL" command. The recalled line could then be edited, and used to reset tabs via "Q<tab><eol>" as described in the previous paragraph.

The <tab> command (in command, insert, and overstrike modes) uses the current tab stop settings to determine the target column.

## 15.8. Changing the Frequency of Save Reminders

MAINEDIT may periodically display a suggestion that you use the "F" command to save your buffers. MAINEDIT maintains a variable called autoSaveLimit, which is the number of keystrokes between these save reminders.

The "+F" command is used to change the value of autoSaveLimit. When the command is issued, MAINEDIT offers the old value as a default and prompts for a new value. You may type <eol> to leave the value of autoSaveLimit unchanged. The smaller the value of autoSaveLimit, the more frequent the save reminders.

An autoSaveLimit of 0 disables the autoSaveLimit feature.

# 16. Macros

MAINED provides macros to facilitate repeated execution of MAINED command sequences. A macro can be either a "macro command" or a "named macro".

A macro command can be invoked with any "control sequence" that is not intercepted by the operating system/terminal combination. A control sequence is usually generated by a single terminal key (also called a "macro ID key"), although some terminals permit multi-character control sequences. Many terminals have keypad keys that function as MAINED macro ID keys and can, therefore, be used to invoke macro commands. Refer to Appendix C for a list of the macro ID keys and multi-character control sequences on your terminal.

A named macro is invoked by typing an identifier in response to the "Macro name:" prompt of the "," command; see Section 16.2.

## 16.1. Defining a Macro

The "/" command is used to define a macro:

- Type "/" to indicate that you are beginning a macro definition.
- MAINED prompts with "Macro id:". Type either:
    - A macro command control sequence, usually a macro ID key, as described in Appendix C.
    - A named macro name (any series of printing characters), terminated by <eol>. Letters are displayed in upper case.
- Your response to the "Macro id:" prompt is the name of the macro being defined. MAINED displays a "D" in the status line, just after the mode character, to indicate that you are defining a macro.
- Perform any sequence of commands (except other macro definitions), no matter how long. Macro invocations may be included in the command sequence. While MAINED performs your commands as usual, it also saves them as the macro definition.

- Terminate the macro definition by giving the "/" command again. The "D" disappears from the status line to let you know that you are no longer defining a macro.

A macro definition is undefined (i.e., forgotten) if a null command sequence is given in its definition. For example, a macro CTRL-A is undefined by typing the characters "/", CTRL-A, then "/". A macro that has been undefined reacquires its default value; keys that are predefined by MAINED (e.g., arrow keys) perform their predefined functions after they have been undefined.

Macro definitions are remembered in the "eparms" file; they are therefore available across edit sessions. The terminal type on which a macro is defined is remembered when the macro is saved in the "eparms" file. A macro defined on one terminal cannot be invoked from another terminal; they must be redefined for each terminal type. This avoids confusion, since two different terminals may generate the same control sequence for entirely different keys.

The keys <bs>, <tab>, <lf>, and <eol> are available as macro ID keys. They may be redefined like any other macro command. However, if these keys are redefined, commands containing them no longer function as described in this manual. It is not possible (or desirable) to redefine <ecm>.

## 16.2. Invoking a Macro

Macro commands are invoked by typing the macro ID key or control sequence. Macros defined by control characters or terminal-dependent special keys may be invoked by typing the control character or special key. Such invocations may occur in command, overstrike, or insert mode.

Named macros may be invoked only from command mode by means of the "," command. The "," command either prompts with "Macro name:" or beeps and informs you that there are no defined named macros. To the "Macro name:" prompt, you may type either the name of the macro to execute, or a "?" to see a list of defined named macro names.

## 16.3. Recursive Macros

MAINED allows a macro to invoke itself. Macros that invoke themselves are called recursive macros. A recursive macro executes until some command in its definition command sequence "fails". For example, the search and skip commands fail when they cannot find their targets. On some operating systems, it is also possible to terminate recursive macro execution with the <abort> key.

To define a recursive macro command, define the macro as described in Section 16.1, except that instead of typing a second "/" to terminate the macro definition, invoke the macro being defined (e.g., if defining CTRL-A, type CTRL-A; if defining a recursive named macro "FRED", type ",FRED<eol>"). MAINED displays:

```
    Macro definition terminated with recursive invocation
```

to indicate that the macro is recursively defined. MAINED does not execute the terminating macro command during definition of the macro. You must invoke the macro command after it is defined for it to be recursively executed.

As an example, consider a macro command, CTRL-A, that searches through a buffer and replaces each occurrence of "2" with "3". The macro definition is shown in Example 16.3-1.

| You type | Description |
|---|---|
| / | start macro definition |
| CTRL-A | macro command name |
| T | text search |
| 2<eol> | search target (prompted for on the message line) |
| O3 | overstrike with 3 |
| <ecm> | back to command mode |
| < | cursor left (in case next character is "2") |
| CTRL-A | repeat the above until search fails |

Example 16.3-1. Example of a Recursive Macro Definition

Recursive macros are appropriate for command sequences that involve searches, skips, or other commands that eventually fail. Beware of defining a recursive macro containing no commands that can fail; when executed, MAINED repeats the command sequence "forever", until either you type the <abort> key (not available on all operating system/terminal combinations), or you use some other operating-system-dependent means to terminate the MAINED session abnormally. In the latter case, you lose any changes made to buffers since the last time you saved them.

When repeating a recursive macro, the screen is not updated until the macro terminates or is aborted.

## 16.4. Macros and Editor Modes

When a macro is invoked, command mode is entered before the macro is executed so that the macro can have the same effect in all modes. The mode after macro execution depends on how the macro was defined.

If the last command in a macro's command sequence is <ecm>, the mode after macro invocation is the same as the mode immediately before the final <ecm> was typed. If the last command in a macro's command sequence is not <ecm>, the mode after macro invocation is the same as the mode immediately before macro termination (usually command mode, although use of the mode manipulation commands in Table 16.4-1 can result in a different mode). These rules imply that, in order to define a macro to enter command mode, the macro definition must consist of two <ecm>'s instead of one; the second <ecm> is discarded.

Whenever a macro is invoked, the current mode is saved in the editor variable "savedMode", and the current mode is set to command mode. The commands shown in Table 16.4-1 allow you to manipulate the mode. The mode stack mentioned is a circular stack 100 deep.

| Command | Effect on Mode |
|---|---|
| CM | push "savedMode" onto the mode stack |
| RM | pop the mode stack into the current mode |
| .RM | set the current mode to the top of the mode stack (the mode stack is not popped) |
| QRM | set the current mode to "savedMode" |

Table 16.4-1. Commands Used to Manipulate the Mode

Using the mode manipulation commands, you can define a macro that saves the mode upon entry ("CM") and restores it upon exit ("RM"), thus leaving the mode unchanged after the macro has been invoked and executed. "QRM" is a simple form for macros that do not themselves invoke macros. Such macros need not push the mode on the stack, since it is available in "savedMode"; they can restore the mode without doing any stack manipulation.

Table C-1 shows a use of mode manipulation commands.

## 16.5. Special Macro "INITIALIZE"

If your "eparms" file contains a named macro called "INITIALIZE", MAINED executes it as soon as you enter the first buffer in a MAINED session. The "INITIALIZE" macro allows you to set up your MAINED environment automatically. For example, if you would like tabs always to be every fourth column (instead of every eight, as by default), do as in Example 16.5-1. The next time "eparms" is updated, it will contain this macro definition. Subsequent invocations of MAINED will start by executing this macro, thereby setting your tab stops.

```
You type              Description
/                     start macro definition
initialize<eol>       for macro named "INITIALIZE"
Q<tab>5<eol>          tab every fourth column
/                     terminate macro definition
```

Example 16.5-1. Setting Tabs to Be Every Fourth Column

# 17. Arithmetic Functions

Several commands are provided for doing arithmetic calculations while in MAINED. These commands use "accumulators" A0 through A99. Internally, the arithmetic is done using the MAINSAIL long real data type.

The commands are summarized in Table 17-1, where An is accumulator number n, and "An := An <op> v" means to compute "An <op> v" and assign the result to An. If n is omitted, "1" is used. Thus, ".+1<eol>" increments the accumulator A1.

| Command | Action | |
|---|---|---|
| `n.+v<eol>` | `An := An + v` | |
| `n.-v<eol>` | `An := An - v` | |
| `-n.-v<eol>` | `An := v - An` | |
| `n.*v<eol>` | `An := An * v` | |
| `n./v<eol>` | `An := An / v` | |
| `-n./v<eol>` | `An := v / An` | |
| `n.^v<eol>` | `An := An ^ v` | `(raise An to the power v)` |
| `-n.^v<eol>` | `An := v ^ An` | `(raise v to the power in An)` |
| `n._v<eol>` | `An := v` | `(and set An's format to v's)` |
| `n.=` | `Display value of accumulator n` | |
| `Q.=` | `Display values of all active accumulators` | |

Table 17-1. Commands to Perform Arithmetic

Each of the arithmetic commands displays the new value of the affected accumulator and also puts the value displayed into the word delete buffer, so that you can recall it with the "RW" or ".RW" command. If you wish to recall the value in an accumulator, but do not know whether the value is on top of the word delete buffer, use "n.=" to display the accumulator and store its value on top, then "RW" to recall it.

MAINED prompts on the message line for v. Respond with either "v<eol>" or just <eol>. In the former case, the value v is to be used in the operation; in the latter, the word at the cursor is interpreted as a number, the value of which is used for v, and the cursor is positioned at the beginning of the next word as if a ")" command had been issued. If v is not properly formatted, the command fails and the bell rings (this terminates a macro call).

If several numbers appear on a line separated by blanks and tabs, they may be used on successive arithmetic commands in which <eol> is specified for v.

v can specify an accumulator or a constant. An accumulator is specified as "An" or "an" where n is the accumulator number. Thus, operations can be carried out among accumulators. For example, "1.+A2<eol>" performs the operation "A1 := A1 + A2".

If v is a constant, it may consist of the following components in the order listed (each component is optional, although at least one component must be present):

- A "-" to signify a negative value.
- A dollar sign, "$".
- An integer part. Commas may be used at three-digit intervals in the integer part.
- A fractional part: a decimal point optionally followed by fractional digits.
- An exponent part, in standard MAINSAIL format "E<sign><exponent>", where:
    - "e" can be used in place of "E"
    - <sign> is an optional "+" or "-"
    - <exponent> is one or more decimal digits for the power-of-ten exponent

As an alternative to an initial "-", a constant can be enclosed in deficit brackets, "<" and ">", e.g., "<1.20>". This form is often used in financial calculations to show a negative cash value.

Examples of valid constant forms are shown in Example 17-2. The numbers must be within the range supported for long real values on your computer. The "MAINSAIL Language Manual" gives guaranteed minimum ranges.

MAINED associates format information with each accumulator to remember whether the accumulator value is to be displayed with:

- Deficit brackets rather than a minus sign, if negative;
- A dollar sign;
- Commas at three-digit intervals in the integer part;
- A decimal point and number of fractional digits;
- An exponent.

```
1
-5
<5>
1.23
-1.23
$1
$1,234.56
-$1234.56
<$1,234,567.89>
123.456E12
.456E+18
```

Example 17-2. Valid Numeric Forms for Arithmetic Commands

Each time a command that modifies An is performed, the format information for An is updated according to v. The format information for each accumulator is cumulative. Thus, once a dollar sign is seen in a command such as "n.+$1.20<eol>", An is displayed with a "$" until its format is reset by means of the "._" command. On each command that modifies An, the number of fractional digits for An is set to the maximum of the current number of fractional digits for An and the number of fractional digits in v.

"n._v<eol>" resets the format information for An according to v. For example, if An is to be used to sum numbers, then you could type "n._<$0,000.00><eol>" to clear An and also set the display format for An to use deficit brackets, a dollar sign, commas, and two fractional digits with no exponent.

Macros using the arithmetic commands can be devised to add rows and columns of numbers.

# 18. Setting and Clearing Options

The ".O" command is used to set editor options; the "-.O" command clears options. See Table 18-1.

| Command | Action: |
|---|---|
| .O | set editor option |
| -.O | clear editor option |
| .O? | show settings of all options |

Table 18-1. Editor Option Commands

These commands prompt for the name of an option to be set or cleared, respectively. Only a unique prefix of an option need be typed to select the option. Case is ignored. If you type "?" in response to the "Option:" prompt, MAINED displays a summary of available options and their current settings. Currently available options are shown in Figure 18-2.

```
stickyTabs
readOnly
globalLeftBorder
globalRightBorder
localLeftBorder
localRightBorder
wordWrap
proportionalWindowsMode
insertModeDefault
overstrikeModeDefault
lowerCaseMode
upperCaseMode
```

Figure 18-2. Options

## 18.1. stickyTabs Option

The "stickyTabs" option, when set, attempts to preserve the position of text preceded by tab(s). By default, stickyTabs is set.

In stickyTabs mode, when non-tab characters are inserted or deleted, MAINED checks to see whether there are any tab characters to the right of the cursor. If there is more than one tab character (in the case of an insertion) or at least one tab character (in the case of a deletion), MAINED deletes or inserts one tab character for each character inserted or deleted, respectively, so as to preserve the position of the text that follows the tab character(s). If there is only one tab character, it is not deleted when a character is inserted; this prevents the text that follows the tab character from being merged with the previous text.

If a tab character is inserted or deleted, all tab characters immediately following the current cursor position are skipped before searching for tab characters to the right. Otherwise, the same algorithm is used as for the insertion or deletion of non-tab characters. See Example 18.1-1.

```
The positions of tab characters are shown in this example
by a period ("."). The cursor is shown as an underscore
("_"), and is on a tab character in both lines shown.

If a line of text looks like:

    abc.._..def.....ghi

then typing a tab while in insert mode results in:

    abc....._..def..ghi

preserving the position of "ghi".
```

Example 18.1-1. How the stickyTabs Option Affects the Insertion of Tab Characters

When stickyTabs mode is not set, tabs act like the appropriate number of spaces.

## 18.2. readOnly Option

When the "readOnly" option is set for an edit buffer, MAINED does not allow you to change the buffer. If you attempt to do so, MAINED rings the terminal bell. By default, readOnly is not set.

readOnly applies to individual buffers rather than to an entire edit session. To change the value of the readOnly option for a particular buffer, you must move the cursor into the buffer and issue the appropriate options command.

## 18.3. Border Options

The "borders" are the colons usually displayed at each side of the screen. The globalLeftBorder and globalRightBorder options control whether or not the left and right borders, respectively, are displayed when a new buffer is created (globalLeftBorder/globalRightBorder set means that the border is displayed). globalLeftBorder and globalRightBorder can be independently set and cleared. The default is both globalLeftBorder and globalRightBorder set.

The localLeftBorder and localRightBorder options control whether or not the left and right borders, respectively, are displayed for the current buffer. localLeftBorder and localRightBorder can be independently set and cleared. The default is the current setting of the corresponding global option.

When the globalLeftBorder or globalRightBorder option is set or cleared, the corresponding local option is set or cleared, respectively, in every buffer. Setting or clearing the localLeftBorder or localRightBorder option does not affect the globalLeftBorder or globalRightBorder option.

## 18.4. wordWrap Option

The wordWrap option was introduced to make it unnecessary to have to type <eol> when entering new text. The default is wordWrap clear. When the wordWrap option is set, the current line is automatically broken at the last word when a line extends exactly to the right margin and a new character is inserted or overstruck at the right margin. When inserting or deleting in existing text, use the ".J" command to refill it.

## 18.5. proportionalWindowsMode Option

When the proportionalWindowsMode option is set, all unanchored windows always occupy 1/mth of the screen, where m is the number of visible unanchored windows. The minimum number of rows in each window (not including the status line) is set with the "..{n}B" and "..{n}F" commands (the default is 1). The default is proportionalWindowsMode clear. Note that the "..B" command can be used to make all unanchored windows proportional when the proportionalWindowsMode option is clear. It does not change the setting of the proportionalWindowsMode option.

## 18.6. insertModeDefault and overstrikeModeDefault Options

insertModeDefault and overstrikeModeDefault are options that set the default mode to insert or overstrike, respectively. When insertModeDefault is set, MAINED is always in insert mode. In order to issue a MAINED command, you must type <ecm>, followed by the command. The command is executed, and the mode immediately returns to insert mode. overstrikeModeDefault has a similar effect, except that overstrike mode is the default.

## 18.7. Case Options

When upperCase mode is set, all inserted and overstruck characters are converted to uppercase (like a shift lock). When lowerCase mode is set, all inserted and overstruck characters are converted to lowercase (like a lowercase shift lock).

# 19. Using MAINED to Invoke Other MAINSAIL Modules

MAINED supports interaction with other MAINSAIL modules, making it an integral component of the MAINSAIL program development system. If you are using MAINED only as a text editor, an understanding of this feature is not necessary.

## 19.1. Executing a MAINSAIL Module

Any MAINSAIL module can be executed from MAINED. The command "QEm" executes a module, where m is either the name of the module or the name of the file that contains the module. MAINED prompts for m on the message line. Terminal I/O for the module is directed to the buffer "CMDLOG". CMDLOG is a buffer that can be altered by normal MAINED commands. It differs from other buffers in that:

- It is used by MAINED and other MAINSAIL modules to display and prompt for multi-line information.
- When you are asked whether to save buffers, you are not asked about CMDLOG, on the assumption that you do not wish to save it. If you need to save the contents of CMDLOG, rename CMDLOG with the "Q.B" command. A new CMDLOG will be created later if needed.
- Any output to the file "TTY", such as a ttyWrite or a write to logFile, is appended to the end of CMDLOG.
- Whenever input from the terminal is requested, as by a ttyRead or a read from cmdFile, the cursor is moved to the end of CMDLOG, insert mode is entered, and you are given control of MAINED. Typing <eol> on the last line of CMDLOG causes MAINED to treat your input as a line read from the terminal.
- On a terminal interaction that takes place in CMDLOG, MAINED positions the cursor at the end of the buffer if CMDLOG exists and its end is visible on the screen. If CMDLOG does not exist, it is automatically created. If a window containing CMDLOG is not visible on the screen, MAINED creates a new window for it occupying the top half of the screen, as if a "-.BCMDLOG<eol>" command had been performed. If CMDLOG appears in a window but the end of the buffer is not visible, MAINED scrolls the buffer as necessary to bring the end into view.
- The "H" command does not work in CMDLOG.

The use of CMDLOG is invisible to user modules. Thus, modules do not have to be modified to take advantage of MAINED's terminal redirection feature. CMDLOG can be used to keep a log of a display-oriented MAINSAIL session. This log can be saved on a file by renaming the buffer CMDLOG, then saving it with the "F" command.

If you create a buffer named CMDLOG, subsequent output from MAINSAIL programs appears in that buffer, regardless of how the buffer was originally created.

## 19.2. Escaping to Caller

When a "QE" command has been performed and the MAINSAIL module so executing has been suspended (presumably by typing <ecm> in CMDLOG while the module is waiting for input), the "E" command is used to return to the suspended module. If the module has a special display-oriented interface, MAINED sets the mode letter to "E" (Escape mode). MAINED commands are not available in escape mode. If the module performs standard terminal I/O through CMDLOG, MAINED positions the cursor on the last line of CMDLOG and enters insert mode to allow you to respond to the suspended module's prompt. If the prompt to which you typed <ecm> to suspend the module is no longer present on the last line of CMDLOG, MAINED rewrites it.

If there is no suspended module, MAINED rejects the "E" command with the message "Nothing to escape to".

The ".E" command dislays the name of the module most recently invoked and still pending. "Q.E" shows all pending invoked modules (most recently to least recently invoked). See Table 19.2-1.

| Command | Effect |
|---|---|
| E | Escape to caller, if any |
| QEs | Execute module s (dispose-bind-unbind) |
| .E | Show name of currently executing module |
| Q.E | Show names of all executing modules |

Table 19.2-1. Commands to Invoke Modules

Any amount of editing activity can take place prior to typing the <eol> at the end of CMDLOG, which allows the suspended module to resume control. You can view and edit any number of other files or buffers before returning to CMDLOG and typing in the response. For example, you may want to view another file to help decide what response to type into CMDLOG. You can even execute other modules, by means of the "QEm" command, before responding.

When you return control to a suspended module by typing <eol> on the last line of CMDLOG, MAINED verifies that the prompt to which you typed <ecm> still appears as the first part of that last line. If not, MAINED reprompts. If so, MAINED takes the remainder of the line as the line of terminal input.

The "-QF" editor command, if available, may be used to exit MAINSAIL even while MAINED is awaiting an <eol> from the last line of CMDLOG. As always, it prompts to save each altered buffer. Use of the "QF" command in this situation, however, raises the exception $abortProgramExcpt, which usually aborts the executing program. If you wish to exit MAINED, you may have to type "QF" more than once.

## 19.3. Sample Invocation of the MAINSAIL Compiler from MAINED

This example shows the invocation of the MAINSAIL compiler from MAINED on a screen 58 characters wide by 11 characters high. A caret ("^") is shown beneath the character at the cursor position. You may find it instructive to perform the steps shown, making the changes required by a larger screen.

The MAINSAIL compiler is described in the "MAINSAIL Compiler User's Guide".

Assume the initial screen configuration is as shown in Example 19.3-1. Type "QEcompil<eol>"; the resulting configuration is shown in Example 19.3-2. The mode character in CMDLOG is set to "I" because the module COMPIL is waiting for input to the "compile (? for help):" prompt. There is no mode character in the buffer POEM because the cursor is not located there.

```
----P.1-----L.1-----C----POEM-----------------------------
:                                                        :
:Nvnc est bibendvm, nvnc pede libero                     :
 ^
:Pvlsanda tellvs, nvnc Saliaribvs                        :
:Ornare pvlvinar deorvm                                  :
: Tempvs erat dapibvs, sodales.                          :
:                                                        :
:                        -- Horativs                     :
:                                                        :
E
```

Example 19.3-1. Initial Screen Configuration

```
Module or file name: compil\
----P.1-----L.1-----I----CMDLOG---------------------------
:Copyright (c) 1984, 1985, 1986, 1987, 1988, and 1989 by *
: Menlo Park, California, USA.                           :
:                                                        :
:compile (? for help):                                   :
                        ^
----P.1-----L.1----------POEM-----------------------------
:                                                        :
:Nvnc est bibendvm, nvnc pede libero                     :
:Pvlsanda tellvs, nvnc Saliaribvs                        :
:Ornare pvlvinar deorvm                                  :
```

Example 19.3-2. After Issuing the Command "QEcompil<eol>"

Assume there exist two files, one called "poem" with the contents shown in the POEM buffer in the example, and one called "poet", which is a short MAINSAIL program. If you type "poem<eol>" instead of "poet<eol>" to the compiler prompt, the compiler attempts to compile the file "poem" and gets an error, as shown in Example 19.3-3.

```
Module or file name: compil\
----P.1-----L.8-----I----CMDLOG---------------------------
:ERROR: file poem page 1 line 2 at ##                    :
:Nvnc## est bibendvm, nvnc pede libero                   :
:Expected BEGIN                                          :
:Error response:                                         :
                  ^
----P.1-----L.1----------POEM-----------------------------
:                                                        :
:Nvnc est bibendvm, nvnc pede libero                     :
:Pvlsanda tellvs, nvnc Saliaribvs                        :
:Ornare pvlvinar deorvm                                  :
```

Example 19.3-3. After Typing "poem<eol>"

In Example 19.3-3, the compiler is prompting for an error response. Part of the compiler's output to CMDLOG has been scrolled out of the CMDLOG window. It is possible at this point to enter command mode and view more of CMDLOG by typing "<ecm>QY-4W", with the result shown in Example 19.3-4. The "QY" command makes CMDLOG occupy the entire screen, so the POEM buffer is no longer visible.

```
Module or file name: compil\
----P.1-----L.1-----C----CMDLOG---------------------------
: intlib(<system intlib file name>)>sys-xxx             :
:                                                       :
:poem 1                                                 :
:                                                       :
:ERROR: file poem page 1 line 2 at ##                   :
:Nvnc## est bibendvm, nvnc pede libero                  :
:Expected BEGIN                                         :
:Error response:                                        :
                  ^
E
```

Example 19.3-4. After Typing "<ecm>QY1G" in CMDLOG

If you now wish to take a look at the file "poet" (to verify that it is the file you really wanted to compile), issue the command ".Fpoet<eol><eol>", with the result shown in Example 19.3-5. The second <eol> is required to answer the "Initial page.line" prompt.

```
Initial page.line for poet (1.1): \
----P.1-----L.1-----C----POET-----------------------------
:BEGIN "poet"                                           :
 ^
:                                                       :
:INITIAL PROCEDURE;                                     :
:IF confirm("Would you like me to write a poem?") THEN  :
:    write(logFile,"Sorry, my Muse is on a break." & eol):
:EL  write(logFile,"OK, I won't." & eol);               :
:                                                       :
:END "poet"                                             :
E
```

Example 19.3-5. Examining the File "poet" with ".Fpoet<eol><eol>"

You may now resume the suspended execution of COMPIL by means of the "E" command. Since CMDLOG is not present on the screen, it is placed in a new window, as shown in Example 19.3-6.

```
Escape with <EOL> on last line
----P.1-----L.11----I----CMDLOG----------------------------
:Error Response:                                          :
                 ^
E
E
E
----P.1-----L.1----------POET------------------------------
:BEGIN "poet"                                             :
:                                                         :
:INITIAL PROCEDURE;                                       :
:IF confirm("Would you like me to write a poem?") THEN    :
```

Example 19.3-6. After Returning to COMPIL with the "E" Command

First, enlarge the CMDLOG window by typing "<ecm>Q+2Y". Then type "I" to return to insert mode. Then type "a c<eol>" (an abbreviation for "MAINSAIL compiler: Abort compilation") to terminate the erroneous compilation. See Example 19.3-7.

```
Escape with <EOL> on last line
----P.1-----L.11----I----CMDLOG----------------------------
:Error Response: a c                                      :
:                                                         :
:Objmod not generated                                     :
:Intmod not stored                                        :
:                                                         :
:compile (? for help):                                    :
                       ^
----P.1-----L.1----------POET------------------------------
:BEGIN "poet"                                             :
:                                                         :
```

Example 19.3-7. Terminating the Compilation with "a c<eol>"

Type in the correct file name, "poet<eol>". The result is shown in Example 19.3-8.

```
Escape with <EOL> on last line
----P.1-----L.17----I----CMDLOG---------------------------
:poet 1                                                  :
:                                                        :
:Objmod for POET stored on poet-xxx.obj                  :
:Intmod for POET not stored                              :
:                                                        :
:compile (? for help):                                   :
                      ^
----P.1-----L.1----------POET-----------------------------
:BEGIN "poet"                                            :
:                                                        :
```

Example 19.3-8. Compiling the Correct File

Finally, type <eol> to the compiler's prompt to end the compiler session. The result is shown in Example 19.3-9. The CMDLOG mode character becomes "C", since the module is no longer waiting for input. Typing the "E" command at this point produces the message "Nothing to escape to", since there is no longer a pending compiler prompt.

```
Nothing to escape to
----P.1-----L.18----C----CMDLOG---------------------------
:                                                        :
:Objmod for POET stored on poet-xxx.obj                  :
:Intmod for POET not stored                              :
:                                                        :
:compile (? for help):                                   :
:                                                        :
 ^
----P.1-----L.1----------POET-----------------------------
:BEGIN "poet"                                            :
:                                                        :
```

Example 19.3-9. After Ending the Compiler Session

## 19.4. Simultaneous File Access Caveat

While a module is suspended, it is possible to attempt to edit a file that the module currently has open. This can lead to problems on operating systems that do not support file locking or multiple versions of files.

For example, suppose that while running the compiler from MAINED with an input file "foo", a syntax error has been reported and the "Error response:" prompt displayed. At this point, you may type <ecm> and edit a buffer associated with the file "foo". If you try to save the buffer before the compiler has gotten back to its "compile (? for help):" prompt, one of three things may happen:

1. If the operating system supports file versions, it creates a new version of "foo". The version of the file that the compiler currently has open is not affected.

2. If the operating system locks open files against write access, but does not support file versions, it may give an error saying that you cannot replace "foo" while it is open.

3. The operating system may replace the file "foo" without notice, giving data from the new version to the compiler the next time it reads from "foo". This usually produces a series of confusing error messages. The consequences of continuing the compilation in this case are unpredictable.

# Appendix A. MAINED Command Summary

The descriptions of commands containing <bs>, <tab>, <lf>, and <eol> presume that these keys have not been redefined as macro commands by the user. See Chapter 16 for a description of macro commands.

Command Mode

```
nA      do last command again, count = n, original modifier
QA      do last command again, original count and modifier
+nA     do last command again, count = n, "+" modifier
Q+A     do last command again, original count, "+" modifier
-nA     do last command again, count = n, "-" modifier
-QA     do last command again, original count, "-" modifier
("-" direction is towards beginning of file)
("+" direction is towards end of file)
.A      anchor current window
+.A     anchor at bottom
-.A     anchor at top
n.A     anchor, change size to n rows
+n.A    anchor at bottom, change size to n rows
-n.A    anchor at top, change size to n rows
..A     unanchor current window
Q..A    unanchor all windows
Q+..A   unanchor all windows at bottom of screen
Q-..A   unanchor all windows at top of screen
```

| | |
|---|---|
| B | break line, remove spaces |
| nB | break line, indent n spaces |
| QB | break line, leave spaces |
| -B | break line, remove spaces, leave cursor |
| -nB | break line, indent n spaces, leave cursor |
| -QB | break line, leave spaces, leave cursor |
| .Bs | edit buffer s, use current window if not on screen |
| +.Bs | edit buffer s, new window at bottom if not on screen |
| n.Bs | same as ".Bs", except n-row window |
| +n.Bs | same as "+.Bs", except n-row window |
| -.Bs | edit buffer s, new window at top if not on screen |
| -n.Bs | same as "-.Bs", except n-row window |
| --{n}.Bs | overlay (n-row) window at top |
| ++{n}.Bs | overlay (n-row) window at bottom |
| ..{n}Bs | edit s, making window 1/mth of screen, where m is the number of windows; but no window is allowed to be smaller than n lines |
| Q.Bs | change bufferName of current buffer to s |
| Q..Bs | change command front end to s |
| +Q..Bs | change command and view front ends to s |
| -Q..Bs | kill front end s |

| | |
|---|---|
| nC[C\|W\|L\|P] | copy n objects at and after |
| QC[C\|W\|L\|P] | copy all objects at and after |
| -nC[C\|W\|L\|P] | copy n objects before |
| -QC[C\|W\|L\|P] | copy all objects before |
| n.C | center n lines at and after |
| Q.C | center all lines at and after |
| CM | push savedMode onto mode stack |

| | |
|---|---|
| nD[C\|W\|L\|P] | delete n objects at and after |
| QD[C\|W\|L\|P] | delete all objects at and after |
| -nD[C\|W\|L\|P] | delete n objects before |
| -QD[C\|W\|L\|P] | delete all objects before |

(".D" copies text into delete buffer, but does not delete it;
"..D" deletes text, but does not copy it into the delte buffer)

| | |
|---|---|
| E | escape to caller, if any |
| QEs | execute module s (dispose-bind-unbind) |
| .E | show name of currently executing module |
| Q.E | show names of all executing modules |

| | |
|---|---|
| F | prompt to save altered buffers, then continue |
| +F | change autoSaveLimit (0 means no autoSave reminder) |
| QF | If a program invoked with "QE" is running, raise the exception $abortProgramExcpt; otherwise, exit from MAINEDIT, continuation not allowed |
| -QF | prompt to save altered buffers, then pause |
| .Fs | edit file s, use current window if not on screen |
| +.Fs | edit file s, new window at bottom if not on screen |
| n.Fs | same as ".Fs", except n-row window |
| +n.Fs | same as "+.Fs", except n-row window |
| -.Fs | edit file s, new window at top if not on screen |
| -n.Fs | same as "-.Fs", except n-row window |
| --{n}.Fs | overlay (n-row) window at top |
| ++{n}.Fs | overlay (n-row) window at bottom |
| ..{n}Fs | edit s, making window 1/mth of screen, where m is the number of windows; but no window is allowed to be smaller than n lines |
| Q.Fs | change fileName of current buffer to s |

| | |
|---|---|
| G | go to first line of next page |
| -G | go to first line of previous page |
| p.lG | go to page p, line l |
| .G | go to first line of current page |
| .lG | go to line l of current page |
| pG | go to first line of page p |
| +nG | first line of current page + n |
| -nG | first line of current page - n |

(start with "Q" to set mark ("@" command) before going)

| | |
|---|---|
| nH | undo previous n changes |
| QH | undo all changes on current line |
| -nH | redo next n changes |
| -QH | redo all changes on current line |

| | |
|---|---|
| I | enter insert mode |
| 1IB | insert a buffer (name is asked) |
| nICc | insert n c's (n required) |
| QICc | insert c's to right margin |
| 1IF | insert a file (name is asked) |
| nIL | insert n blank lines (n required) |
| QIL | insert blank lines to end of window |
| .I | insert blank line, enter insert mode |
| n.I | insert blank line, indent n spaces, enter insert mode |

| | |
|---|---|
| nJ | join next to current line, n separating spaces |
| QJ | join next to current line, leave spaces |
| -nJ | join current to previous line, n separating spaces |
| -QJ | join current to previous line, leave spaces |
| .J | fill current paragraph to right margin of window |
| n.J | fill n lines |
| +.J | fill and justify to right margin of window |
| -.J | fill starting at cursor column |
| Q.J | fill all remaining paragraphs in buffer |
| .mJ | fill to right margin (justify) n column m |
| nQ.J | fill next n paragraphs |

(All modifiers may be combined; i.e., nQ+-.m.J means fill next n paragraphs from cursor column, justifying to column m).

| | |
|---|---|
| nK | delete (kill) n characters at and to right |
| QK | delete all characters at and to right |
| -nK | delete n characters to left |
| -QK | delete all characters to left |
| .K | prompt to kill each buffer (prompts to save) |
| Q.K | kill one buffer (prompts for name) |
| ..K | kill a character without copying into delete buffer |

| | |
|---|---|
| nL[C\|W\|L] | make n objects lower case |
| QL[C\|W\|L] | make all objects lower case |

| | |
|---|---|
| nM[C\|W\|L\|P] | move current object n further |
| QM[C\|W\|L\|P] | move current object to end |
| -nM[C\|W\|L\|P] | move current object n before |
| -QM[C\|W\|L\|P] | move current object to start |
| .M[C\|W\|L] | mark the appropriate delete buffer |

| | |
|---|---|
| 0N | refresh message line |
| nN | refresh n lines at and below in current window |
| -nN | refresh n lines above in current window |
| QN | refresh entire screen |
| Q.N | refresh current window |

| | |
|---|---|
| O | enter overstrike mode |
| nOCc | overstrike n c's (n required) |
| QOCc | overstrike c's to right margin |
| .O | set editor option |
| -.O | clear editor option |
| .O? | show option settings |

| | |
|---|---|
| P | insert page mark above |

```
Q      emphasize the command

R[C|W|L|P]       recall and insert group of objects
nR[C|W|L|P]      recall and insert n objects
QR[C|W|L|P]      recall and insert all objects to mark
(".R" means leave in delete buffer)
RM               pop mode stack into curMode
.RM              set curMode to top of mode stack (not popped)
QRM              set curMode to savedMode

nSc     skip right to nth occurrence of character "c"
+Sc     skip over c's
-nSc    skip left to nth occurrence of character "c"
-+Sc    skip left over c's
QnSc    skip down to nth "c"-line
Q+Sc    skip down to next line not starting with c
-QnSc   skip up to nth "c"-line
Q+Sc    skip up to next line not starting with c
(a "c"-line is a line with first visible char equal to "c".
A <sp>-line is one with no visible characters.)

Ts<eol>      search right and all lines down for s ...
T<eol>       search right and all lines down for last target(s)
nTs<eol>     search right and n-1 lines down for s ...
QTr<eol>s...<eol><eol>
             search right and all lines down for r or s or ...
QnTr<eol>s...<eol><eol>
             search right and n-1 lines down for r or s or ...
(-T searches left and up)
(qualifying with "+" wraps around buffer beginning or end)
(qualifying with "QQ" makes into a line search)
("{-}.T" is an "identifier" search, i.e. the target cannot be
bordered by an alphabetic or digit)

U       same as "L", except convert to upper case

V       give character position, prompt for new one
Q..Vs   change view front end to s

W       scroll up 4/5 of a window
nW      scroll up n lines
QW      scroll up all lines
-W      scroll down 4/5 of a window
-nW     scroll down n lines
-QW     scroll down all lines
n.W     move current line to line n from top of window
-n.W    move current line to line n from bottom of window
```

| | |
|---|---|
| nX | move to column n of window (x-coordinate) |
| -nX | move to window width - n + 1 |
| QnX | put right margin at column n (from line origin) |
| -QnX | put left margin at column n (from line origin) |
| | |
| nY | move to row n of window (y-coordinate) |
| -nY | move to row n of window, count from bottom up |
| QY | set current window to maximum size |
| QnY | set number of window rows to n (n = 0 kills window) |
| +QY | expand window to bottom of screen |
| +QnY | expand window n rows |
| -QY | synonym for Q0Y |
| -QnY | contract window n rows |
| n.Y | move cursor to nth next window on screen |
| -n.Y | move cursor to nth previous window on screen |
| Q.Y | move cursor to bottommost window on screen |
| -Q.Y | move cursor to topmost window on screen |
| | |
| Z | same as "S", except delete skipped objects |

(".Z" means do not delete, but put into delete buffer;
"..Z" means delete, but do not put into delete buffer)

| | |
|---|---|
| n<bs> | move left n columns |
| Q<bs> | move to left margin |
| | |
| n<tab> | move cursor to nth next tab stop |
| -n<tab> | move cursor to nth previous tab stop |
| Q<tab> | set tab stops |
| | |
| n<lf> | move down n rows |
| Q<lf> | move to last row |
| | |
| <abort> | abort current command, enter command mode |
| | |
| n<eol> | move to left margin of nth next line |
| -n<eol> | move to left margin of nth previous line |
| | |
| n<sp> | move right n columns |
| Q<sp> | move to last column |

(equivalent to ">")

| | |
|---|---|
| 'n | insert character with decimal code "n" |
| 'Bn | insert character with binary code "n" |
| 'Hn | insert character with hexadecimal code "n" |
| 'On | insert character with octal code "n" |

| | |
|---|---|
| n( | move left n words |
| n.( | move to one past end of nth word to left |
| Q( | move to first visible character of line |
| n) | move right n words |
| n.) | move to one past end of (n-1)st word to right |
| Q) | move to after end of line |
| , | invoke named macro |
| /x.../ | define x to be ..., where x is a macro name (... is carried out as it is typed in) |
| n< | move left n columns |
| Q< | move to first column |
| = | show line info |
| Q= | show buffer info |
| +Q= | show buffer info with front end info |
| n> | move right n columns |
| Q> | move to last column |
| .@ | set mark to current location |
| @ | go to marked location |
| Q@ | set mark, go to previously marked location |
| n\ | move down n rows |
| Q\ | move to bottom row of window |
| n^ | move up n rows |
| Q^ | move to top row of window |
| n&lt;del&gt; | move left n columns |
| Q&lt;del&gt; | move to left margin |

|  |  |  |
|---|---|---|
| n.+v<eol> | An := An + v |  |
| n.-v<eol> | An := An - v |  |
| -n.-v<eol> | An := v - An |  |
| n.*v<eol> | An := An * v |  |
| n./v<eol> | An := An / v |  |
| -n./v<eol> | An := v / An |  |
| n.^v<eol> | An := An ^ v | (raise An to the power v) |
| -n.^v<eol> | An := v ^ An | (raise v to the power in An) |
| n._v<eol> | An := v | (and set An's format to v's) |
| n.= | Display value of accumulator n |  |
| Q.= | Display value of all active accumulators |  |

<u>Overstrike Mode</u>

|  |  |
|---|---|
| <bs>, <del> | move left 1 column, except end of previous line if at left margin |
| <tab> | overstrike spaces to next tab stop |
| <lf> | move down 1 row |
| <eol> | move to left margin of next line |

<u>Insert Mode</u>

|  |  |
|---|---|
| <bs> | move left 1 column |
| <tab> | insert spaces to next tab stop |
| <lf> | move down 1 row |
| <eol> | break line, move cursor to start of new line |
| <del> | delete character to left, except join current line to previous line if at left margin (like "-QJ" command) |

# MAINVI User's Guide

# 20. MAINVI Introduction

The MAINVI front end is an emulation of the text editor "vi" available on most UNIX systems. Unlike vi, MAINVI can be used on any system where MAINSAIL is available, not just UNIX; it can just as easily be run, e.g., on VAX/VMS.

The MAINVI front end is provided so that users of vi can immediately start using the XIDAK's MAINEDIT text editor with a minimum of training. The text editor provides multiple windows, multiple buffers, and display interfaces to important MAINSAIL tools, including MAINDEBUG, the MAINSAIL debugger. MAINVI is not intended to be an exact emulation of vi; the MAINVI command structures and key sequences are as similar as possible to vi's, but screen refresh is often different.

This document describes only the differences between MAINVI and vi; a standard reference should be consulted if you are unfamiliar with the original text editor. Most UNIX system documents include a description of vi.

# 21. Differences between MAINVI and vi

XIDAK has not been able to determine in advance which deviations from the behavior of vi will prove annoying to regular vi users and which will go unnoticed. The list given here is probably not complete. An attempt has been made to order the list roughly with the most egregious differences first, but only the experience of individual users will determine which differences they find significant.

In many cases, screen update may differ somewhat in MAINVI, especially in ex mode (the mode entered by the "Q" command) and shell escape commands.

## 21.1. Initialization

MAINVI examines the UNIX "EXINIT" shell variable only on UNIX. On all operating systems, it attempts to read the file ".exrc" in the home directory. If an "eparms" file is present, MAINVI also remembers in it the last file edited and all abbreviations and macros from MAINVI sessions (the "DONOTUPDATEEPARMS" keyword may be used to suppress this remembered information).

## 21.2. Status Line

MAINVI uses one line of each window for a "status line", which includes the line number (preceded by "L#"), page number (preceded by "P."), and page-relative line number (preceded by "L.") of the line at the top of the window, a letter representing the current mode ("C" for command, "I" for input), and the name of the buffer. This line is required to separate the current window from other windows that may also be on the screen (a feature vi does not provide). Prompts appear on the top line of the screen instead of the bottom.

## 21.3. Tabs

Tabs are treated very differently in MAINVI and vi. MAINEDIT "expands" tabs to "tabified blanks", as described in Section 1.11. It is possible to position the cursor at each position within a tab in a MAINVI buffer, as if the tab were completely expanded to blanks; i.e., the cursor-left and cursor-right commands do not jump to the last position of the tab as in vi. The editor does, however, remember which blanks originated as blank characters and which as tabs, so that when a file is written out, undisturbed tabs are written as tabs.

## 21.4. Lines Too Long to Fit on the Screen

Lines too long to fit on the screen are wrapped onto subsequent lines by vi. MAINVI displays a tilde in the last column of the overly long line and does not display the characters extending beyond the edge of the screen, so that there is always exactly one line on the screen for each line in the buffer. It is possible to scroll the screen left or right to see these invisible characters by issuing the ":leftcolumn" command.

## 21.5. Display of Control Characters

The "^X" notation is not used; an asterisk is displayed on most terminals for a non-printing character. The actual contents of a range of lines may be displayed with the ":list" command. In a non-ASCII character set, ":list" writes all characters not in the MAINSAIL character set as "\nnn", where nnn are octal digits. MAINVI's ":print" differs from ":list" in that it does no special processing of control characters.

## 21.6. Page Marks (CTRL-L in a File)

The page mark character (CTRL-L on ASCII systems) is always preceded by an end-of-line, and is always displayed on the screen as being followed by an end-of-line. The first, but not the second, end-of-line is actually present in the file when it is written. This is a property of the editor back end required by MAINEDIT, but a nuisance in MAINVI.

The "g" command may be used to move to the beginning of a page.

## 21.7. Deletions during Input Mode

vi does not immediately erase characters from the screen deleted by the backspace key in input mode, although these characters are erased when input mode is exited. MAINVI erases them each time the backspace key is pressed, so that the screen always reflects the buffer contents.

The "c" command does not put a dollar sign at the end of text to be deleted, but goes ahead and deletes it before insertion starts.

The characters used for deletions during insert mode are hardwired as CTRL-U (for line deletions), CTRL-W (for word deletions), and backspace or delete (for character deletions). They are not read from the current UNIX terminal mode information.

## 21.8. CTRL-L Command

CTRL-L is a screen refresh command in vi (a near synonym is CTRL-R). On most terminals, CTRL-L is the <abort> command, so CTRL-R should be used instead. The <abort> key is used in some situations where CTRL-C (or the UNIX interrupt character) would be used in vi. The regular interrupt character should be used to abort a shell command invoked from MAINVI.

## 21.9. Shell Commands

Commands that invoke the shell or another process work only on systems that have MAINSAIL STREAMS installed. Consult the "MAINSAIL STREAMS User's Guide" for details.

## 21.10. Delete Buffers and Text Recovery

Deletions always go into the numbered buffer ring (buffers are numbered from "1" to "9"); if a lettered buffer is specified, the deletion also goes into it. All deletions are saved except those made in input mode. The default buffer from which text is recovered is "1". Text is recovered in the order deleted, without extra line breaks inserted between character-oriented deletions.

## 21.11. Macros

Command mode macro left-hand sides are limited to a single keystroke (most keys that send escape sequences are considered a single keystroke). Input mode macros (those defined by ":map!") are usually limited to keys that generate a single character code.

## 21.12. Ignored Modes and Variables

The following ":set" variables are ignored:

- directory
- hardtabs
- lisp
- list

- open
- optimize
- redraw
- slowopen
- taglength
- term
- terse
- w300, w1200, w9600
- writeany

## 21.13. tabstop

The "tabstop" variable controls both the amount of space inserted by the <tab> command and the interpretation of tabs from the current file. At present, changing the "tabstop" variable affects only buffers created after the change, not the current buffer.

## 21.14. Undoing Commands

Undo (the "u" command or ":undo") undoes the same amount of work as MAINED's "H" command, which may differ in some situations from vi's undo. Entire macro calls are undone not as a unit, but as individual changes. "u" is its own inverse, as in vi. A more extensive undo facility is provided by the ":hundo" and ":hredo" commands.

## 21.15. Marks

Marks remember only the page, line, and character number; they do not hold onto the actual text the way vi marks do. The insertion of a page mark character before a mark, or of text before a mark on the same page, will change the mark position with respect to the text.

## 21.16. Quoting Input Characters

When responding to a prompt, CTRL-V may be used to escape characters that would normally terminate the input of text, and backslash to escape characters that would be treated specially by the command being given. On UNIX, the operating system may also intercept CTRL-V, so two CTRL-V's may be necessary to get the effect of one CTRL-V.

## 21.17. Tags

Tags are not supported and the commands dealing with them are not implemented.

## 21.18. "=" (Indent for LISP)

The "=" (indent for LISP) command is not implemented.

## 21.19. Crash Facilities

Facilities for recovering from crashes (":preserve", ":recover") are not implemented. ":write" should be used frequently to save the state of the current file, if necessary.

## 21.20. Other Unimplemented Commands

We have not been able to figure out what the ":~" command is supposed to do. The documentation we have is no help, and we have been unable to observe useful behavior produced by this command in experiments with vi. Any help on this would be appreciated.

## 21.21. File Argument List

The initial file argument list does not exist in MAINVI. Commands dealing with this list (e.g., ":next") are not implemented. Use the ":edit" and ":buffer" commands to edit other files in the same editing session.

## 21.22. Reading a File

MAINVI reads text from a file on an as-needed basis. The file therefore remains open until the end of the file is read.

## 21.23. Autoindent Space and CTRL-D in Input Mode

Autoindent space inserted at the beginning of a new line can be backspaced or deleted like normal text; vi requires the use of the CTRL-D command to remove this space. CTRL-D works anywhere on the line in input mode, not just at the beginning.

## 21.24. Warnings about Switching among Buffers

Warnings are not issued when switching among buffers or writing out files; the editor prompts for whether to write out unwritten files and replace existing files.

## 21.25. The Bell

vi sometimes ring the bell if a cursor movement command fails, but sometimes not. MAINVI always rings the bell.

## 21.26. Error Messages

Error messages often differ from vi's (although they should be at least as intelligible). Inverse video is never used in error messages.

## 21.27. Printing the Current Line in ex Mode

Flags governing printing ("p", "l", and "+") of the current line after a command is executed are ignored; MAINVI prints lines by default more frequently than vi, so the flags are not very useful.

# 22. Commands and Variables Added to MAINVI

Several non-standard commands and ":set" variables have been added to MAINVI to allow interaction with the rest of the MAINSAIL editor environment.

MAINVI is not case-sensitive with respect to line-oriented command or variable names (case is distinguished in single-character commands, of course). Command and variable names may be abbreviated to their shortest unique prefix.

## 22.1. Commands

### 22.1.1. "g": Moving to a Page Mark

Since the MAINSAIL compiler and other tools often refer to file positions in terms of pages, it is convenient to have a command that moves based on pages. The "g" command, with an argument, goes to the start of the page specified; with no argument, it moves to the start of the next page. The "g" command may be used as an object of the usual operators; e.g., "d3g" deletes to the start of page 3.

### 22.1.2. :buffer

The ":buffer" command moves to another editor buffer. The new buffer occupies a window in the top half of the screen (to make it occupy the whole screen, or otherwise change the height of the window, use the "z" command, or change the "window" variable). The syntax of the ":buffer" command follows that of the ":edit" command, except that a buffer name is specified instead of a file name. The buffer name associated with a file is generally the same as the file name, except that buffer names are case-insensitive.

For example, to switch to a buffer containing a file "foo.msl":

```
:buffer foo.msl<eol>
```

or, if "FOO.MSL" is the only the buffer name containing the substring "FOO":

```
:buffer foo<eol>
```

The buffer named "CMDLOG" is used for most program interaction.

A buffer need not have a file associated with it. When the ":buffer" command creates a new buffer, no file is associated with the buffer, by default. To create a buffer containing a file, use the ":edit" command, as in vi.

More than one buffer may be created into the same data. Since several buffers may be displayed on the screen simultaneously, this allows multiple windows into the same file. To create a buffer BAR with the same data as a buffer FOO, end the buffer name with a comma:

```
:buffer bar,<eol>
```

The editor then prompts for a front end (answer "vi") and a back end; answer the latter question with "=foo", meaning the same data as in the buffer FOO. Consult the "MAINEDIT User's Guide" for more information.

### 22.1.3. :hredo, :hundo

The ":hundo" command provides a more extensive facility for undoing commands than vi (the "H" comes from the fact that ":hundo" works the same way as the MAINEDIT "H" command). It accepts a numeric argument, the number of changes to be undone. ":hundo" is not its own inverse; subsequent :hundo's undo progressively more changes until the limit of saved changes (currently 300) is reached.

":hredo" redoes undone changes; it is the inverse of ":hundo". Like ":hundo", it accepts a numeric argument telling how many undone changes to redo.

For example:

```
:hundo 10<eol>
```

undoes the last ten changes.

All the undo commands, including "u" and "U", have the same granularity as MAINED's undo command. In some cases, this means they undo a different amount of work from real vi's.

### 22.1.4. :invoke

":invoke modName args" invokes a MAINSAIL module modName and passing the arguments args to it. ":invoke" with no argument returns control to the currently executing module, if any (module executions are stacked; only the one on top of the stack is active at any given time).

Interaction with line-oriented programs (e.g., the MAINSAIL compiler) appears in the buffer "CMDLOG". An end-of-line in input mode in "CMDLOG" causes the last line typed (minus prompt, if any) to be sent to the currently executing program. More details on the use of "CMDLOG" may be found in Chapter 19.

Some programs, e.g., MAINDEBUG, have a special display interface. Consult the user's guide for the program you are interested in for details. MAINDEBUG's interaction with the text editor is independent of the front end used; it can be used with either MAINEDIT or MAINVI.

### 22.1.5. :leftcolumn

The ":leftcolumn" command accepts a numeric argument that specifies which column of the buffer is to be displayed at the left of the current window (column numbers are one-origin). For example, to scroll the screen left by 20 lines:

```
                    :leftcolumn 21<eol>
```

This command is necessary to see the contents of lines longer than the width of an editor window, since MAINVI does not wrap lines on the display.

### 22.1.6. :lset, :set

":set" sets options globally (i.e., for all MAINVI buffers).

":lset" is used to set an option for just the current buffer. When option values are displayed, they include the name of the applicable buffer, since the global ":set" used for inquiry shows values in all MAINVI buffers.

#### 22.1.6.1. :reinit

A new command, ":reinit", has been added to reinitialize the terminal mode and redisplay the screen.

### 22.1.7. :unabbreviateall, :unmapall

The ":unabbreviateall" command may be used to forget all abbreviations (as set by the ":abbreviate" command); the ":unmapall" command to forget all macros (as set by the ":map" and ":map!" commands). These are useful if undesired macros have been restored from the "eparms" file.

## 22.2. Variables

Variables are maintained on a per-buffer basis, except as noted; changing a variable in one MAINVI buffer does not affect the value of the variable in any other buffer.

### 22.2.1. leftBorderChar, rightBorderChar (Numeric)

By default, MAINVI does not display "border" characters (MAINEDIT, by default, displays columns of colons on each side of an editor window). If MAINEDIT-style borders are desired, they may be specified by giving a non-zero character code as the value of leftBorderChar or rightBorderChar. For example, to set both margins to colons on an ASCII system:

```
:set leftborderchar=58 rightborderchar=58<eol>
```

(58 is the ASCII code for ":").

### 22.2.2. width (Numeric)

The "width" variable controls the displayed width of the current window. For example:

```
:set width=20<eol>
```

causes only the first 20 characters of each line to be displayed.

### 22.2.3. pageAndLinePrompt

If the option "pageAndLinePrompt" is set, MAINED-style initial page and line prompts are given when entering a new buffer. If not set, these prompts are suppressed. This also applies to any other front ends (e.g., MAINED) that may be used in the same editing session. By default, this option is not set.

### 22.2.4. replaceWithoutPrompt

If the option "replaceWithoutPrompt" is set, the prompt "Replace existing file ..." is suppressed for all files (this also applies to any other front ends (e.g., MAINED) that may be used in the same editing session). By default, this option is not set.

# 23. Known MAINVI Problems

Searches and word, paragraph, and section commands are unduly slow.

":shell" and related commands do not work on all systems, even if the MAINSAIL STREAMS package is installed. Not all operating systems provide the capabilities necessary to implement the commands. An error message is issued if you attempt to invoke an unavailable command.

# MEDT User's Guide

# 24. MEDT Introduction

The MEDT front end is an emulation of the text editor "EDT" included with Digital Equipment Corporation's VAX/VMS operating system. Unlike EDT, MEDT can be used on any system where MAINSAIL is available, not just VAX/VMS; it can just as easily be run, e.g., on UNIX.

The MEDT front end is provided so that users of EDT can immediately start using the XIDAK's MAINEDIT text editor with a minimum of training. The text editor provides multiple windows, multiple buffers, and display interfaces to important MAINSAIL tools, including MAINDEBUG, the MAINSAIL debugger. MEDT is not intended to be an exact emulation of EDT; the MEDT command structures and key sequences are as similar as possible to EDT's, but screen refresh is often different.

This document describes only the differences between MEDT and EDT; a standard reference should be consulted if you are unfamiliar with the original text editor. The standard reference on EDT is the "VAX EDT Reference Manual", DEC order number AA-Z300A-TE.

# 25. Differences between MEDT and EDT

XIDAK has not been able to determine in advance which deviations from the behavior of EDT will prove annoying to regular EDT users and which will go unnoticed. The list given here is probably not complete. An attempt has been made to order the list roughly with the most egregious differences first, but only the experience of individual users will determine which differences they find significant.

In many cases, screen update may differ somewhat in MEDT.

## 25.1. Terminals

DEC's EDT runs only on a small set of terminals: VT100, VT52, and LK201 terminals and compatible models. XIDAK's MEDT runs on any terminal for which an editor display module exists; however, default key mappings for keypad mode are provided only for the VT100 and compatible terminals (XIDAK does not have display modules for the VT52 or LK201 terminals). Nokeypad commands may be used on terminals for which keypad mappings are not supplied, and users may define their own kepypad mappings for any terminal.

## 25.2. Initialization Files

When MEDT is first invoked, it tries reading line mode commands from "sys$library:edtsys.edt", "edtini.edt", and a file name formed from the current terminal name followed by the extension ".edt" on the user's home directory (e.g., "vt100.edt" if VT100 is the current terminal). Keypad key names and mappings for the current terminal are remembered in the "eparms" file even if no initialization file is found.

To redirect the reading of the initialization files, it is sufficient to provide logical file names for the files that map to the desired file names. If you want to suppress reading a file entirely, map it to the name of a non-existent file. A sample MAINSAIL program that invokes the editor after setting up such a logical name for "edtini.edt" based on a command line argument is shown in Table 25.2-1.

## 25.3. EDT Line Mode

EDT line mode is not implemented. EDT line mode commands may be given as usual with the nokeypad mode "EXT" command, or the keypad COMMAND command on a VT100.

```
BEGIN "iEdt"

# invoke editor; command line argument is name of file to
# be read instead of "edtini.edt", or "-" if no file is to
# be read.

INITIAL PROCEDURE;
BEGIN
STRING s;
$getCommandLine(s);
enterLogicalName("edtini.edt",
    IF s = "-" THEN
        cvcs($nulChar) # or other unlikely name
    EL s);
$invokeModule("EDIT");
    # gives MEDT front end only if specified in eparms
END;

END "iEdt"
```

Table 25.2-1. Redirecting "edtini.edt"

The line mode "CHANGE" command doesn't really change from line mode. It does, however, switch buffers if a buffer is specified, and allows nokeypad commands to be appended to a line mode command.

The nokeypad "EX" command doesn't switch to line mode. It issues an error message and switches to nokeypad mode. If no definitions were found for the terminal's keypad mode, <ecm> is initially mapped to "EX." so that you can get out of keypad mode if you get into it accidentally.

## 25.4. Initial Mode

If the display module is not VT100, then the initial mode by default is nokeypad mode, unless a definition for "GOLD" is found, in which case it is assumed that a command to get back out of keypad mode has been defined. The editor initialization files may contain commands to override the default mode.

## 25.5. Display of Tabs and Control Characters

Tabs are expanded by MEDT to "tabified blanks". This means that moving the cursor across a tab character is the same as moving it across the equivalent number of blanks; the cursor pauses at each position on the screen, and tabified blanks may be deleted individually. There is no visual difference between tabs and blanks, but tabs that are read in from the input file and not modified during an edit session are written out again as tabs.

Control characters are displayed on most terminals with an asterisk ("*") character instead of the traditional "^X" notation. Other special characters are also displayed as asterisks.

## 25.6. Word Entity and Tabs

If the word entity includes a blank, tabified blanks are considered to be included as well.

## 25.7. Searches and Tabs

Blanks in a search string match both blanks and tabified blanks in the buffer, but tabs in a search string match only tabified blanks in the buffer.

## 25.8. End-of-Line Character

The end-of-line character stored in a file edited by MEDT is the MAINSAIL eol character, which varies from operating system to operating system. For example, on a system where eol is a linefeed (CTRL-J), the linefeed should appear in a search string containing an end-of-line; e.g., ";^JBEGIN" to search for a semicolon followed by an end-of-line followed by "BEGIN".

## 25.9. Message and Status Lines

The message line (on which messages are displayed and input is read) is the top line of the screen, not the bottom line of the screen as in EDT. Since an MEDT window may be only one of several windows on the screen, one line per window is occupied by a divider called the status line. It lists the current front end associated with the window, the name of the buffer in the window, and the absolute line number (preceded by "L#"), page number (preceded by "P."), and line number with respect to the start of the current page (preceded by "L.") of the line at the top of the window. If the cursor is in the window, a mode letter is displayed to indicate the MEDT mode: "N" for nokeypad, "K" for keypad, "I" for input, and "E" for escape (when a program has control of the window).

Nokeypad commands are not highlighted as they are echoed on the message line.

## 25.10. Borders and the Display of End of Buffer

"[EOB]" is not displayed at the end of a buffer. If borders are enabled (see Section 26.1), the left border of a window is displayed as "E" beyond the end of the window.

## 25.11. Keypad in Nokeypad Mode

The keypad keys in nokeypad mode do not revert to their numeric function; they continue to produce special codes. The numeric keys on the keyboard must be used to enter digits.

## 25.12. Select Range

The select range is not highlighted.

## 25.13. CTRL-C and CTRL-Z

The <abort> key, CTRL-L on most terminals, is used instead of CTRL-C to abort most long-running editor commands. <ecm> may be used instead of CTRL-Z to end insert mode (CTRL-Z also works on systems which do not intercept CTRL-Z).

## 25.14. Main Buffer Name

The first buffer entered in the editor is not called "MAIN"; the name chosen is the initial file name converted to upper case.

## 25.15. Buffer Name Formats

EDT buffer names may contain only alphanumeric characters. MEDT buffer names may contain any characters, including periods and spaces. Since periods and spaces are used to terminate buffer names in many commands, buffer names with periods and spaces in them should be quoted, e.g., use:

```
="foo.msl"
```

instead of:

```
=foo.msl
```

for a buffer named FOO.MSL. The single quote character may be used in place of the double quote character. If a quote character is included in a buffer name, it must be doubled, as in a MAINSAIL string constant.

## 25.16. Select Range

The select range anchor position is remembered as a line and character position. Inserting or deleting text before the selected position moves the anchor relative to the text.

A successful search sets the select range anchor to the other end of the string found.

## 25.17. Sequence Numbers

MEDT does not use line numbers in the same way EDT does. Line numbers are always specified as the number of lines from the beginning of the buffer; inserting or deleting a line changes the line numbers of all subsequent lines. Commands and switches that rearrange or record line numbers (e.g., "RESEQUENCE", "/SEQUENCE") are not implemented.

## 25.18. Specifying Line Numbers in Line Mode Commands

In line mode commands, line numbers may be specified as:

```
{n}{.{m}}
```

The possible combinations mean:

| | |
|---|---|
| `n` | `absolute line number n` |
| `n.` | `first line of page n` |
| `n.m` | `line m of page n` |
| `.` | `current line` |
| `.m` | `line m of current page` |

where pages are separated by page mark (eop) characters.

## 25.19. "EXIT" and File Names

The line mode "EXIT" command ignores its file name argument; the file name used is the one associated with the buffer (use "SHOW BUFFER" to see which file names are associated with which buffers). All commands that exit MEDT prompt to save all altered buffers.

The EDT journal file facility is not supported.

## 25.20. "WRITE" and File Names

The line mode "WRITE" command does not require a file specification; the last file name used is used if none is specified (but if no file name is specified, no other argument can be specified; otherwise it will mistake the other argument for the file specification).

## 25.21. "CLEAR"

The line mode "CLEAR" command does delete the "PASTE" buffer if "PASTE" is specified as an argument; however, the "PASTE" buffer is recreated if necessary.

## 25.22. Ignored "SET/SHOW" Options

The following SET/SHOW options are ignored:

- (NO)AUTOREPEAT
- CASE NONE/LOWER/UPPER
- ENTITY PAGE: pages are always delimited by eop characters
- (NO)FNF: user is always prompted for new file
- HELP: help not implemented
- MODE LINE/CHANGE: line mode is not supported
- (NO)NUMBERS: line mode is not supported
- PROMPT
- SUMMARY

- TERMINAL: all suboptions are ignored
- TEXT END/PAGE
- (NO)TRUNCATE: lines too long to display are always truncated

## 25.23. "SET SEARCH"

No spaces allowed before "INSENSITIVE" in "SET SEARCH CASEINSENSITIVE" or "SET SEARCH DIACRITICALINSENSITIVE".

"SET SEARCH GENERAL", "SET SEARCH CASEINSENSITIVE", and "SET SEARCH DIACRITICALINSENSITIVE" are synonyms, since diacritical marks are not supported.

## 25.24. "SET COMMAND"

There is no default extension appended to the line mode "SET COMMAND" file names; the full file name must always be specified. The "SET COMMAND" immediately sourcefiles the specified file, then returns to the file from which the command was given, if it was given in a file.

## 25.25. "SET LINES"

The line mode "SET LINES" command adjusts only the height of the current window, not of the entire screen used by EDT; other windows on the screen, if any, are moved or resized as necessary.

## 25.26. "SHOW BUFFER" and "SHOW FILES"

"SHOW BUFFER" and "SHOW FILES" both show the same information, which is the same as shown by the MAINEDIT "Q=" command.

## 25.27. "." at End of Key Definition

"." at the end of a key definition may be displayed as a pair of asterisks, since it is replaced by an internal sequence of unprintable characters that cannot be confused with a period inside a string or file or buffer name.

## 25.28. "DEFK" Limitations

The nokeypad "DEFK" command (keypad CTRL-K) redefines a two-key sequence only if the first key is GOLD. Otherwise it accepts only one-key sequences. It can't accept key sequences containing ENTER (<ecm>). The definition is terminated with <eol> instead of ENTER, so only one-line definitions are allowed. Use the line mode "DEFINE KEY" for arbitrary definitions (using "^M" to represent <eol>). If you want to change which key is GOLD, use "DEFK"; the new "NAME" command changes the name of a key but does not affect the properties associated with GOLD (repeat, etc.). Changing GOLD does not affect previously defined key sequences starting with GOLD; they are still composed of the same keystrokes as before.

## 25.29. Leaving Escape Mode

<ecm> is always the key used to leave Escape mode.

# 26. MEDT Extensions

## 26.1. "(NO)LEFTBORDER", "(NO)RIGHTBORDER"

The following "SET/SHOW" options have been added:

- (NO)LEFTBORDER: display MAINEDIT-style left window border (a colon, or the letter "E" after end-of-buffer, or an asterisk if characters are hidden to the left of the window border).
- (NO)RIGHTBORDER: display MAINEDIT-style right window border (a colon, or nothing if after end-of-buffer, or an asterisk if characters are hidden to the right of the window border).

## 26.2. Invoking a Program

The nokeypad "G" command may be used to invoke a program; e.g.:

```
"gdebug<eol>"
```

invokes MAINDEBUG. If no module name follows "G", MEDT escapes to the current program, if any.

## 26.3. Switching Buffers

The nokeypad "O" command may be used to move to another buffer. It has the form:

```
{+/-}{count}O{=buffer}
```

If an initial plus or minus is specified, and the buffer must be brought onto the screen, then the buffer is inserted at the top of the screen (for "-"), or the bottom of the screen (for "+"); if neither plus nor minus is specified, the buffer replaces the contents of the current window. If the initial count is specified, and the window for the buffer must be created, then the count specifies the number of lines to use. If the buffer name is not specified, the cursor moves to the current "next" buffer, usually the last buffer visited.

## 26.4. Switching Files

The line mode "VISIT" command may be used to edit another file. A new buffer is set up for the new file. The form of the command is:

```
VISIT fileName {+/-}{count}
```

If fileName contains spaces, it may be quoted with single or double quotes. If plus or minus is present, the window for the new buffer is inserted below or above, respectively, the current buffer; otherwise, the current window is used. If a new window must be inserted, count is the size of the new window, in lines; otherwise, a default size is used.

If fileName is terminated with a comma, the editor prompts for front and back end names to use in the new buffer. If the file is more than one line long, the editor prompts for initial page and line.

## 26.5. Keypad Mode Key Definitions

Any key may be defined to have any value in keypad mode. Even printing keys may be redefined. For example, though the original definition of uppercase "A" is "65ASC." on an ASCII system, it can be redefined to have any definition. Key sequences of any length may be defined, and need not begin with GOLD; for example, if <del> is undefined, you may do:

```
DEFINE KEY DELETE DELETE CONTROL K AS "(3L2C)."
```

which defines the three-character sequence <del> <del> CTRL-K to advance three lines and two characters in the current direction. If <del> is currently defined, this definition will be remembered but will have no effect, since the <del> key will be executed as soon as it is pressed. To undefined a key sequence, just define it as the null string.

## 26.6. Key Numbering and Naming

Since MEDT does not know how to name or map most keys on a keyboard other than that of a VT100 terminal, the user must issue commands to assign names and mappings to keyboard keys. Those keys that are always present, e.g., "DELETE", "ENTER" (which is always the initial name of the <ecm> key), the control characters, etc., are always assigned names and given the mappings described in the EDT manual.

Each special key on the keyboard generates a unique integer code. You must determine this code in order to name it or use it in a key definition. The nokeypad "KEYNUM" command may be used to determine the code of a key. The command prompts for you to type a key; then it prints the key code. The <ecm> key (ENTER) and the <abort> key will not work with

"KEYNUM". The code of the <ecm> key is always -1; the <abort> key has no code, since it cannot be typed as part of a command (it cannot be mapped).

A key with code n may be referred to with the name "@n"; e.g., in "DEFINE/SHOW KEY", "@-1" may be used instead of "ENTER", or "@127" may be used instead of "DELETE" on an ASCII system. However, it is more convenient to refer to keys by name. The line mode "NAME" command establishes a name for a key; it has the format:

```
NAME <key number> AS "string"
```

For example, to establish the definitions of ENTER and DELETE on an ASCII system, MEDT initially does:

```
NAME -1 AS "ENTER"
NAME 127 AS "DELETE"
```

The "SHOW NAME" command may be used to examine key names. It has three forms:

```
SHOW NAME "key name"
SHOW NAME <key number>
SHOW NAME *
```

The first form shows what code corresponds to the name specified; for example, on an ASCII system:

```
SHOW NAME "CONTROL K"
```

shows that "CONTROL K" is the name for key number 11.

The second form shows the name(s) corresponding to a given key code; for example:

```
SHOW NAME 15
```

shows that "CONTROL O" is the name for key number 15. More than one name may correspond to a single key code.

The last form, "SHOW NAME *", shows all key name-code correspondences.

For convenience, names have been defined for the numeric keyboard keys and letters, in addition to the control keys, DELETE, and ENTER. "#n" is the keyboard digit n ("n" cannot be used, since the VT100 names of some keypad keys use this form). The uppercase letters are named "A" through "Z"; the lowercase letters, "LC A" through "LC Z".

## 26.7. Front Ends and Views

More than one front end may be used with a single buffer. A buffer generally has one front end that displays the text of the buffer (this is called the "view" front end) and front end that processes the user's commands (the "command" front end). These two front ends are usually the same but may be different. It is useful to switch front ends if one front end provides a desired command but the current front end does not; e.g., an MEDT user may wish to switch front ends to issue a MAINED or MAINVI command.

The line mode commands provided to manipulate view and command front ends are:

```
KILLF frontEnd
SWF frontEnd {commands}
SWFV frontEnd {commands}
SWM {commands}
SWV frontEnd
```

The commands perform as follows:

- KILLF frontEnd: discard the front end frontEnd for the current buffer.
- SWF frontEnd {commands}: switch the command front end to frontEnd. If commands are provided, the commands are executed and control is returned to MEDT; otherwise, the command front end remains frontEnd.
- SWFV frontEnd {commands}: switch the command and view front ends to frontEnd. If commands are provided, the commands are executed and control is returned to MEDT (but the view front end remains frontEnd); otherwise, the command front end remains frontEnd.
- SWM {commands}: equivalent to SWFV MAINED {commands}.
- SWV frontEnd: switch the view front end to frontEnd.

# Appendix B. The DATMGR Back End

The DATMGR back end can be used to edit data files. It translates the data in a data file into readable text.

Example B-1 shows a few lines of a data file edited with back end DATMGR. The number in the leftmost column is the position of the data in the file. The next three columns are the data shown in hexadecimal, octal, and characters, respectively. Non-printable characters are displayed as a character code surrounded by angle brackets.

```
 0:    'H0000013C   '00000000474   <0><0><1><
 4:    'H267C0000   '04637000000   &<124><0><0>
 8:    'H00884EFB   '00042047373   <0><136>N<251>
12:    'HB800286E   '27000024156   <184><0>(n
16:    'HFFFC4E75   '37777047165   <255><252>Nu
20:    'H4E714E71   '11634247161   NqNq
24:    'H4E56FFF2   '11625577762   NV<255><242>
28:    'H2D4CFFFC   '05523177774   -L<255><252>
32:    'H2D7C0041   '05537000100   -<124><0>A
36:    'H0003FFF8   '00000777770   <0><3><255><248>
```

Example B-1. Sample Buffer Using Back End DATMGR

Data are modified by changing the hexadecimal values. The octal and character values do not have to be modified (only the first long bits on the line affects the value in the file). When DATMGR writes the data to the file, it scans each line for a single quote ("'"). If it finds one, then it looks for a valid long bits constant (without the trailing "L"). A valid long bits constant is a sequence of characters preceded by a single quote and a letter that indicates the base: "B" (or "b") for binary, "O" (or "o") for octal, or "H" (or "h") for hexadecimal. The base letter may be omitted for octal; i.e., octal is the default.

Lines may be inserted or deleted. A line is legal as long as it contains at least one valid long bits constant. If lines are inserted or deleted, the reported positions will no longer be correct; they are displayed for informational purposes only and are not checked when the buffer is saved.

When DATMGR saves the buffer, it writes one long bits value for each line in the buffer. An error message is displayed and the user is prompted for a new value for each invalid line in the buffer.

DATMGR is not appropriate for very large data files, since it builds up in memory the text representation that it displays.

# Appendix C. Display Modules

The currently available set of display modules is described on the following pages. Additional display modules can be made available; contact XIDAK if you need support for another terminal.

For those terminals that have a keypad or special keys, macro ID keys are listed. Arrow keys and macro ID keys can be used in any mode (command, insert or overstrike). Arrow keys (which on most terminals have an arrow drawn on the keycap) map to default macros as shown in Table C-1. These macro ID keys may be defined by the "/" command like any other control sequence.

Macro ID keys are initially undefined. Invoking an undefined macro causes MAINED to beep.

| Key Code Name | Default Mapping |
|---|---|
| dpyUp | ^ Q R M |
| dpyDown | \ Q R M |
| dpyLeft | < Q R M |
| dpyRight | > Q R M |

Table C-1. Names of the Arrow Key Codes and Default Macros

On many terminals, two <esc>'s constitute a single control sequence. On such terminals, it is often convenient to define "<esc><esc>" to be <ecm>. To do this, type "/<esc><esc><ecm><ecm>/". The two <esc>'s are mapped to a single control sequence, which is taken as the macro command name. The last <ecm> is removed as described in Section 16.4, leaving <esc> defined as just <ecm>.

## C.1. Display Modules AM48 and AM60

| | |
|---|---|
| Terminals | Ambassador |
| <ecm> | RESET key |
| <abort> | CTRL-L |
| Maximum columns in window | 77 |
| Maximum rows in window | 46 with AM48, 58 with AM60 |

The numeric keypad keys 8 (up arrow), 2 (down arrow), 6 (right arrow), and 4 (left arrow), generate dpyUp, dpyDown, dpyRight, and dpyLeft, respectively. The other numeric keypad keys and the keys on the top row (except SETUP and SEND) are macro ID keys.

It is important to specify the correct baud rate for the Ambassador display modules, as the terminal requires padding for a number of display commands at high baud rates.

The Ambassador terminal should have the following configuration settings for use with the AM48 and AM60 display modules:

- XON/XOFF should be disabled if the operating system does not intercept XON/XOFF.
- Parity detection should be disabled.
- The "PAUSE" key may be enabled to set the parity bit in characters transmitted by the terminal. If the operating system does not remove the parity bit, the "PAUSE" key may be used to generate additional MAINED macro ID's.

## C.2. Apollo Display Modules

|  |  |
|---|---|
| `Terminals` | `Apollo portrait, landscape, and color displays` |
| `<ecm>` | `HOLD/GO or ABORT/EXIT key` |
| `<abort>` | `CTRL-L` |
| `Maximum columns in window` | `depends on font and display` |
| `Maximum rows in window` | `depends on font and display` |

These display modules run only under the Aegis and DOMAIN/IX operating systems on systems manufactured by Apollo Computer, Inc.

The up arrow, down arrow, left arrow, and right arrow keys on the left-hand keypad generate dpyUp, dpyDown, dpyLeft, and dpyRight, respectively. All other left-hand keypad keys, the function keys at the top of the keyboard, and the MARK, SHELL, READ, and EDIT keys are macro ID keys in borrowed display mode; these keys are also macro ID keys in frame mode, except that the CTRL-P, CTRL-W, MARK, SHELL, CMD, NXT WNDW, READ, and EDIT keys all perform their usual display manager functions.

The Apollo display modules are called BORRO, FRAME, FBORRO, and FFRAME. The module BORRO always borrows the entire display and uses the standard font, normal video, with the touchpad in absolute mode.

The module FRAME runs MAINEDIT in the current Aegis window. The standard font is used. The character dimensions of the display depend upon the size of the window. If the dimensions of the window are changed during the course of the MAINEDIT session, you may need to enter and leave the window again and give the MAINED "QN" command to allow MAINEDIT to perceive that the size of the display has changed (this leads to undesirable behavior in the MAINVI and MEDT front ends).

The modules FBORRO and FFRAME are versions of BORRO and FRAME, respectively, that allow you to specify a fixed-width font to be used by MAINEDIT. No check is made that the font is in fact of fixed width, and undesirable behavior results if a variable-width font is used. The display module normally prompts for the name of the Aegis font file.

If running on a portrait display and the logical name "(Default Font)" is defined, that name is displayed in parentheses after the "Font:" prompt. If you respond <eol> to the prompt, that font is used. Similarly, default fonts may be specified by defining the logical names "(Default Font.191)" for a landscape display and "(Default Font.color)" for a color display. If it is desired that the "Font:" prompt not appear at all, and the specified font used automatically, define instead the logical names "(Unconfirmed Default Font)", "(Unconfirmed Default Font.191)", or "(Unconfirmed Default Font.color)", respectively.

In responding to the "Font:" prompt, you may follow the font name with any number of the options below, separated by spaces. The options are:

- i: Use inverse video (prompts for colors on color node unless b or t is given)
- a: Absolute touch pad mode
- r: Relative touch pad mode
- o: Both relative and absolute (rel-abs)
- b<n>: Use background color numbered <n>
- t<n>: Use text color numbered <n>

Colors on the color node are numbered between 0 and 15 on displays with four-bit pixels and between 0 and 255 on displays with eight-bit pixels. The first eight colors are preset by Aegis as in Table C.2-1. These mappings are not accurate if your color map has been changed. Also, it is recommended that "1" not be used for the background color because this is the color to which the cursor is mapped.

| | |
|---|---|
| 0: | black |
| 1: | red |
| 2: | green |
| 3: | blue |
| 4: | cyan |
| 5: | yellow |
| 6: | magenta |
| 7: | white |

Table C.2-1. Color Map as Preset by Aegis

## C.3. Display Module AT386

| | |
|---|---|
| Terminals | IBM PC/AT and compatibles |
| <ecm> | <esc><esc> |
| <abort> | CTRL-L |
| Maximum columns in window | 77 |
| Maximum rows in window | 22 |

The keys marked with up arrow, down arrow, left arrow, and right arrow generate dpyUp, dpyDown, dpyLeft, and dpyRight, respectively. Other special keys generate macro ID's.

## C.4. Display Module BIGSUN

|  |  |
|---|---|
| Terminals | Sun Microsystems workstations (both M68000 and SPARC processors) |
| <ecm> | <esc><esc> |
| <abort> | CTRL-L |
| Maximum columns in window | depends on font and display |
| Maximum rows in window | depends on font and display |

These display modules run only under the UNIX operating system on systems manufactured by Sun Microsystems, Inc.

BIGSUN is a display module that takes advantage of the Sun display hardware and software to allow the use of user-created fonts and any number of columns on the screen. In two-column mode with the default font on a screen 1152 by 900, BIGSUN effectively provides an ASCII display terminal with over 170 lines.

A special bootstrap must be used to run BIGSUN, since the appropriate Sun library routines must be linked in. To make such a bootstrap, restore from the CONF configuration file "sun2b.cnf" on the MAINSAIL directory. Link the resulting output with "mb.o" instead of "m.o" on the MAINSAIL directory. The boostrap "mainsab" on the MAINSAIL directory is configured to be able to run BIGSUN.

When BIGSUN is initialized, it opens a text file called "(BIGSUN font file)" and reads the description of the font to be used. The default "(BIGSUN font file)" describes a font 7 by 10 pixels.

The font file is divided into pages. The first page of the font file gives the name and dimensions of the font in the form:

```
FIXED <font name>
<height of character, including descender>
<width of character, including spacing>
<left extension of characters>
<descender>
```

Subsequent pages describe bitmaps. A bitmap consists of spaces (no tabs) and lowercase x's to indicate clear and set pixels, respectively. Each bitmap contains a period (".") or uppercase "X" to indicate the origin of the character; the period is used if the pixel at the origin is clear, and the uppercase "X" if the pixel is set.

The second page of the file is the pattern for the cursor; subsequent pages are for the ASCII character codes, starting at 0. After the last character in the file should appear a page with just the letter "Z" on it. If a character is not present in the font, its page should be blank.

For example, a page describing an uppercase "A" might look like:

```
# Uppercase A (ASCII 65)
      x
     xxx
     xxx
    x  xx
    x  xx
   x    xx
   xxxxxxx
  x      xx
  x      xx
 x        xx
Xxx      xxxx
```

A lowercase "g" might look like:

```
# Lowercase g (ASCII 103)
          x
   xxxxxxx
  x     xx
 xx       xx
 xx       xx
  xx      x
   xxxxxx
  xxx
.  xxxxxx
 x     xxx
xxx     x
  xxxxxx
```

A comment line beginning with "#" may be used anywhere in the font file.

When BIGSUN is initialized, it also attempts to open a file called "(BIGSUN columns)". If unsuccessful, it prompts for the number of columns to use in the display, and allows no margins (unused space at the edge of the screen). If it opens the file, it reads from it five integers in the order:

```
<number of columns to use>
<top margin, in pixels>
<bottom margin, in pixels>
<left margin, in pixels>
<right margin, in pixels>
```

Margins are useful if any of the edges of the display are invisible.

When rsh'd to another Sun system, do not use BIGSUN, since it will execute (i.e., take over the screen) on the system where it is invoked, not the system at which you are sitting.

## C.5. Display Module D400

| | |
|---|---|
| Terminals | Data General DASHER |
| <ecm> | <esc> key |
| <abort> | CTRL-L |
| Maximum columns in window | 77 |
| Maximum rows in window | 22 |

The keys to the right of the space bar marked with up arrow, down arrow, left arrow, and right arrow generate dpyUp, dpyDown, dpyLeft, and dpyRight, respectively.

## C.6. Display Modules D460 and D460C

| | |
|---|---|
| `Terminals` | `Data General D410/460` |
| `<ecm>` | `<esc> key` |
| `<abort>` | `CTRL-L` |
| `Maximum columns in window` | `78 with D460, 132 with D460C` |
| `Maximum rows in window` | `22` |

The keys to the right of the space bar marked with up arrow, down arrow, left arrow, and right arrow generate dpyUp, dpyDown, dpyLeft, and dpyRight, respectively.

## C.7. Display Module DATAME

| | |
|---|---|
| Terminals | Datamedia 3000, Telemedia |
| <ecm> | <esc> |
| <abort> | CTRL-L |
| Maximum columns in window | 77 |
| Maximum rows in window | 22 |

Simultaneously pressing an EDIT key and typing any other character generates a macro control sequence. A unique control sequence is generated for every character, i.e., "EDIT CTRL-A", "EDIT a", and "EDIT A" are all different control sequences.

The up arrow, down arrow, left arrow, and right arrow keys generate dpyUp, dpyDown, dpyLeft, and dpyRight, respectively.

This display module works only on the Datamedia 3000, not the Datamedia 2500.

## C.8. Display Module EWY100

| | |
|---|---|
| Terminals | Wyse WY-100 as modified by ELXSI, Inc. |
| <ecm> | <esc><esc> |
| <abort> | PAGE (unshifted) |
| Maximum columns in window | 77 |
| Maximum rows in window | 22 |

The arrow keys on the edit keypad marked with up arrow, down arrow, left arrow, and right arrow generate dpyUp, dpyDown, dpyLeft, and dpyRight respectively. Other edit keypad keys and the black keys on the top row are macro ID Keys. Most of these keys generate different macro ID's when shifted. The numeric keypad keys generate only numbers.

## C.9. Display Module HEATH

| | |
|---|---|
| Terminals | Heath (or Zenith) H-19 |
| <ecm> | ENTER key |
| <abort> | CTRL-L |
| Maximum columns in window | 77 |
| Maximum rows in window | 22 |

The numeric keypad keys with the up arrow (8), down arrow (2), left arrow (4), and right arrow (6) generate dpyUp, dpyDown, dpyLeft, and dpyRight respectively. The remaining keys on the right-hand numeric keypad (except for ENTER), the function keys F1 through F5, the ERASE key, and the keys with a colored square in the middle are macro ID keys.

Sample internal Heath switch settings are shown in Figure C.9-1. The settings that are important to the proper functioning of the HEATH display module are marked with an asterisk. Other settings (parity and baud rate in particular) may be necessary to talk to certain computers, and certain settings (e.g., keyclick) are purely a matter of personal taste.

The terminal initialization sequence sets "no auto lf", "no auto cr", and "wrap at end of line". The terminal initialization does not work correctly if the terminal has been set to ANSI mode. If the terminal has been set to ANSI mode, the only way to restore it to normal operation is to reset it by simultaneously pressing the right-hand SHIFT key and the RESET key.

```
SWITCH 402:     +-----------------+
                | 0 1 1 0 0 0 0 0 |
                | 0 1 2 3 4 5 6 7 |
                +-----------------+
                  | | | | | | | |
                  | | | | | | | +-----> 60 Hz refresh
                  | | | | | | +-------> normal keypad *
                  | | | | | +---------> Heath mode *
                  | | | | +-----------> no auto CR *
                  | | | +-------------> no auto LF *
                  | | +---------------> wrap line *
                  | +-----------------> no keyclick
                  +-------------------> underscore cursor

SWITCH 401:     +-----------------+
                | 1 0 1 0 0 0 0 1 |
                | 0 1 2 3 4 5 6 7 |
                +-----------------+
                  | | | | | | | |
                  | | | | | | | +-----> full duplex
                  | | | | | | +-------> normal parity
                  | | | | | +---------> odd parity
                  | | | | +-----------> no parity
                  | | | +------------->
                  | | +---------------> baud rate = 1200
                  | +----------------->
                  +------------------->
```

Figure C.9-1. Sample Heath Switch Settings

## C.10. Display Modules HP300H and HPTERM

| | |
|---|---|
| Terminals | HP2392A and compatible models |
| <ecm> | home key |
| <abort> | CTRL-L |
| Maximum columns in window | 77 (HPTERM); 126 (HP300H) |
| Maximum rows in window | 22 (HPTERM); 43 (HP300H) |

The black arrow keys marked with up arrow, down arrow, left arrow, and right arrow generate dpyUp, dpyDown, dpyLeft, and dpyRight respectively. On the HP2392A, the home key is a white arrow pointer to the upper left, just above the black arrow keys.

## C.11. Display Module LINDPY

| | |
|---|---|
| Terminals | any |
| <ecm> | ! or #C |
| <abort> | #A |
| Maximum columns in window | user-settable |
| Maximum rows in window | user-settable |

LINDPY performs input and output to the standard MAINSAIL files cmdFile and logFile, and may therefore run on any terminal. It is more cumbersome to use than a terminal-specific display module, but may be used on terminals for which a terminal-specific display module has not yet been written or which do not provide the required display functions (e.g., a hardcopy terminal).

LINDPY is not able to respond to keystrokes as they are typed or position the cursor on the screen. Instead, it accepts input one line at a time, and redisplays the entire "screen" after processing each line of input.

When LINDPY is initialized, it prompts for the size of the "screen". It then displays the number of lines specified after each line of input, except that trailing blank lines are not displayed. The cursor appears as a caret ("^") character immediately under the character on which it is positioned. The caret is displayed on a line by itself.

Because keypad and other special keys cannot be recognized by LINDPY, special sequences are used to simulate special keys. See Table C.11-1.

cmdFile and logFile may be redirected when LINDPY is used, so that a "batch editing" facility may be simulated.

| Character(s) | Key Simulated |
|---|---|
| ! | <ecm> |
| #D or #d | <del> |
| #E or #e | <eol> |
| #T or #t | <tab> |
| #L or #l | <lf> |
| #S or #s | <sp> |
| #B or #b | <bs> |
| #C or #c | <ecm> |
| #A or #a | <abort> |
| #X or #x | <esc> |
| #! | ! |
| #? | ? |
| ## | # |
| #<any other character> | macro ID |

Table C.11-1. LINDPY Special Sequences

## C.12. Display Modules SUN, SUN3, and SUN46

| | |
|---|---|
| Terminals | Sun Workstation monitor |
| <ecm> | <esc><esc> |
| <abort> | CTRL-L |
| Maximum columns in window | 77 (SUN and SUN46); depends on display (SUN3) |
| Maximum rows in window | 33 (SUN), 44 (SUN46); depends on display (SUN3) |

The SUN display module can be used only on the Sun Workstation, which is a Motorola M68000-based UNIX system produced by Sun Microsystems, Inc., or on another computer system using a Sun Workstation as a remote terminal.

The keys in the upper right part of the main keyboard with the up arrow, down arrow, left arrow, and right arrow generate dpyUp, dpyDown, dpyLeft, and dpyRight respectively. The PF keys are macro ID keys. The numeric keypad cannot be mapped, that is, it produces only numbers.

SUN46 is identical to SUN, except that it assumes a display with 46 lines rather than 35.

SUN3 is identical to SUN, except that it determines the size of the window dynamically. It may hang if run on a terminal that does not provide the Sun function that returns the terminal size; if it does hang, repeatedly hit carriage return until MAINSAIL starts responding.

## C.13. Display Modules TELEVI and TVI950

| | |
|---|---|
| Terminals | Televideo (various models) |
| <ecm> | HOME key |
| <abort> | FUNCT q |
| Maximum columns in window | 77 |
| Maximum rows in window | 23 |

TELEVI is used for Televideo models 912, 912B, 912C, 920, 920B, and 920C. TVI950 is used for model 950.

The code "FUNCT q" is generated by holding down the key labeled "FUNCT" and typing a lowercase letter "q".

The keys to the right of the space bar marked with up arrow, down arrow, left arrow, and right arrow generate dpyUp, dpyDown, dpyLeft, and dpyRight, respectively. The keys labeled with words across the top of the keyboard and numeric keypad are macro ID keys. Holding down the FUNCT key and typing any other key also generates a macro control sequence. Case is distinguished; i.e., "FUNCT a" is not the same as "FUNCT A".

The numeric keys on the numeric keypad are not mappable; they generate only numbers.

CTRL-A is not available as a macro ID on the Televideo terminals, since most keypad codes generate sequences beginning with CTRL-A. A series of three CTRL-A's, however, constitute a single control sequence. Also, <abort> is not mapped to CTRL-L since the right arrow key generates CTRL-L.

When the TELEVI and TVI950 modules are used on a half-duplex system, the cursor position is marked by displaying the character at the cursor with the half-intensity attribute (a half-intensity "^" is displayed when the cursor is on a blank).

## C.14. Display Module TRMCAP

| | |
|---|---|
| Terminals | Those listed in data base file |
| `<ecm>` | Depends on terminal entry |
| `<abort>` | Depends on terminal entry |
| Maximum columns in window | Depends on terminal entry |
| Maximum rows in window | Depends on terminal entry |

TRMCAP attempts to read a data base with the same format as the UNIX TERMCAP terminal capability file. This is a file with a list of terminal names and descriptions of the command sequences they require. Consult a "UNIX Programmer's Reference Manual", Chapter 5, for further information on TERMCAP files (this document is not supplied by XIDAK).

The TRMCAP display module assumes the data base file is called "(termcap data base)". A MAINEX ENTER subcommand may be used to define this as a logical name for the real file name of the data base (see the "MAINSAIL Utilities User's Guide"). TRMCAP prompts for the terminal entry to be used when it is first invoked. For example, to select the VT100 entry from the data base:

```
                    Termcap display: VT100<eol>
```

TRMCAP is no more reliable than the available data base. Official UNIX TERMCAP data bases vary greatly in quality from site to site. If a XIDAK display module exists for a particular terminal, it is almost certainly better to use that display module than to use TRMCAP.

## C.15. Display Module VIS550

| | |
|---|---|
| Terminals | Visual 300, 550 |
| <ecm> | ENTER key or <esc><esc> |
| <abort> | CTRL-L |
| Maximum columns in window | 77 |
| Maximum rows in window | 30 |

The VIS550 display module is unsupported.

The arrow keys marked with up arrow, down arrow, left arrow, and right arrow generate dpyUp, dpyDown, dpyLeft, and dpyRight respectively. The function keys generate macro ID's.

## C.16. Display Modules VT100, VT102, and VT102M

| | |
|---|---|
| Terminals | VT100, VT102, and compatible models |
| <ecm> | ENTER key |
| <abort> | CTRL-L |
| 80 Column mode: | |
| Maximum columns in window | 77 |
| Maximum rows in window | 22 |
| 132 Column mode: | |
| Maximum columns in window | 129 |
| Maximum rows in window | 12 |

DEC's VT100 line of terminals (and compatible terminals made by other manufacturers) includes some terminals that support character and line insert and delete, and some that do not. Use the VT100 module for terminals without these capabilities, and VT102 or VT102M for the others.

The numeric keypad keys labeled 8, 2, 4, and 6 generate dpyUp, dpyDown, dpyLeft, and dpyRight, respectively. Other numeric keypad keys (except ENTER) are macro ID keys.

When the display modules VT100 and VT102 initialize the terminal, they normally check automatically to determine whether the terminal is in 80-column mode or 132-column mode. Such a check is not possible on half-duplex systems, so 80-column mode is always assumed on such systems. VT102M does not perform this check, even on full-duplex systems. It should be used on VT102-compatible terminals that do not support the check.

When the display module deinitializes the terminal (as when MAINEDIT returns to the operating system), the terminal's scroll region is set to be the entire screen.

It is important to specify the correct baud rate for the VT100 and related display modules, as the terminals may require padding for a number of display commands at high baud rates. The VT100 display modules do not work correctly if the terminal is in the "smooth scroll" mode; users are responsible for ensuring that the terminal is in "jump scroll" mode before using the display modules.

## C.17. Display Modules WY43, WY50, WY5043, and WY75

| | |
|---|---|
| `Terminals` | `Wyse WY-50 (WY50 and WY5043);` `Wyse WY-75 (WY75);` `Wyse WY-60 (WY43)` |
| `<ecm>` | `ENTER` |
| `<abort>` | `CTRL-L` |
| `Maximum columns in window` | `77` |
| `Maximum rows in window` | `22 (WY50 and WY75);` `41 (WY5043 and WY43)` |

The yellow arrow keys below the RETURN key marked with up arrow, down arrow, left arrow, and right arrow generate dpyUp, dpyDown, dpyLeft, and dpyRight respectively. The function keys and keypad keys are macro ID keys. The function keys generate different macro ID's when shifted.

# Index

' command 40

(
  command 23
  commands 30

)
  command 23
  commands 30

* 8, 10, 25

+ 28
+F command 73

, command 75

- 28

.* command 79
.+ command 79
.- command 79
./ command 79
.= command 79
.^ command 79
._ command 79
.A commands 58
.B commands 59, 62
.C commands 46
.F commands 59, 62
.J commands 50
.K commands 62
.O commands 82

/ command 74

: 25, 84

<
  command 23
  commands 30
<abort> 2, 69

<bs> 2, 30
  as macro ID 75
<del> 2, 30
<DELETEDPAGES> 42
<ecm> 2, 22
<eol> 2, 30
  as macro ID 75
<esc> 2
<lf> 2, 30
  as macro ID 75
<sp> 2, 30
<suppress-output> key 71
<tab> 2, 30, 72, 83
  as macro ID 75

= command 10, 71

>
  command 23
  commands 30

@ commands 33

[ and ] in command descriptions 3

\
  command 23
  commands 30

^
  command 23
  commands 30

{ and } in command descriptions 3

| in command descriptions 3

1IB command 39
1IF command 39

A commands 69
abbreviation of buffer names 26
abort 2
aborting
  a command 69

terminal output 71
accumulators 79
again command 69
AM48 module 134
AM60 module 134
Ambassador 134
anchoring a window 56
Apollo Computer, display module for 135
arbitrary characters, inserting 40
arguments, command line 7
arithmetic functions 79
arrow keys 133
AT386 module 137
autosave limit 73

B commands 48
back end 4
back end, specifying 64
BACKEND keyword 16
basic command 29
batch editing 148
baud rate 6
BAUDRATE keyword 14
BIGSUN module 138
border 84
BORRO module 135
breaking lines 48
BS (backspace) key 2
buffer 3
delete 42
deleting 62
end 25
inserting 39
listing status of 63
buffer names
abbreviated 26
changing 62
buffers, moving among 59

C commands 46
case
conversion commands 49
upper and lower 20
centering lines 46
CM command 77

CMDLOG (special buffer) 86
command
  front end 17
  line arguments 7
  mode 21
CONTEXT keyword 12
CONTROL key 3
control
  characters 10, 40
  characters (on message line) 8
  sequence 74
converting to upper or lower case 49
copying text 46
counts 3, 28
creating a text file 5
CTRL key 3
current character, word, line, and page 24
current window, status line 9
cursor 10, 24
  movement 23, 30
  movement with arrow keys 133
cursor movement, among buffers 59

D commands 24, 42
D400 module 141
D460 module 142
D460C module 142
data file, editing 131
DATAME module 143
Datamedia 3000 143
DATMGR back end 4, 131
default
  baud rate 14
  display module 13
  file 12
  page and line 12
  window width 15
defining a macro 74
DELETE key 2
delete buffer 42
DELETEDPAGES 42
deleting 24, 42
  buffer 62
  pages 42
direction of MAINED commands 28

display module 6, 133
display module, default 13
DISPLAYMODULE keyword 13
DL command 25
DONOTPROMPTFORBAUDRATE keyword 14
DONOTPROMPTFORDISPLAYMODULE keyword 13
DONOTUPDATEEPARMS keyword 15
duplicating text 46

E
  commands 87
  in margin of window 25
EDIT module 5
editing
  a data file 131
  a text file 5
  in batch 148
editor 1
EDITORPARMSFILE keyword 12
emphasis 29
end of buffer 25
ENTER key 2
enter command mode 2, 22
eparms file 5, 12, 20
eparms file, changing 16
ESCAPE key 2
escape mode 21, 22, 87
EWY100 module 144
executing modules from MAINED 86
exiting
  MAINED 21
  MAINEDIT 11, 67

F commands 67
FBORRO module 135
FFRAME module 135
file
  association with buffer 3
  editing 5
  inserting 39
  name 2
  selection 59
file name, changing 62
files
  multiple 59

multiple opens 93
filling text 50
FRAME module 135
front end 4
front end, specifying 64
FRONTEND keyword 4, 16

G
command 25
commands 30
globalLeftBorder 84
globalRightBorder 84

H commands 70
hardcopy terminals and MAINEDIT 148
HEATH module 145
Heath H-19 145
Hoops! 70
HP300H module 147
HPTERM module 147

I commands 38
identifier search 34
information 63
INITIALIZE (special macro) 77
insert mode 21, 22, 38
inserting
blank lines 38
characters (I commands) 38
files and buffers (I commands) 39
insertModeDefault 85
invoking modules from MAINED 86

J commands 48
joining lines 48
justifying text 50

K
command 24
commands 42
keyboard macros 74
keys, special 2
killing text 42

L commands 49
leaving
  MAINED 21
  MAINEDIT 11
left-right scrolling 55
LINDPY module 148
line search 34
LINEFEED key 2
localLeftBorder 84
localRightBorder 84
lower case, converting to 49

M commands 47
MACRO keyword 14
macro 74
  and modes 77
  ID keys 133
  special 77
  stored in eparms 14
MAINED 1
  front end 4
MAINEDIT 1
  batch mode 148
  hardcopy terminals 148
MAINVI 1
  front end 4
margins 25
marked location 33
MEDT 1
  front end 4
message line 8
mode 21
  and macros 77
  as displayed in status line 9
module, invoking from MAINED 86
moving
  text 47
  the cursor 23
multiple
  buffers into one file 64
  files 59, 93
  windows 55

N commands 72
n in command descriptions 3, 28

named macro 74
NAMEDMACRO keyword 14
numbers, manipulating 79

O commands 36
object of command 29
Oops! 70
options 82
overstrike mode 21, 22, 36
overstrikeModeDefault 85
overstriking characters (O commands) 36

P command 25
page
  marks 10, 30, 39, 42, 45
  of text 10
pages, deletion and recovery 42
proportionalWindowsMode 85

Q modifier (emphasis) 29
Q= command 63
QF command 21
quitting
  MAINED 21
  MAINEDIT 11

R commands 44
read-only buffer 84
readOnly 84
rearranging text 47
recalling text 42, 44
recognition of buffer names 26
recovering pages 42
recursive macros 75
redirection of I/O in MAINEDIT 148
refreshing the screen 72
repeated execution of commands 74
repetition of previous command 69
RETURN key 2
RM commands 77

S commands 32
save reminders 73
saving files 11, 21, 67
screen

format 8
refresh 72
scrolling 23, 30, 35
left-right 55
searching 33
selecting a file 59
skip to character or line 32
SPACE bar 2
special keys 2
speed of terminal 6
status
line 9
obtaining 63, 71
stickyTabs 83
STREAMS and MAINVI 107
SUN module 150
Sun Workstation 150
Sun Microsystems, display module for 138
SUN3 module 150
SUN46 module 150
suppressing terminal output 71

T commands 33
TAB key 2
tab
inserting before 83
special treatment of 10
stops 72
Telemedia 143
TELEVI module 151
Televideo 151
termcap data base 152
terminal speed 6
terminal output, suppressing 71
text
editor 1
search 33
TRMCAP module 152
TVI950 module 151
TXTMGR back end 4

U commands 49
undeleting 42
text 44
undoing commands 70

UNIX termcap data base 152
unkilling text 42
UPDATEBAUDRATE keyword 14
UPDATEDISPLAYMODULE keyword 13
upper case, converting to 49

V command 31
view front end 17
VIS550 display module 153
visiting a file 59
VT100 module 154
VT102 module 154
VT102M module 154

W commands 23, 35
window 3, 55
  anchoring 56
  size 55
  status line of 9
windowing 23, 30, 35
WINDOWWIDTH keyword 15
wordWrap 84
WY43 module 155
WY50 module 155
WY5043 module 155
WY75 module 155
Wyse
  WY-100 144
  WY-50 155
  WY-60 155
  WY-75 155

X commands 30, 55

Y commands 30, 55

Z commands 43
zapping text 43
Zenith H-19 145

XIDAK, Inc., 530 Oak Grove Avenue, M/S 101, Menlo Park, CA 94025, (415) 324-8745