MicroSPARC Inc.

Apple II/IIe

G.A.L.E.

by Sandy M. Mossberg MD

GALE

How to get the most out of this manual

This Manual is designed in two parts. The first part contains all of the information that you need to know to take full advantage of all of GALE's powerful features.

The second part, Appendix A on page 49, is a tutorial that shows you how to use many of GALE's commonly-used commands. You should jump ahead to Appendix A if you are unfamiliar with Applesoft line editors, or even if you just want to get started with GALE right away!

REPLACEMENT POLICY

All MicroSPARC software may be returned for free replacement (please add $1.50 for shipping) anytime within 30 days of purchase if it is found to be defective. After 30 days, we will replace a damaged disk upon receipt of that disk along with $5.00 to cover our costs (please add $1.50 for shipping).

TECHNICAL SUPPORT

Technical support is available at our offices during normal business hours (Eastern time). We welcome your phone calls (at 617-259-9039) if you have a problem with using our software. We encourage you to write with suggestions concerning our programs and documentation.

ISBN 0-912341-06-8

Revised January 1984

GALE
TABLE OF CONTENTS

SECTION I: INTRODUCTION 1

SECTION II: STARTUP INFORMATION 2
PART 1: SYSTEM REQUIREMENTS 2
PART 2: INSTALLATION AND CONTROL 2

SECTION III: IMMEDIATE MODE 4
PART 1: OVERVIEW 4
PART 2: EDIT MODE ENTRY COMMANDS 4
NORMAL 4
COMPACTED 5
GET 5
PART 3: GLOBAL COMMANDS 5
AUTO 5
BLOAD 6
CHANGE 6
DISK SPACE FREE 8
ESCAPE FUNCTION (MACRO) 8
HIDE 8
LINEFIND 9
MANUAL 9
POINTERS 9
RENUMBER 9
SEARCH 13
UNHIDE 13
VARIABLE XREF 14
EXHUME 14
APPEND 15

PART 4: MISCELLANEOUS COMMANDS 15
HELP SCREEN 15
DECIMAL TO HEX 15
HEX TO DECIMAL 16

SECTION IV: EDIT MODE 17
PART 1: OVERVIEW 17
PART 2: EDIT MODE SCREEN 17
PART 3: EDIT COMMANDS 18
CASE TOGGLE 18
COMPACT 19
DELETE 19
END OF LINE 19
FIND 19
INSERT 20
KILL 20
OVERRIDE 20
PEEK 21
QUIT 21
RESTORE 22
START 22
TRUNCATE 22
ZAP 22
MACRO INSERT 23
FLIP 23
EXIT EDIT MODE 23
ALTERNATE EXIT 23
ARROWS 24
HELP SCREEN 24
PART 4: TYPEOVER 24

SECTION V: ESCAPE FUNCTIONS (MACRO MODE) 25
PART 1: OVERVIEW 25
PART 2: MACROSCOPE 25
ADD ESCAPE FUNCTION 26
DELETE ESCAPE FUNCTION 28
LIST ESCAPE FUNCTION 28
LIST DEFINED KEYS 28
LOAD GALE FILE 28
SAVE GALE FILE 29
CATALOG DISKETTE 31
EXIT MACROSCOPE 31
PART 3: LIST OF STANDARD MACROS 31

SECTION VI: OTHER FILES ON PROGRAM DISKETTE 33
PART 1: GALE OUT 33
PART 2: HELLO 33
PART 3: ESCTBL.ALT 34
PART 4: GALE.64 34
PART 5: GALE//E.40 35
GENERAL 35
EDIT 35
PART 6: GALE.AMP 35
PART 7: GALE MACRO PRINTERS 36

SECTION VII: MEMORY USAGE 39
PART 1: MEMORY MAPS 39
GALE.48 39
GALE.64 40
GALE.AMP 41
GALE//E.40 42
PART 2: FREE AREA 42

SECTION VIII: SUMMARY OF COMMANDS 43
PART 1: IMMEDIATE MODE COMMANDS 43
PART 2: EDIT MODE COMMANDS 44
PART 3: STANDARD ESCAPE FUNCTIONS (MACROS) 45

APPENDIX A: A SAMPLE GALE SESSION 46

GALE

by Sandy Mossberg

SECTION I: INTRODUCTION

A sophisticated utility designed to facilitate all phases of Applesoft program development, GALE (Global Applesoft Line Editor) is to the programmer what a word processor is to the author of prose or poetry. Not only is GALE beautiful, she is intelligent! Line editing, output control, hexadecimal/decimal conversion, auto line numbering, search and change, renumber, append, variable cross reference and user-definable macro function are but some of GALE's extensive capabilities. Convenient help screens provide a complete list of commands.

Despite the fact that all commands are simple and mnemonic, practice makes perfect! Studying the manual and experimenting with GALE will result in optimal utilization of this comprehensive utility.

SECTION II: STARTUP INFORMATION

Part 1: System Requirements

Operation of GALE requires an Apple II, Apple II Plus, or Apple //e with Applesoft in ROM (motherboard or ROM card), at least 48K RAM, one or more Disk II drives, DOS 3.3, and a video device. A printer is useful but not essential.

Part 2: Installation and Control

There are four versions of GALE on the disk: one for 48K computers, one for 64K (or larger) computers, one for the Apple //e in 40-column mode, and one for use with MicroSPARC's AmperSoft software. The correct version is automatically installed upon booting the disk or running the HELLO program.

The discussion below pertains to all versions of GALE, since they are functionally identical. Minor differences are listed in Section VI.

GALE is written entirely in machine language and does not require an annoying BASIC interface. A write protect tab should be placed over the notch on the Program Diskette. Since the Program Diskette is not copy protected, make a back-up copy (use any copy program such as COPYA on the DOS 3.3 SYSTEM MASTER diskette) or move all programs to your work diskette. FID on the DOS 3.3 SYSTEM MASTER diskette is useful for this purpose.

Since GALE controls the input (KSW) and output (CSW) hooks, the commands IN#0 and PR#0 will disconnect GALE's input and output hooks, respectively. This chore may be simplified by defining a macro to do the work (see SECTION V, PART 2). RESET (autostart ROM) and CALL 29952 (old monitor and autostart ROM) reactivate GALE.

GALE does not take kindly to resetting the text window. Doing so may create "garbage" in the current program, thus damaging it irreparably. If you desire to work with a contracted text window, first disconnect GALE's input hook.

Output, e.g. listing a program, may be aborted by hitting RETURN, temporarily halted by pressing any other key and restarted by striking a non-RETURN key. Sound output (bell) is less raucous than that produced by Applesoft.

GALE is transparent to the user. On installing the program, HIMEM is set to the value as shown in the Memory Usage Table and MAXFILES to 3. An immediate or deferred command that resets HIMEM to a value greater than that noted above or MAXFILES to less than 3 may destroy GALE and/or DOS.

The ampersand vector is not used by GALE, allowing the user to employ this powerful character for his/her own purposes.

When entering control characters, e.g. CTRL-C, depress the control key and the character simultaneously.

SECTION III: IMMEDIATE MODE

Part 1: Overview

Edit mode, macro mode, global commands, hexadecimal/decimal conversion, automatic creation of new line numbers and a HELP SCREEN are all accessed via the immediate mode. A nonflashing cursor (NFC) indicates that a command or portion thereof is expected. Edit commands start with one or two slashes. Global commands begin with a period and an inverse letter or symbol. Deferred mode commands (those in a running program) bypass GALE's input hook, thus allowing normal program operation and testing.

Error messages can originate from three sources: (a) A leading question mark signifies generation from Applesoft ROM. (b) A leading exclamation mark indicates origin from GALE. (c) No leading punctuation mark defines a DOS message.

Syntactic abbreviations used below are self-explanatory except for d which means "delimiter". Parentheses define optional command structure. The hyphen is used for clarity and should not be entered on the screen. No space should exist between a command and its argument. Whereas Applesoft confines users to line numbers smaller that 64000, GALE enables entry of line numbers as large as 65535. Pressing RETURN ends all sequences that require an argument after the command.

Typing PR#slotnumber of printer before entering a command will usually cause the information following the command to be hardcopied. Notable exceptions are the .E and .L global commands which may cause erratic activity of your printer because of control characters. On termination of printer activity, an automatic RESET is executed.

Part 2: Edit Mode Entry Commands

/ linenum

The line whose number follows the slash is presented for editing in a format similar to that of a normal Applesoft listing. If the line is longer

than 251 characters, the message USE // flashes in the MODE section of the edit screen and the user is returned to BASIC. If the entered line number does not exist, a bell rings and the Applesoft prompt reappears.

// linenum

The line whose number follows the double slash is offered for editing in compacted form, i.e. PRINT statements are replaced by ? to conserve space in the input buffer. If the line is still longer than 251 characters, the message TOO LONG flashes in the MODE display and edit mode is aborted. Although GALE cannot create an illegal line length and Applesoft allows only 239 characters per line, with a bit of ingenuity one can produce a line that contains more than 251 characters from the keyboard or the system monitor. The practice is a poor (and unnecessary) one. When encountered, such a line should be broken into two parts either by cursor editing or by rewriting. Again, entry of a nonexistent line is signalled by a beep.

CTRL-G

This command GETS the edit mode. When writing a program line, command, or other text in the immediate mode, pressing CTRL-G transfers the user directly to the edit mode with the cursor placed on the first character in the line. No minimum number of characters is required, i.e. typing CTRL-G directly following the Applesoft prompt presents the edit mode with a blank flashing cursor on the left edge of the screen. A need to make changes in the line being typed or to enter lower case within a REM statement would evoke usage of this option. Implementing the edit mode in this manner disables the RESTORE command in edit mode (see SECTION IV, PART 3) and does not take advantage of the EXHUME command (see this SECTION, PART 3).

Part 3: Global Commands

.A linenum (,increment)

AUTO line numbering facilitates entry of program line numbers in an

orderly sequence. A NFC prompts for the starting line number which is required. The increment defaults to 10 if not specified.

Once AUTO is set, a line number is entered by pressing either the forward or backward arrow immediately following the Applesoft prompt. The forward arrow produces the starting line number and subsequent increments. The backward arrow always prints the expected line number (starting or incremented) minus the increment. If the backward arrow is used to produce the starting line number, odd but predictable numbers may be obtained, e.g. (a) .A10,11 prints 65535 (10−11=−1 which to the Apple means 65535). (b) .A10,1 prints 9 (10−1=9). It is always wiser to use the forward arrow to produce the initial line number. The backward arrow should be employed to repeat a subsequent line number for the purpose of rewriting the line. When a line number has already been printed, if the next line number expected by pressing the forward arrow exceeds 65535, a bell sounds and AUTO is disconnected.

The flexibility of this option is enhanced by the fact that by pressing neither arrow, you may enter any line number desired.

.B

The starting (A) and length (L) parameters for the last BLOADED program are printed. The optimal time for use of this option is immediately after loading a binary program.

.C d-searchstring-d-changestring-(d)-(linenum or linerange)
.C d-searchstring-d-d-(linenum or linerange)

The first CHANGE option allows substitution of one string for another over an optional range of lines or within a single line. Unless a line number or range is required, the ending delimiter may be omitted. The second CHANGE option involves replacing the search string with a null string, i.e. deleting the search string. A flashing cursor prompts for input, and normal Apple ESCAPE editing functions are supported. The backward arrow does not invade the .C command; otherwise, both arrows operate normally. CTRL-C aborts input.

The delimiter may be any of the following ASCII characters

(! " # $ % & ' ( ) * + , − . /). A potential delimiter may be placed within either string provided that another valid delimiter is used. The most popular delimiters are the quotation mark and the slash. To search for the quote, you may use a slash as the delimiter, and vice versa. For example, to change the string "FOR ME" to "FOR US", enter .C/"FOR ME"/"FOR US" not .C""FOR ME""FOR US.

Strings may contain no more than 30 characters. Any control character may be included in a string by preceding its insertion with a CTRL-O (override). If the entry following CTRL-O is not a control character, it is rejected and a bell sounds. Despite the listing formats used by GALE and by Applesoft, spaces may exist only between quotation marks (see above example) and after DATA and REM statements. For example, to change B = 15 to BB = 15, type .C/B=15/BB=15, not .C/B = 15/BB = 15. Failure to adhere to this convention causes undesirable results.

A line range must be separated by a comma and produces an error message if the ending line number is less than or equal to the starting line number. Failure to enter a line number or range results in a complete program search/change.

The search string supports a universal match (wildcard) character, the "at" symbol (@). If found in the change string, @ is interpreted literally.

GALE generates a !LINE #XX TOO LONG message if a replacement causes the input buffer to exceed 249 characters (CHANGE aborts to BASIC) or if a line in program memory detokenizes to more than 249 characters (CHANGE continues at next line). When the replacement moves a program to within $100 (256) bytes of HIMEM, a !MEMORY FULL message terminates the routine with all prior changes remaining intact.

In using either CHANGE option, the user is given an opportunity to visualize each change before it is made. The message CHECK CHANGES (Y/N)? appears. By typing N, all replacements are done automatically. Whereas this process usually occurs rapidly, if numerous changes are required on a single line, a considerable time may elapse before the line is printed, so be patient and do not press RESET. An affirmative response causes GALE to present each change for review. A flashing cursor appears

at the start of the altered line. If you type N, the cursor is replaced by an inverse N, the change is rejected and the original line reappears. Pressing Y converts the cursor to an inverse Y and directs the change to be made in program memory. CTRL-C returns the user to BASIC. Any other reply evokes a beep and the cursor continues prompting for a response. When all changes have been acted upon, immediate mode is reentered.

Output control is described in SECTION II, PART 2.

.D

This command displays the number of free sectors on the diskette in the most recently accessed drive. A sample message would be DISK SPACE FREE (D2) = 164.

.E

This option displays the installed ESCAPE FUNCTION TABLE. Within the realm of GALE, "macro" is synonymous with "escape function". The meaning of special keys is presented on the second display line to aid interpretation of the macro definition (see SECTION V, PARTS 1 and 2). After one screen page of macros is printed, the message PRESS ANY KEY appears. By pressing a key, the next page comes into view. When the macro table is completed, the Applesoft prompt emerges.

.H

The HIDE command transfers the program currently in memory to that location immediately below HIMEM. HIMEM is reset to protect the hidden program, normal program memory is cleared and the message PROGRAM NOW HIDDEN! appears. This option may be utilized for two purposes: (a) Temporarily to tuck away your current program so that new Applesoft code may be written and tested. (b) As a prelude to the APPEND command.

If no program exists in memory, the HIDE command evokes the message !NO PROGRAM IN MEMORY. If a program is already hidden, !PROGRAM ALREADY HIDDEN is displayed. If the program end is within $200 (512) bytes of HIMEM, !MEMORY FULL is printed.

.L linenum

LINEFIND locates the starting and ending RAM addresses of the line whose number is given. It is useful for observing the tokenized line so that the line may be altered within the system monitor. For example, if one desired two or more credit lines (REM statements) with identical numbers at the end of a program, the appropriate line or lines could be located by LINEFIND and the line number bytes (3rd and 4th bytes in the line) changed.

Should the given line number not be present, a beep sounds and Applesoft is reentered.

.M

MANUAL line numbering turns off the AUTO line numbering mode.

.P

POINTERS displays the following Applesoft pointers: (a) START of program. (b) LOMEM (equal to the start of simple variable space and one or two bytes higher than the end of program). (c) Beginning of ARRAY space. (d) Beginning of STRING storage. (e) HIMEM. (f) FREE bytes available for programming (the space between the top of arrays and the bottom of strings). Free space will be meaningful only if the Applesoft program in memory has been run.

Consult the APPLESOFT II BASIC PROGRAMMING REFERENCE MANUAL, pp 140-141 (or the APPLESOFT BASIC PROGRAMMER'S REFERENCE MANUAL, p. 278), for the zero page addresses of the pointers used above.

.R (N-linenum)(,I-increment)(,F-linenum)(,L-linenum)

The RENUMBER command must be followed by at least one (any one) of the four possible parameters: (a) N-linenum is the new starting line number (default = 10). (b) I-increment is the increment between renumbered lines (default = 10). (c) F-linenum is the number of the first line to be renumbered (default = first line of program). (d) L-linenum

is the number of the last line to be renumbered (default = last line of program).

This syntax allows maximal ease and flexibility by using mnemonic letters to indicate parameters, by not demanding a special order of entry, i.e I may precede N, L may precede F, etc., and by permitting all or some lines to be renumbered. The requirement for at least one parameter is a fail safe mechanism to prevent the user from renumbering the program after inadvertently pressing .R-RETURN. To renumber an entire program in increments of 10, a shortcut command is available : .R- (no RETURN is necessary).

Only valid input characters (N, I, F, L, decimal numerals, comma) are accepted. A bell informs you of spurious input. The backward arrow does not invade the .R command, and the forward arrow does not space out of the input field. Should a nonexistent line number be chosen, the next highest line number is selected.

If no program exists in memory, the Applesoft prompt returns following a beep. You are assured of normal operation by observing back-and-forth motion of an asterisk on the command line. Each movement of the asterisk indicates that a line has been renumbered and the entire program has been searched for compatible GOSUB, GOTO, THEN, ON...GOTO, ON...GOSUB, ONERR GOTO, LIST, DEL, and RUN statements which also have been renumbered. If RESET is pressed before RENUMBER ceases, the program effectively is destroyed.

RENUMBER may convert a legal GALE line length (251 detokenized characters or less) into an illegal length by adding numerals to the line number. For example, by changing line number 1 to line number 10000, four characters are added. If the detokenized line length originally was 148 characters, it would be 152 characters after renumbering and could not be presented to edit mode (consult // linenum in this SECTION about how to deal with illegal line length). For the above reason, try to keep line size smaller than 248 detokenized characters.

Six error messages may occur, most for a variety of reasons: (a) !RANGE ERROR is printed when L equals or is less than F, a zero increment is entered, I would cause a line number larger than 65535, or

no line exists in the specified range. (b) !OVERLAP ERROR occurs when any combination of parameters would cause lines out of numeric sequence or duplicate line numbers. (c) !SYNTAX ERROR (GALE) occurs if no parameter is given, the comma is the first character entered or duplicate parameter letters are given. (d) ?SYNTAX ERROR (Applesoft) is generated when no number is placed between a parameter letter and a comma. (e) ?ILLEGAL QUANTITY ERROR results from a line number or increment greater than 65535. (f) !MEMORY FULL occurs when the program end is within $A00 (2560) bytes of HIMEM. This leeway is adequate even for huge programs and is unlikely to prevent you from using RENUMBER. Error examples are listed below.

SAMPLE PROGRAM

```
100 TEXT : HOME
110 INPUT "CHOOSE NUMBER ";N$: PRINT
120 ON VAL (N$) GOSUB 200,300: END
130 PRINT CHR$(7): GOTO 100
200 PRINT "YOU CHOSE 1": RETURN
300 PRINT "YOU CHOSE 2": RETURN
```

EXAMPLES OF RENUMBER ERRORS

!RANGE ERROR

(1) .RF10,L9 or .RF10,L10	(F <= L)
(2) .RN10,I0	(I = 0)
(3) .RF120,N1000,I22000	(old #300 becomes new #6700 which is illegal)
(4) .RF210,L290,N250	(no # in specified range)

!OVERLAP ERROR

(1) .RF110,L200	(since N defaults to 10, old #110 becomes new #10 and #10 cannot follow #100)
(2) .RF120,N60	(old #120 becomes new #60 and #60 cannot follow #110)
(3) .RF110,L200,N120,I100	(old #130 becomes new #320 which is greater than unrenumbered #300)
(4) .RN100,F110	(old #110 becomes new #100 which duplicates unrenumbered #100)
(5) .RN200,I50,F120,L200	(old #200 becomes new #300 which duplicates unrenumbered #300)

!SYNTAX ERROR

(1) .R	(no parameter)
(2) .R,	(leading comma)
(3) .RN100,I20,L200,N50	(N duplicated)

?SYNTAX ERROR

(1) .RN100,F,L500	(no # between F and comma)

?ILLEGAL QUANTITY ERROR

(1) .RF10,L65570,N1000	(L parameter illegal)

.S d-searchstring-(d)-(linerange or linenum)

SEARCH displays the program lines that contain the search string over an optional range of lines or within a single line. Unless a line number or range is required, the ending delimiter may be omitted. A flashing cursor prompts for input, and normal Apple ESCAPE editing functions are supported (ESC-A-F, ESC-I,J,K,M, ESC-@; see APPLE II REFERENCE MANUAL, pp 34-35, or the APPLE //e REFERENCE MANUAL, pp 55-56). The backward arrow does not invade the .S command; otherwise, both arrows operate normally. CTRL-C aborts input.

The delimiter may be any of the following ASCII characters (! " # $ % & ' ( ) * + , - . /). A potential delimiter may be placed within the string provided that another valid delimiter is used. The most popular delimiters are the quotation mark and the slash. To search for the quote, use a slash as the delimiter, and vice versa. For example, to search for the string "FOR ME", enter .S/"FOR ME" not .S""FOR ME".

The search string may contain no more than 30 characters. Any control character may be included in a string by preceding its insertion with a CTRL-O (override). If the entry following CTRL-O is not a control character, it is rejected and a bell sounds. Despite the listing formats used by GALE and by Applesoft, spaces may exist only between quotation marks (see above example) and after DATA and REM statements. For example, to search for B = 15, type .S/B=15, not .S/B = 15.

A line range must be separated by a comma and produces an error message if the ending line number is less than or equal to the starting line number. Failure to enter a line number or range results in a complete program search.

The search string supports a universal match (wildcard) character, the "at" symbol (@).

Output control is described in SECTION II, PART 2.

.U

The UNHIDE command transfers a hidden program to normal program memory. If a program already exists in memory, the message OVER-

WRITE CURRENT PROGRAM (Y/N)? appears. Pressing N aborts UNHIDE and prints PROGRAM STILL HIDDEN!. Hitting Y replaces the current program with the hidden one. A successful restoration of the hidden program produces the message PROGRAM UNHIDDEN!.

If no program is hidden, the message !NO PROGRAM HIDDEN is printed.

.V

VARIABLE CROSS REFERENCE produces an alphabetical (ASCII order) list of all program variables and the numbers of the lines in which each variable occurs. Simple and array variables of each form (real, integer, string) are referenced. If a function variable is defined (DEF FN), it appears at the bottom of the list. Should program memory be empty, a bell sounds and the prompt emerges.

Aside from being an invaluable aid in program development, this option assists in determining if a desired new variable is already present. Be alert to the fact that although names of variables may be different, Applesoft uses only the first two characters to distinguish one name from another.

.X

GALE stores the original version of each line presented to the editor by the / or // commands. While in edit mode, if you change a line and wish to recall the original line, the .X command accomplishes this feat (see SECTION IV, PART 3). On exiting the edit mode, however, the altered line is placed in program memory and the original line cannot be recovered. EXHUME enables you to press .X to print the original line in compacted form. By moving the cursor to the first numeral of the line number (ESC-I,J,K,M) and tracing over the entire line with the forward arrow, the original line may be restored. If edit mode is entered via CTRL-G, the EXHUME buffer will retain the prior line.

If no line has been edited, EXHUME usually prints 0 REM GALE but may display garbage.

.+

APPEND causes a hidden program to be attached to the end of a program in memory. The first line number of the hidden program must be higher than the a last line number of the program in memory, else the following sample error message appears:

```
!OVERLAP ERROR
PART 1: LL = 700
PART 2: FL = 500
```

The error has occurred because the last line (LL) of the program in memory (PART 1) is greater than the first line (FL) of the hidden program (PART 2).

Blank program memory evokes the message !NO PROGRAM IN MEMORY. If no hidden program exists, !NO HIDDEN PROGRAM is printed.

Part 4: Miscellaneous Commands

CTRL-@

Pressing CTRL-SHIFT-P clears the screen and displays all immediate mode commands. This HELP SCREEN moves out of view as scrolling occurs.

decnum

The DECIMAL TO HEX converter gives the hexadecimal equivalent of the decimal argument (decnum). Any number between −65535 and 65535 is accepted. If this range is violated, ?ILLEGAL QUANTITY ERROR is generated. Only decimal numerals and the minus sign are valid input. Leading zeroes are permissible. A !SYNTAX ERROR results from invalid or absent input. A quirk of Applesoft produces a rare ?OUT OF MEMORY ERROR. If this occurs, reenter your number and all will work normally.

$ hexnum

The HEX TO DECIMAL converter gives the decimal equivalent of the given hex number (hexnum). Positive numbers between 0-$FFFF that contain one to four hex digits are accepted. Leading zeroes are allowed. Numbers between $8000-$FFFF produce positive and negative (complementary) values. Any deviation from the above format forces a !SYNTAX ERROR.

<— and —>

See .A in this SECTION.

SECTION IV: EDIT MODE

Part 1: Overview

The GALE method of program line editing is unique in that GALE continually updates the input buffer ($200-$2FF) by reading the line directly from the screen. Other editors first update the input buffer and then reprint the line in fixed format. The advantages of GALE's method are numerous. The line to be edited, once presented, requires no special formatting, thus making readability optimal. No flickering of the line is noted with each alteration. As graphically seen by the remarkable PEEK command (see this SECTION, PART 3), the input buffer is maintained in a continually packed state. Editing a line containing a REM or DATA statement does not create that irritating extra space after the REM or DATA. Whereas Applesoft and other line editors allow spaces within DATA statements, GALE automatically packs all data not enclosed within quotation marks, thus making the final product neater.

Experiment with lines formatted by Applesoft and by GALE. Although many similarities exist, you will be pleased with the differences.

Part 2: Edit Mode Screen

Edit mode screen formatting is illustrated in FIGURE 1. The row beneath the title displays the last issued command and the number of significant line characters. The sample line contains 39 significant characters. The first space after REM never is counted but the space between ACCESS and COMMAND is meaningful. All spaces preceding the REM statement are extraneous to the input buffer but make the line easier to read.

FIGURE 1: Sample EDIT MODE Screen With PEEK On (Cursor On A, < And 0 Inverse):

```
                GALE LINE EDITOR
          MODE: INSERT [ ]      CHARS: 39

 10 D$ =  CHR$ (13) +  CHR$ (4): REM ACCE
 SS COMMAND <

10D$=CHR$(13)+CHR$(4):REMACCESS COMMAND0
```

In contrast to Applesoft which allows a maximum of 239 line characters, GALE permits 251 characters. On reaching the 245th character, the display becomes inverse and a bell sounds with each additional entry. Good programming habit dictates that the line be ended at this point or sooner. When 250 characters are present, MAX flashes in place of the actual count. One additional character is permissible, but on entering the 252nd character, edit mode is terminated, all characters save the final one are preserved and the message ABORT flashes in the MODE display (the flashing is designed to irritate, not to please).

The line to be edited is shown between the two horizontal boundaries. If present, control characters are displayed. An inverse < serves as the end of line (EOL) marker and always is preceded by a space. Entry of code is prevented if the EOL marker would be caused to invade the lower boundary (a situation that may be provoked by playfulness, not need).

The PEEK buffer (see this SECTION, PART 3) is displayed below the lower boundary.

Part 3: Edit Commands

CTRL-A (MODE: UP CASE) — Not Functional in Apple //e Version

The CASE TOGGLE switches between upper and lower case mode.

When lower case is activated, an inverse L appears in the word GALE on the title line. Unless your Apple is equipped with a special adapter, lower case characters are displayed on the screen as apparently meaningless non-alpha "garbage" but are interpreted correctly by printers with lower case capability.

CTRL-C (MODE: COMPACT)

The COMPACT command converts PRINT to its Applesoft abbreviation, ?. This conserves space, thus allowing more characters to be entered in a given line. If an ending DATA or REM statment is followed by text, a trailing space appears.

CTRL-D (MODE: DELETE [ ])

The DELETE command causes disappearance of the character under the cursor, with appearance of that character between the brackets in the MODE display. The cursor does not move, and the remainder of the line is pushed left to fill the space.

CTRL-E (MODE: LINE END)

The END command moves the cursor to the end of the line (one position to the left of the EOL marker). It is valuable as a prelude to adding additional code or, with the backward arrow, to edit the last few line characters.

CTRL-F searchchar (MODE: FIND [ ])

The FIND command is used to locate any control or noncontrol character to the right of the cursor. If the search character is present, the cursor jumps to its next occurrence in the line and the character is placed between the brackets in the MODE display. Subsequent occurrences of the search character are located by again pressing that character. Absence of the search character causes a bell to sound and the mode display to clear.

For example, to find all occurrences of CTRL-D to the right of the cursor, enter CTRL-F followed by CTRL-D repeatedly until the bell is heard. When still active, FIND is terminated by pressing any character

other than the search character. If that entry represents a valid command, it is executed. A complete line search is best accomplished by using the START command to position the cursor at the beginning of the line, and the FIND command to search the entire line except for the character under the cursor. CTRL-X, CTRL-Y and CTRL-Z are used to find \ , [, and __ , respectively. The trade-off is that one cannot search for these three symbolic control characters.

The FIND command is particularly useful for jumping from statement to statement by using : as the search character.

CTRL-I insertchar or insertstring (MODE: INSERT [ ])

The INSERT command allows insertion of one or more noncontrol character to the left of the character under the cursor. With each insertion, the remainder of the line and cursor move right to make room for the new character, which appears between the brackets in the MODE display. To insert control characters, see CTRL-O.

For example, using FIGURE 1 as a reference, you will note by the MODE display that INSERT is active. Typing the word DISK followed by a space will make the REM statement more descriptive. Insertion mode is terminated by entry of any control character. If that control character represents a valid command, the command is executed; otherwise, the MODE display clears and the editor awaits another command.

CTRL-K (MODE: KILL)

The KILL command prints \ and aborts to immediate mode with the line unchanged from its original form.

CTRL-O ctrlchar (MODE: CTRL ENTRY [ ])

The OVERRIDE command allows insertion of one control character behind the cursor. A NFC prompts for input. The remainder of the line and cursor move right to make room for the new character, which appears between the brackets of the MODE display. The OVERRIDE command becomes inactive following the insertion.

Characters such as CTRL-M (RETURN) and CTRL-J (linefeed) will assist you in formatting REM statements.

CTRL-P (MODE: BUFFER)

The unique PEEK command toggles a display of the contents of the input buffer below the lower boundary line (see FIGURE 2). PEEK is an instructive method of watching GALE's editor in action and observing how the input buffer always remains packed. If the display is on when the edit mode is exited, a hex printout of the tokenized input buffer is presented along with a character count (see FIGURE 2). By comparing the untokenized character count to the tokenized count, one usually finds the latter to be smaller. Tokenization is Applesoft's method of conserving space by replacing keywords with tokens. /un/Consult the APPLESOFT II BASIC PROGRAMMING REFERENCE MANUAL, pp 121, 138-139, or the APPLESOFT BASIC PROGRAMMER'S REFERENCE MANUAL, Volume 2, pp 280-282, for a keyword table and ASCII character codes.

If the COMPACT, RESTORE or HELP SCREEN option is invoked, the PEEK buffer disappears.

FIGURE 2: Sample Screen On Completion of EDIT MODE (PEEK ON) (< And 0 Inverse):

```
                GALE LINE EDITOR
MODE:                          CHARS: 11
-------------------------------------------

20  FOR I = 1 TO 5 <

TOKENIZED INPUT BUFFER
32308149D031C13500
 CHARS: 8
```

CTRL-Q (MODE: DONE)

The QUIT command deletes all characters from the cursor to the EOL marker. Edit mode is exited with preservation of all code behind the

cursor. Lines with illegal line numbers may readily be deleted by positioning the cursor between the line number and the first line character and pressing CTRL-Q.

CTRL-R (MODE: RESTORE)

The RESTORE command recalls the original line, thus negating all changes. The cursor is repositioned on the first character after the line number. If the edit mode was entered by CTRL-G, the RESTORE command is disabled.

CTRL-S (MODE: LINE BEG)

The START command places the cursor on the first character in the line. It is convenient as a prelude to changing the line number, deleting a line with an illegal number and searching the entire line by using the FIND command.

CTRL-T (MODE: TRUNCATE)

The TRUNCATE command deletes all characters from the cursor to the EOL marker. Unlike the QUIT command, edit mode is preserved.

CTRL-Z zapchar (MODE: ZAP [ ])

The ZAP command deletes all characters from the cursor to the zap character to the right of the cursor. The cursor appears on the zap character, which is now shown between the brackets in the MODE display. Pressing the zap character again performs another deletion. If no zap character exists to the right of the cursor, a bell rings and the MODE display clears.

For example, to delete two statements from a line, position the cursor on the first character of the initial statement to be zapped, and type CTRL-Z : :. When still active, ZAP is terminated by pressing any character other than the zap character. If that entry represents a valid command, it is executed.

CTRL-X, CTRL-Y and CTRL-Z are used to designate \ , [, and _ , respectively, as the zap character. Of course, these three symbolic control characters cannot be used as zap characters.

ESC indexchar (MODE: MACRO [ ])

Pressing ESCAPE followed by a defined ESCAPE FUNCTION key, i.e. entering a macro, inserts the macro to the left of the cursor. The remainder of the line and cursor move forward to accommodate the new code, and the function key appears between the brackets in the MODE display. The MACRO INSERT mode becomes inactive following the insertion.

Any macro is available, regardless of whether it is recursive or it ends with a carriage return (see SECTION V, PARTS 1 and 2).

The availability of macros in both immediate and edit modes is a powerful feature of GALE.

CTRL-^ , (MODE: FLIP)

Pressing CTRL-^ , replaces the edit mode screen with the screen present before entering edit mode. The very next keypress returns you to edit mode. This FLIP option is convenient for reviewing the contents of a line that was erased by the edit mode screen. Exiting to the original text page one is accomplished by the .X command which is described below in this Part.

RETURN (MODE: DONE)

Pressing RETURN (CTRL-M) ends edit mode and places the line into program memory.

CTRL-X

This alternative method of EXIT from the edit mode places the line in program memory and returns you to the screen page that was present on entering edit mode.

——> and <—— (MODE: ——>) (MODE: <——)

The forward arrow (CTRL-U) and backward arrow (CTRL-H) work normally except that the former does not impinge upon the EOL marker and the latter does not back into the upper boundary line.

CTRL-@

Pressing CTRL-@ toggles a HELP SCREEN display beneath the lower boundary line. All edit mode commands are listed. If the PEEK buffer (CTRL-P) is entered, the HELP SCREEN disappears.

Part 4: Typeover (MODE: TYPE [ ])

If none of the edit commands (see PART 3 of this SECTION) is entered, typing a noncontrol character replaces the character under the cursor. The new character appears within the brackets in the MODE display. Control characters are inserted by using the OVERRIDE command (see this SECTION, PART 3).

SECTION V: ESCAPE FUNCTIONS (MACRO MODE)

Part 1: Overview

ESCAPE FUNCTIONS are user-definable commands that may be elicited by pressing the ESCAPE key followed by a character (index code) that evokes the function. A NFC prompts for the index code. Typing an undefined key ends macro mode, and a flashing cursor returns. ESCAPE FUNCTIONS are stored in GALE's memory as strings of characters that are printed on the screen exactly as if they had been typed by the user. Pressing .E in immediate mode (see SECTION II, PART 3) lists all ESCAPE FUNCTIONS, i.e. the index codes followed by the character strings.

In defining ESCAPE FUNCTIONS, a macro command is created by placing a carriage return (CR) at the end of the string. Recursion is the ability of a macro to call itself, i.e. to remain in macro mode when the function is completed. Recursive function is defined by placing an asterisk (*) at the end of a character string and is particularly valuable in manipulating cursor movements.

Macros may be employed for numerous purposes. A partial list includes: (a) Inserting commonly used words or phrases into program lines in immediate and edit modes. (b) Issuing standard Applesoft and DOS commands in immediate mode. (c) Evoking creative and useful commands in immediate mode. (d) Controlling cursor movement in immediate mode.

Part 2: MACROSCOPE

This machine language program requires no Applesoft interface. Run it with the command BRUN MACROSCOPE (with the appropriate suffix, if necessary), or, if GALE is installed, press ESC-@. To move MACROSCOPE to your work diskette, the FID program on the DOS

3.3 SYSTEM MASTER diskette may be used. Running MACROSCOPE destroys the Applesoft program in memory, so save it before using MACROSCOPE.

A brief explanation of how GALE is installed is useful when employing the LOAD and SAVE options of MACROSCOPE. When GALE is BRUN, it is loaded at $3200, the DOS buffers are moved down and the program is transferred into the freed space between DOS and its buffers where it becomes activated. When MACROSCOPE refers to the "currently installed program," it denotes the active GALE program. When MACROSCOPE loads GALE from the diskette, it loads the inactive version, which can be saved back to the diskette.

Temporary macros are created by loading the currently installed macro table into MACROSCOPE's edit buffer, adding or changing macro definitions and saving the altered table back to the active GALE program. Permanent changes must be made to the GALE program on the diskette. GALE first is loaded, its macro table comes to reside in the MACROSCOPE edit buffer and, after making the desired changes, the entire GALE program must be saved back to the diskette.

On running MACROSCOPE a menu is presented, below which a count of free and used bytes is listed. No more than 512 bytes can exist in the macro table. At any time, pressing RESET (autostart ROM)) returns you to the menu; in fact, the only method of terminating the program is to choose the proper menu selection or to turn off the power. In most cases, pressing RETURN at the beginning of an input sequence or typing CTRL-C during the input also places you back in the menu.

Your first step should be to employ the LOAD option to move a macro table into the MACROSCOPE buffer. If this is not done, an attempt to choose an option other than EXIT produces the error message !NO FILE LOADED. Each menu choice is explained below:

(1) **A - ADD ESCAPE FUNCTION**: This option evokes a HELP SCREEN. The following important characters are defined: CTRL-[(ESCAPE), CTRL-M (CR), CTRL-U (forward arrow), CTRL-H (backward arrow) and * (recursion). Special characters include the final four listed above plus] ([), / (\ ), – ( ), CTRL-C and CTRL-O. To en-

ter a special character, press CTRL-O followed by the desired character. @cmd\@CTRL-[may be entered directly by hitting the ESCAPE key. Free bytes are displayed below the HELP SCREEN and are continually updated as macro definitions are constructed.

Creation of a macro is prompted by the message ADD WHICH KEY? RETURN (CTRL-M) is the only key that permits direct escape from this input request. Any character except CTRL-M is acceptable and may be typed without use of CTRL-O. The character that you enter is the index code for the macro. If your chosen character already is in use, an error message is shown. For example, if A is pressed, the message A ALREADY DEFINED! appears, and by pressing any key you are prompted for another selection.

To redefine an existing macro, it first must be deleted. Entry of an unused character evokes the message DEFINE FUNCTION:. Creation of the macro is simple — just type it in. The forward arrow is disabled and the back arrow is destructive. No more than 254 characters are allowed. A bell warns when the 251st character is entered. If the limit is exceeded the message MORE THAN 254 CHARACTERS! appears and you must start again. Remember that CTRL-O must precede entry of special characters (NFC appears). For example, to place a CR after a macro, type CTRL-O CTRL-M. When RETURN is pressed, the defined macro is added to the edit buffer of MACROSCOPE.

In defining macros, when five or fewer free bytes remain and another character is typed, a warning bell sounds. If no bytes remain, further entry generates the message !MACRO TABLE LIMIT EXCEEDED. On pressing any key, the menu is reentered.

To make a macro recursive, CTRL-O * is required. Since no character may follow the symbol for recursion, the cursor disappears and two options exist. CTRL-C aborts the input, while any other key places the definition in the edit buffer.

Avoid using ESCAPE as the final character in a macro definition. It serves no purpose and may create a flashing index character which you cannot access or delete.

An example of creative macro programming is the definition

CALL−375: CALL−365: CALL 1002 CTRL-M. Its function is to disconnect GALE's hooks by resetting normal hooks. The calls to $FE89 (−375) and $FE93 (−365) are equivalent to IN#0 and PR#0, respectively. The final call notifies DOS, and CTRL-M is a CR. Define the X key (mnemonic for "exit GALE") in this manner and you will have a valuable new macro. As you can see, the possibilities are vast.

(2) **D - DELETE ESCAPE FUNCTION**: On pressing D from the menu, the query DELETE ENTIRE TABLE (Y/N)? appears. A Y response produces CONFIRM DELETION (Y/N)?. Rejection of the confirmation causes the menu to reappear, whereas an affirmative answer wipes out the entire macro table. An N reply to the initial question evokes DELETE WHICH KEY? RETURN is the only key that allows direct escape from this input request. If the character chosen (index code) has not been defined, a NOT DEFINED message appears, and pressing any key returns you to the last input request. Selecting a defined key prints the macro character string and CONFIRM DELETION (Y/N)? Confirmation deletes the macro and returns you to the menu. A negative answer allows you another chance to delete a macro. The display of free and used bytes is not updated until the menu is reentered.

(3) **E - LIST ESCAPE FUNCTION**: As with the .E command in immediate mode (see SECTION II), a list of all macros, one screen at a time, is presented in ascending ASCII sequence. The index code key appears on the left of each line, and the character string follows. A long macro definition at the bottom of a screen may cause scrolling to obliterate the top macro on the screen. Arrange the macro table carefully to prevent this from happening.

TABLE EMPTY! appears if no macro has been defined.

(4) **K - LIST DEFINED KEYS**: This convenient option simply shows the index code characters that are defined currently.

(5) **L - LOAD GALE FILE**: All sessions of macro editing must begin with this option. When L is pressed, all or part of the following submenu appears:

GALE MACRO TABLE TO LOAD:
CURRENTLY INSTALLED (C)
PROGRAM ON DISKETTE (P)
TABLE ON DISKETTE (T)

SELECTION:

If GALE is not active, the first option is not presented. Assuming that you have run MACROSCOPE with GALE installed, choosing C loads GALE's escape table into the MACROSCOPE edit buffer and recalls the main menu.

Pressing P replaces the submenu with NAME OF PROGRAM TO LOAD:. "Program" means the inactive diskette version of GALE. Typing GALE (assuming you have not renamed her) and pressing RETURN loads the program into RAM with its macro table in the MACROSCOPE edit buffer. Typing the name of a nonexistent program produces the error message FILE NOT FOUND. In either instance, the menu is reentered.

Pressing T replaces the submenu with the message NAME OF TABLE TO LOAD:. "Table" refers to a diskette macro table. By typing the name of the desired table and pushing RETURN, the table comes to rest in the edit buffer and the menu is returned to the screen.

(6) S - SAVE GALE FILE: For a clear understanding of this option, please read the LOAD option first. To become effective, additions and deletions must be saved.

Pressing S brings all or part of the following submenu onto the screen:

GALE MACRO TABLE TO SAVE:
CURRENTLY INSTALLED (C)
PROGRAM ON DISKETTE (P)
TABLE ON DISKETTE (T)

SELECTION:

If GALE is not installed, the first option is not printed. With GALE active, pressing C replaces GALE's macro table with the table in the MACROSCOPE edit buffer, thus creating temporary macros. The main menu is reentered.

If the GALE diskette program was the last option loaded, typing P prints NAME OF PROGRAM TO SAVE: GALE? (assuming that you have loaded the file name GALE). The cursor is positioned on the question mark. Pressing the Y key saves GALE forthwith. Pressing any other key moves the cursor to the first letter of the name (G), where pressing RETURN terminates the input sequence, or saving GALE under another name may be accomplished by typing that name followed by RETURN. If the GALE diskette program was not the last option loaded, no name appears after the request. Simply type a name, press RETURN and the program will be saved.

If you desired to load GALE but mistakenly chose another LOAD option, you are likely to discover the error when no name of the loaded table is printed automatically. If so, to preserve your editing session and end up with a new permanent macro table within GALE, employ these steps: (a) Press CTRL-C to enter the main menu. (b) Using the SAVE option, press T. (c) Type a name for the inadvertently loaded and edited macro table, press RETURN, and the MACROSCOPE edit buffer is saved as a new table. (d) Load GALE by using the P option of the LOAD submenu, typing GALE and pressing RETURN. (e) Load the newly created macro table by pressing the T option of the LOAD submenu and typing the name assigned to the table. (f) Save GALE (the new table is contained within GALE) by pressing the P option and typing Y. It may seem confusing initially, but by purposely making the error and by recovering from that error as described above, you will learn much about the flexibility of macro table interchange. Please use a noncritical diskette for the experiment!

Should you choose the P SAVE option, be certain that the P LOAD option was invoked at some time during the current editing session, otherwise the GALE program on diskette effectively will be destroyed or a nonfunctional program will be written.

Why should you consider renaming GALE? The only important reason is that you may prefer to have two versions of the entire program, each with a different macro table. In practice, however, this is less wieldy than transferring macro tables from one location to another.

Pressing T replaces the submenu with the message NAME OF TABLE TO SAVE:. If you had loaded a table, its name and a question mark is printed after the above message, and by pressing Y you may save the table under the name indicated. Pressing any other key positions the cursor on the first letter of the name and permits escape by pressing RETURN or typing another name and pressing the RETURN key. In this situation, inadvertent loading of the GALE diskette program instead of the macro table causes no problem, even if the mistake goes undetected. If you had loaded the currently installed table or the GALE diskette program, no name appears after the above message. Simply type a name, press RETURN and the table will be saved.

(7) **C – CATALOG DISKETTE**: Pressing C will catalog the diskette in the current drive. After viewing the directory, press any key to recall the menu.

(8) **X – EXIT MACROSCOPE**: Pressing X prints the message BACK FOR "SAVE" OPTION (Y/N)?. If you have forgotten to save your work, this is your final chance. Pressing Y returns you to the menu, whereas pressing N exits MACROSCOPE.

Short of turning the power off, this option is the only method of exiting MACROSCOPE. Since an automatic FP is performed, a preexisting Applesoft program will have been wiped out.

Part 3: Standard Macros

The ESCAPE FUNCTIONS listed in SECTION VIII, PART 3 are contained within GALE's macro table and also are present as the table ESCTBL.STND on the Program Diskette. Hyphens are used for clarity and commas indicate that the prior character is repeated. CTRL-M (CR),

CTRL-H (back arrow), CTRL-U (forward arrow), and * (recursive final character) are frequently used symbols.

Several definitions require explanation. Cursor functions employing the standard ESC-A,B,C,D sequences are nonrecursive. Cursor functions using the standard ESC-I,J,K,M sequences are recursive. Because of near redundancy, you may wish to redefine the nonrecursive set. ESC-CTRL-U (ESC- ——>) and ESC-CTRL-H (ESC- <——) move the cursor six columns forward and backward, respectively. ESC-^ recursively moves the cursor up six rows. ESC-Z prints the address defined by the variable Z. For example, if you type Z = 115 followed by RETURN in the immediate mode, and then type ESC-Z, HIMEM is printed (115 is the pointer for HIMEM). As you see, nearly any imaginable function probably can be defined as a macro.

SECTION VI: OTHER FILES ON PROGRAM DISKETTE

Part 1: GALE Out

Although GALE may be disconnected by the IN#0 and PR#0 commands (see SECTION II, PART 2), the program remains intact between DOS and its buffers. If more memory is required to run a huge program, complete removal of GALE is in order. The command BRUN GALE OUT (with the proper suffix, if necessary) destroys GALE, resets HIMEM to $9600 (38400), normalizes the RESET vector, and prints GALE OUT!. An Applesoft program in memory is not destroyed.

To move this short binary program to your work diskette, use FID on the DOS 3.3 SYSTEM MASTER diskette.

Part 2: HELLO

The sample HELLO program on the GALE disk determines the type and memory size of your computer, prints GALE BEING INSTALLED, and runs the proper version of GALE. Normally, only BASIC programs can be booted. Should you prefer to run a binary program like GALE directly from DOS 3.3 boot code, a minor patch is required. Follow these instructions: (a) Boot the DOS 3.3 SYSTEM MASTER diskette. (b) POKE 40514,52 from Applesoft or type 9E42:34 from the system monitor. (c) Insert a work diskette containing no important material and type INIT GALE. (d) Following initialization, type DELETE GALE. (e) Transfer GALE to the newly created diskette by using one of the methods described in SECTION II, PART 2. You now have a DOS 3.3 work diskette that will BRUN GALE on booting.

Part 3: ESCTBL.ALT

An alternate diskette macro table is provided on the GALE disk to demonstrate commands not found on ESCTBL.STND, which is discussed in SECTION V, PART 3. Two macros are worthy of comment: (a) ESC-CTRL-T prints TEXT: HOME: POKE−16298,0: POKE−16300,0: POKE−16368,0. In many programs, this is an excellent starting line because it sets a normal text window, clears the screen and homes the cursor, turns off HIRES, sets text page one, and resets the keyboard strobe. (b) ESC-CTRL-W prints W=20−(LEN(T$)/2): IF W THEN HTAB(W1): PRINT T$: RETURN. This subroutine centers text assigned to the variable T$. If the next program line is PRINT T$: RETURN, you have a complete subroutine that centers literals containing 40 characters or less and prints longer strings flush left. Assign the latter line to another macro if you wish.

ESCTBL.ALT also includes ideas for graphic and DOS file commands. Eventually, you will devise primary (within the diskette GALE program) and secondary (diskette macro table) macro tables that best suit your needs.

Part 4: GALE.64

This version of GALE is designed to work with computers which have 64K or more RAM. This includes Apples with RAM cards of 16K or larger in any slot. GALE.64 resides entirely in upper RAM, leaving motherboard RAM for DOS and your programs.

When using GALE.64, you should run the versions of GALE OUT, MACROSCOPE, and ESCTBL which have the .64 suffix.

NOTE: When using the GALE.64 and GALE //E.40 versions, do not press the RESET key while performing GALE functions. The system may hang.

Part 5: GALE//E.40

This version of GALE is designed to work on the 40-column screen of the Apple //e. GALE//E.40 differs from the other versions in the following respects:

1. General:

(a) Most non-error message text is in mixed upper/lower case.
(b) Cursor appearance and function are compatible with the new monitor input routine (KEYIN).

2. Edit Mode:

(a) The DELETE key deletes the character under the cursor.
(b) The TAB key activates insert mode (as does CTL-I).
(c) All four arrows move the cursor within the active edit window (between the beginning of the line and the EOL marker).
(d) CTL- \ replaces CTL-K as the kill line command.
(e) The case toggle (CTL-A) has been removed.
(f) The help screen reflects the above changes.

When using GALE //E.40, you should run the versions of GALE OUT and MACROSCOPE which have the //E.40 suffix.

Part 6: GALE.AMP

This version of GALE is designed to work with AmperSoft, version 2.3 (MicroSPARC, Inc.), a programmer's developmental tool that moves DOS 3.3 into an extra memory bank, e.g. a RAMCARD in slot zero, and provides enhanced Applesoft and DOS commands.

GALE.AMP must be installed by typing BRUN GALE.AMP. If AmperSoft is not in place, the message AmperSoft NOT INSTALLED appears. Successful installation places GALE.AMP above HIMEM

($9200) and beneath the AmperSoft RAM vector table ($BD00-$BFFF).

AmperSoft features a FREE SECTOR display when the diskette is cataloged and a new DOS command, PADDR, which prints the starting address and length of the last BLOADed program. For these reasons, the global commands .B and .D (see SECTION III. PART 3) are not found in GALE.AMP.

GALE.AMP may be moved to another diskette by employing the FID program on the DOS 3.3 SYSTEM MASTER diskette or by the sequence: (a) BLOAD GALE.AMP, (b) BSAVE GALE.AMP, A$31B7, L$2A75.

The programs GALE.AMP OUT and MACROSCOPE.AMP must be used in conjunction with GALE.AMP. Parameters are listed in SECTION VII, PART 3. ESCTBL.48STD (see SECTION V, PART 3) and ESCTBL.48ALT (see this SECTION, PART 3) are supported by GALE.AMP.

Part 7: GALE MACRO PRINTERS

A hardcopy of the macro table of any version of GALE may be obtained by utilizing GALE MACRO PRINTER. It may be used whether or not GALE is installed. The printer should recognize the code CTRL-I 80N to set page width. To transfer the program to another diskette, use FID on the DOS 3.3 SYSTEM MASTER diskette or follow these steps: (a) BLOAD GALE MACRO PRINTER, (b) BSAVE GALE MACRO PRINTER, A$6000, L$457.

The first step is to BLOAD the desired version of GALE or its compatible ESCTBL. BRUN GALE MACRO PRINTER to activate the program. The following questionnaire ensures maximal printing flexibility:

(1) **HAS GALE BEEN BLOADED (Y/N)? Y.** As with all subsequent input, the default response ("Y" in this instance) may be accepted by hitting RETURN, and the program may be aborted by pressing CTRL-C. A negative reply returns you to BASIC and allows you to BLOAD the

GALE program, at which point GALE MACRO PRINTER must again be run. An affirmative answer evokes the next request.

(2) **48K, 64K, OR AMP VERSION (4/6/A)? 6.** Pressing RETURN (64K default — also used for the IIe version), 4 (48K version), or A (AmperSoft version) sets the starting address of the desired macro table.

(3) **PRINTER SLOT (1-7)? 1.** The slot number containing the printer interface card should be selected.

(4) **AUTO OR MANUAL FORMFEED (A/M)? A.** Continuous sheet paper makes the auto option convenient. Single sheet paper requires a manual formfeed.

(5) **LINEFEEDS PER PAGE (30-60)? 60.** The program counts carriage returns. Output is double-spaced. On standard 9.5 by 11 inch paper (66 lines per page), the default option (60) is most efficient. If shorter sheets are used, choose the option that best fits your needs.

(6) **CHARACTERS PER LINE (40-80)? 80.** Standard paper accepts 80 column output. Shorter line length is available for narrower paper.

On completing the above queries, the message TURN PRINTER ON AND PRESS ANY KEY appears. Even at this point, CTRL-C will abort the program. If you proceed, turn the printer on and set the "top of form" switch when the printer head is positioned several lines below the top of the page. Pressing any key starts the printout, striking a non-ESCAPE key causes printing to pause, pressing a non-ESCAPE key again causes printing to resume, and hitting ESCAPE aborts output and returns you to Applesoft. With auto formfeed, once the printout begins you may relax until the printer stops. If paper insertion is performed manually, as each page is completed, the message INSERT NEW PAGE flashes. Once the new sheet is in place, striking any key causes the flashing message to

disappear and printing to resume. When the Applesoft prompt reappears, the job is finished.

The last page of the printout contains a convenient key to the symbols employed. Upper case indicates non-control characters, lower case denotes alpha control characters, and special symbols represent important non-alpha control characters.

SECTION VII: MEMORY USAGE

Part 1: Memory Map — GALE.48

	GALE ACTIVE	GALE INACT	MACROSCOPE	GALE OUT
ZERO PAGE	$06–$09, $1E–$1F	none	$06–$09	$06–$07
EXT. BUFF.	none	none	$2000–$20FF $2800–$28FF $5929–$5B29	none
HIMEM	$6C00	unaffected	unaffected	$9600
INT. BUFF.	$7300–$74FF	none	none	none
START	$7500	$3200	$5C00	$370
END	$9C00	$5B2A	$6979	$3CE
LENGTH	$2701	$2926	$D7A	$5F

Memory Map — GALE.64

	GALE.64 ACT	GALE.64 INACT	MACROSC.64	GALE.64 OUT
ZERO PAGE	$06–$09, $1A–$1B $1E–$1F	none	$06–$09	$06–$07
EXT. BUFF.	variable	none	$7BCE-$7FCE $8000-$83FF	none
HIMEM	$9600	unaffected	unaffected	$9600
INT. BUFF.	$D000–$D1FF	none	none	none
START	$D000	$5000	$8600	$360
END	$FF00	$7FCE	$93E9	$3C7
LENGTH	$2F01	$2FCF	$DEA	$68

Memory Map — GALE.AMP

	GALE.AMP ACT	GALE.AMP INACT	MACROSC.AMP	GALE.AMP OUT
ZERO PAGE	$06–$09, $1A–$1B $1E–$1F	none	$06–$09	$06–$07
EXT. BUFF.	variable	none	$5A2B-$5C2B $7000-$73FF	none
HIMEM	$9200	unaffected	unaffected	$9200
INT. BUFF.	$9200–$93FF	none	none	none
START	$9200	$31B7	$5D00	$360
END	$BC00	$5C2B	$6ADA	$3C6
LENGTH	$2A01	$2A75	$DDB	$67

MEMORY MAP — GALE//E.40

	G//E.40 ACT	G//E.40 INACT	MACROSC//E.40	G//E.40 OUT
ZERO PAGE	$06–$09, $1A–$1B $1E–$1F	none	$06–$09	$06–$07
EXT. BUFF.	variable	none	$7BB9–$7FB9 $8000–$83FF	none
HIMEM	$9600	unaffected	unaffected	$9600
INT.BUFF.	$D000-$D1FF	none	none	none
START	$D000	$5000	$8600	$360
END	$FF00	$7FB9	$93D1	$3CB
LENGTH	$2F01	$2FBA	$DD2	$6C

Part 2: Free Area

Just as $300−$3CF is an excellent area to tuck away short machine language programs. GALE possesses a free area ($9C01−$9CEF) which also may be used to locate similar programs. This convenient space is protected only if GALE is installed. Running GALE OUT causes the DOS buffers to overwrite this segment of memory.

SECTION VIII: SUMMARY OF COMMANDS

Part 1: Immediate Mode Commands

COMMAND	MEANING
/linenum	NORMAL edit
// linenum	COMPACTED edit
CTRL-G	GET edit directly
.A linenum (,inc)	AUTO numbering
.B	BLOAD parameters
.C d-searchstr-d-changestr-(d)-(linenum or range)	CHANGE — regular
.C d-searchstr-d-d-(linenum or range)	CHANGE — delete
.D	DISK space free
.E	ESCAPE function
.H	HIDE
.L linenum	LINEFIND
.M	MANUAL numbering
.P	POINTERS
.R (N-linenum)(,I-inc)(,F-linenum)(,L-linenum)	RENUMBER
.S d-searchstr-(d)-(linenum or range)	SEARCH
.U	UNHIDE
.V	VARIABLE XREF
.X	EXHUME
.+	APPEND
CTRL-@	HELP screen
# decnum	DECIMAL to hex
$ hexnum	HEX to decimal
<— and —>	See .A

Part 2: Edit Mode Commands

COMMAND	MEANING
CTRL–A	CASE TOGGLE
CTRL–C	COMPACT
CTRL–D	DELETE
CTRL–E	END OF LINE
CTRL–F	FIND
CTRL–I	INSERT
CTRL–K	KILL
CTRL–O	OVERRIDE
CTRL–P	PEEK
CTRL–Q	QUIT
CTRL–R	RESTORE
CTRL–S	START OF LINE
CTRL–T	TRUNCATE
CTRL–Z	ZAP
CTRL–@	HELP SCREEN
CTRL–^	FLIP
ESC	MACRO INSERT
RETURN	EXIT EDIT MODE
NONE OF ABOVE	TYPEOVER

Part 3: Standard Escape Functions (MACROS)

CODE	CHARACTER STRING	CODE	CHARACTER STRING
CTRL-@	BRUN MACROSCOPE	A	ESC-A
CTRL-B	BLOAD	B	ESC-B
CTRL-C	CHR$(	C	ESC-C
CTRL-D	DELETE	D	ESC-D
CTRL-F	FLASH	E	ESC-E
CTRL-G	GOSUB	F	ESC-F
CTRL-H	CTRL-H,H,H,H,H,H,H,H- *	G	GOTO
CTRL-I	FOR I = 1 TO	H	HTAB
CTRL-J	FOR J = 1 TO	I	ESC-D- *
CTRL-K	MID$(	J	ESC-B- *
CTRL-L	LEFT$(	K	ESC-A- *
CTRL-N	NORMAL	L	LIST-CTRL-M
CTRL-O	INVERSE	M	ESC-C- *
CTRL-P	POKE	N	ESC-C,C,C,C,C,C- *
CTRL-R	RIGHT$(	O	ONERR GOTO
CTRL-S	STR$(	P	PEEK(
CTRL-U	CTRL-U,U,U,U,U,U,U,U- *	Q	HOME-CTRL-M
—	—	R	RETURN
/	\	S	SAVE
0	LIST	T	TEXT
1	CATALOGD1-CTRL-M	U	UNLOCK
2	CATALOGD2-CTRL-M	V	VTAB
:	CALL-151-CTRL-M	Z	?PEEK(Z)PEEK(Z1) *256-CTRL-M
<	D$=CHR$(4)	]	[
>	D$=CHR$(13)+CHR$(4)	^	ESC-D,D,D,D,D,D- *

* Recursive function.

APPENDIX A

A Sample GALE Session

This Tutorial is a step-by-step walkthrough designed to help you become familiar with the use of GALE as quickly as possible.

We will cover most but not all of the major GALE commands and functions here.

Before you continue, you must make sure that you have completed Section II of this manual: Startup Information which tells you how to install GALE on your computer.

Let's Get Started!

First you have to load GALE into your Apple. To do this place your GALE disk into your disk drive and turn your Apple on. You will see a disk catalog displayed on the screen, and the message GALE.48 BEING INSTALLED (GALE.64 if you have a 64K computer; GALE//E.40 if you are using an Apple //e), followed by the Applesoft prompt. GALE is now ready.

Remove the GALE disk from the drive an replace it with the disk on which you want to store your program. We will be using the following short Applesoft BASIC program to demonstrate GALE (enter it exactly as shown, including the apparent mistakes):

```
10  REM GALE DEMONSTRATION PROGRAM
20  PRINT "THIS PROGRAM DRAWA A TRIANGLE ON
    THE SCREEN"
30  PRINT "FROM THREE SETS OF COORDINATES THAT
    YOU ENTER"
```

```
40   PRINT "ENTER THE COORDINATES IN THE
     FOLLOWING FORM:"
50   PRINT " X,Y AND PRESS THE RETURN KEY AFTER
     XY PAIR"
60   PRINT "MAKE SURE THAT YOUR X VALUES DO
     NOT EXCEED 279"
70   PRINT "AND YOUR Y VALUES DO NOT EXCEED 159"
80   INPUT "ENTER YOUR FIRST COORDINATES ";X1,Y1
90   INPUT "ENTER YOUR SECOND COORDINATES ";
     X2,Y2
100  INPUT "ENTER YOUR THIRD COORDINATES ";X3,Y3
110  HIME: HGR: HCOLOR=3
120  HPLOT X1,Y1 TO X2,Y2 TO X3,Y3 TO X1,Y1
```

Let's start by telling GALE that we want to use automatic line numbering. To do this type the following:

.A10

A tells GALE to turn on automatic line numbering, and the 10 tells GALE to start with line number 10. Automatic line numbering is controlled by the Apple's left and right arrow keys. Pressing the right arrow key gives you the next line number, and the left arrow key gives you the previous line number.

Now press the right arrow key, and the line number 10 will appear on the screen. Enter line 10 of the program now, and press the RETURN key. Before you type in the second line, press the right arrow key. You will see the next line number, which is 20, appear on the screen.

Enter the next line of the program exactly as it appears below, mistakes and all, without pressing the RETURN key:

```
20   PRINT"THIS PROGRAM DRAWA A TRIANGLE ON
     THE SCREEN"
```

Editing the Line We Have Just Entered

As you can see, there is a mistake in the line — the word DRAW.

Press CTRL-G. The screen will change and you will see the line at the top of the screen with the cursor flashing at the beginning of the line, like this:

```
                GALE LINE EDITOR
                MODE:  CHARS: 48
-------------------------------------------------
  20 ? "THIS PROGRAM DRAWA A TRIANGLE ON
  THE SCREEN" <
-------------------------------------------------
```

You are now in Edit mode. Move the cursor to the A at the end of DRAWA using the right arrow key, and press S. The A will be replaced by S. Press the RETURN key, and the cursor will jump down to the bottom of the screen. The message DONE will appear next to MODE.

LIST the program, and you will see that the error has been corrected.

There are three ways to edit a line of the program. The one we have just used is called the IMMEDIATE mode, which means that we were already typing that line when we decided to edit it.

Enter the rest of the lines of the program EXACTLY as they are listed above (mistakes and all. Don't forget to press the right arrow key each time you press the RETURN key to select the next line number.

Editing a Line That Has Already Been Entered

Line 110 has an error in it. Let's fix it now. Type the following:

/110

and press the RETURN key. You will see the following on your screen:

```
GALE LINE EDITOR
MODE:     CHARS: 20
-------------------------------------------
110 HIME: HGR : HCOLOR= 3 <
-------------------------------------------
```

We are now back in the Edit mode. There is a difference here, though. The cursor is not at the very beginning of the line. Instead it is positioned at the first character of the line that is not part of the line number.

We can edit this line in the same way that we edited line 20. Move the cursor to the I with the right arrow key. Press O and the RETURN key. The message DONE will appear next to MODE at the top of the screen, and the cursor will appear at the bottom of the screen.

The CHANGE Command

Let's assume that you decide that you don't like one of your variable names, and you want to change it. GALE gives you pwoer to do this quickly throughout the program with the CHANGE command.

Let's change the variable name X1 to XRAY. Here's how to do it: Type the following:

.C/X1/XRAY/

and press the RETURN key. The message: CHECK CHANGES (Y/N)?

will appear. Press N. If you press Y instead, you will have to OK each of the changes by typing Y. (This will give you the option to refuse a particular change.)

LIST the program. You will see that all occurrences of X1 have been replaced by XRAY. This form of the CHANGE command will change every occurrence of X1 anywhere it appears in the program. If you only wanted to change X1 on one line, say line 80, the command would look like this:

.C/X1/XRAY/80

The CHANGE command can be used to change any string in the program as long as it is less than 31 characters long.

Renumbering Our Program

GALE has a powerful renumber command. Let's say that we want to change the line number increment of our program from 10 to 100. To do this type the following:

.RN100,I100

and press the RETURN key.

LIST the program, and you will see that the first line is now numbered 100, and the last line is now numbered 1200. RENUMBER can also be used to renumber part of the program without disturbing the other parts of the program. Say, for example, that we want to renumber lines 300-800 with an increment of 50 (instead of 100). To do so type the following:

.R300,I50,F300,L800

and press the RETURN key.

LIST the program. You will see that lines 300-800 are now lines 300-550.

Searching for a Particular String

Let's say that you want to find the program lines that have the variable XRAY in them. GALE provides a command that will do just this. Type:

.S/XRAY

and press the RETURN key. Lines 550 and 1200 will appear on the screen, since these are the two lines of the program that contain XRAY.

If you are only interested in finding the occurrences of XRAY in the lines following line 600, for example, type the following:

.S/XRAY/600,1200

and press the RETURN key. Line 1200 will appear on the screen, since it is the only line after 600 that contains XRAY.

Listing All of Your Variables

GALE will list all of the variables that are in your program, along with the line numbers where they are found. To do this, type the following:

.V

and press the RETURN key. GALE will list all of your variables on the screen with their associated line numbers.

A Brief Review

In this tutorial we have:

Edited a line in the Immediate mode.
Edited a line that we have previously entered.

Used the CHANGE command to change a variable name.
Renumbered our program with the RENUMBER command.
Searched our program for a particular string.
Listed all of our program's variables.

We hope that this tutorial has helped you to become comfortable with the operation of GALE. GALE is very powerful, and we are sure that after having used GALE to edit your programs, you will find your programming time much more enjoyable.

MicroSPARC Inc.

Apple II/IIe