

VOLUME 1 NUMBER 2

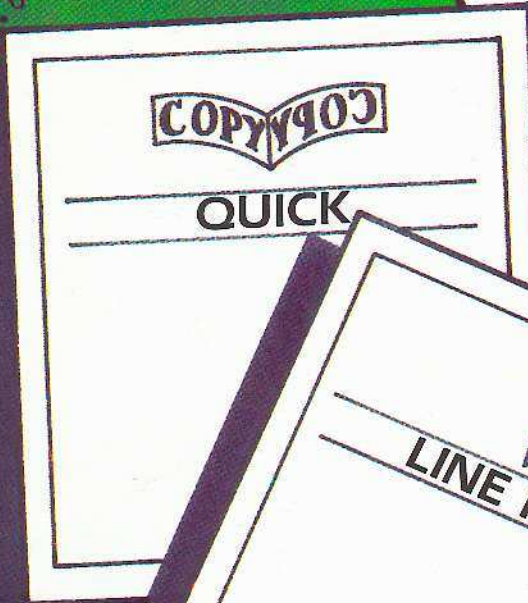
1983

\$3.50

Apple

FOR APPLE USERS

HIGH
RESOLUTION



GOTO LABEL

DYNAMIC
MENU

UTILITIES

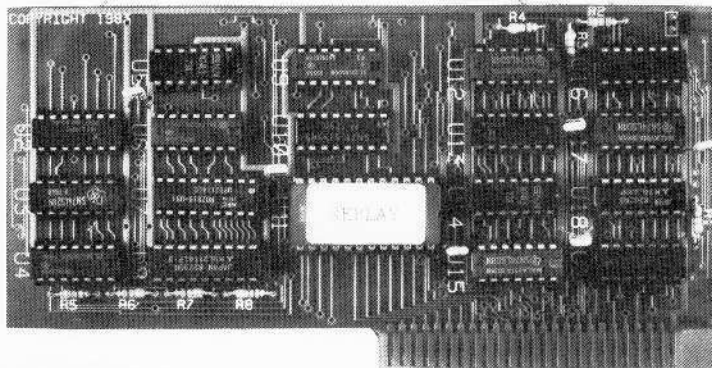


REPLAY II

Apple Program Copy And Development System

Now for
Ile Also

- Disk Formatting Irrelevant
- Does not interfere with other cards
- Card is transparent until copy is desired
- Copies all 64K with *no* compression
- Menu driven
- Copy and restart in under 15 seconds
- Copies memory resident programs
- Analyze programs
- Development tool
- Transparent step & trace
- Backwards & forward disassembly scroll



Why ours is better!

1. Accurate copy of memory
2. Language card not needed to copy or run - when using Replay Card
3. Fast copy & restart
- 15 seconds -
4. Analysis programs and 60 pages of documentation
5. Copy system in eeprom
- No need to boot other disks at copy time -

To order or request info:
Write or call

Micro-Analyst, Inc.
P.O. Box 15003
Austin, TX 78761
(512) 926-4527

Dealer Inquiries Welcome

Cost:
\$130.00 (includes postage)
Outside U.S./Canada add \$10 shipping
Texas residents add \$7.50 sales tax
VISA/MASTER CARD accepted!!

REPLAY II is intended to be used as an analysis tool, for program development, and for making archival backup copies.

*Apple is a registered trademark for Apple Computers Inc.

REPLAY II is an interface card that is slot independent. Users can stop a program, examine and change memory, or copy the program, and restart. Control of the APPLE is obtained by pressing the remote switch which comes on an 18 inch cord outside the APPLE. REPLAY II does not copy the original disk, rather it copies the program executing in memory. If a copy is desired a blank disk is inserted in drive 1 and the options on the menu are contained in the eeprom on the REPLAY II card, no other disk needs to be booted for copying, unlike other copy cards. The very act of booting another disk alters memory which is detectable by some protected software.

REPLAY II does not change ANY memory. Extra memory is buffered to allow copying and analysis without altering the original memory contents. Other copy cards always change specific points in the original memory. REPLAY II faithfully reproduces the lower 48K of memory in a fast load format. The upper 16K can also be copied for a 64K copy. Standard DOS 3.3 files are created automatically for storage on floppy or hard disks. A RAM card is needed for this.

REPLAY II is fully documented in a 60 page manual. Utility programs supplied with the REPLAY II card include Program Analysis, Comparisons, Packing and Compression. A language card is not needed to run packed program copies.

Because most programs are written in Assembly language, the user should be familiar with Assembly in order to fully utilize the advanced Analysis and Packing programs. Users can now freeze a binary program and perform a transparent step or trace while continuous disassembly is shown. View text or hires during trace.

REPLAY II can automatically move protected APPLESOFT programs to a standard DOS 3.3 disk for listing or modification.

Now game players can save a game at any level and QUICKLY restart with the REPLAY II card. Users can freeze games, change variables to obtain unlimited ships or power, etc., then restart the program. Saving high scores is easy!

Minimum requirements are an APPLE II and a single disk drive.

CORETM UTILITIES

Dynamic Menu 4
program..... 5

High Resolution 12
program..... 13
machine code..... 16
checksums..... 16
source code..... 19

Scroll Demonstration
program..... 17
machine code..... 18
checksums..... 17
source code..... 22

GOTO Label 24
program..... 25
checksums—hexdump..... 25

Replace
checksums—hexdump..... 27
program..... 30

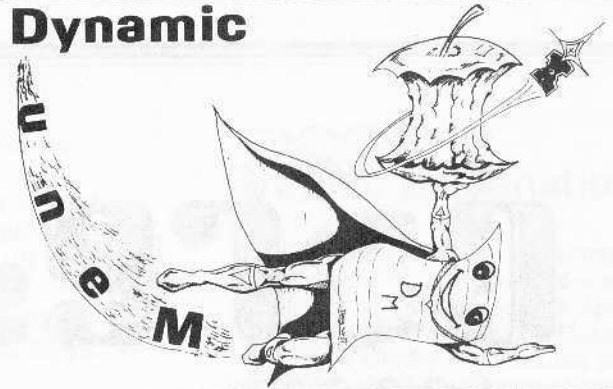
Line Find 34
program..... 35
checksums..... 38

Quick Copy 40
program..... 41
checksums—hexdump..... 29
checksums..... 42

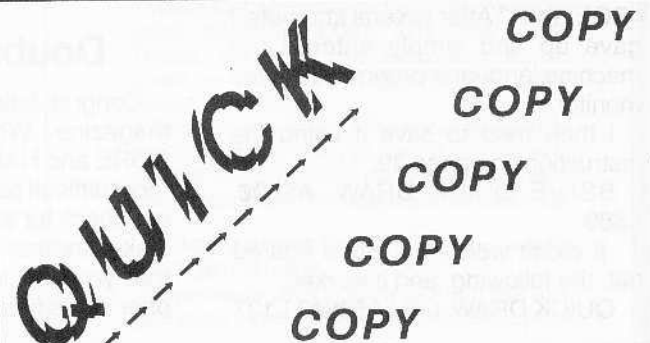
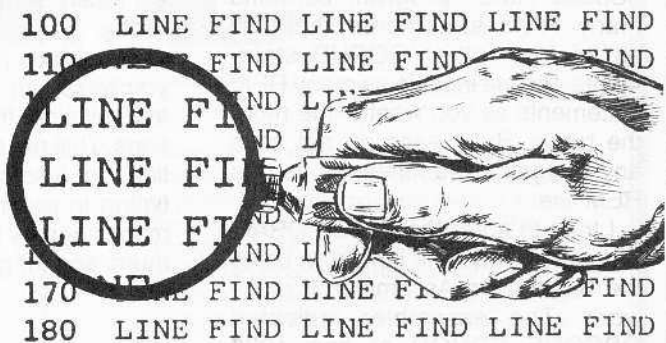
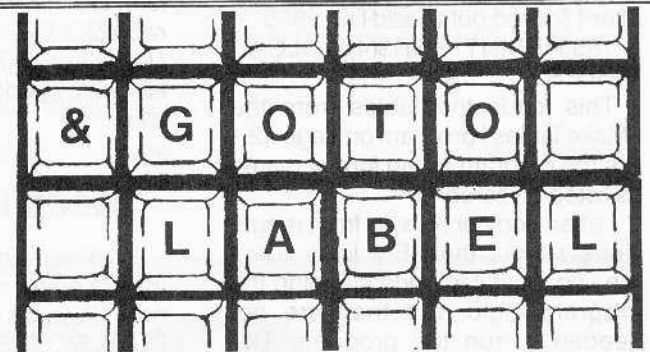
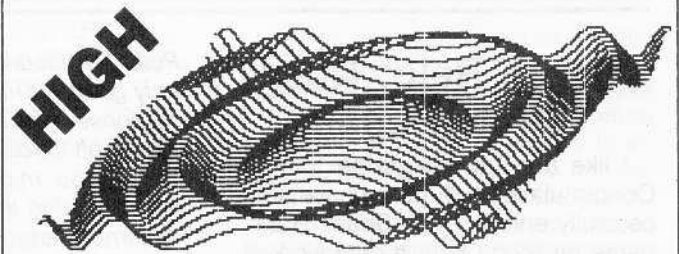
Copy
program..... 44

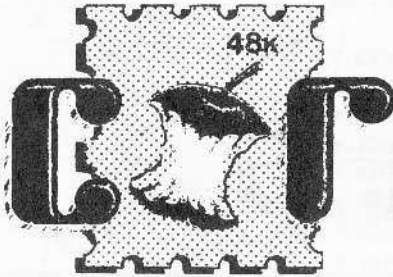
COREspondence..... 2
COREctions..... 47
Advertising Index..... 48

Dynamic



RESOLUTION





Espondence

Space Raid, Quick Draw Repair

I like the new magazine format. Congratulations. Bugs—I have successfully entered the "Space Raid" game by Rich Orde. It only worked after I figured out to add line 1555:

```
1555 PRINT CHR$(4) "BLOAD TABLES"
```

This loads the tables from the "Make Tables" program on page 42. I like the program listing format (same as listed on screen.)

I often copy programs from magazines without the REM lines. They are very useful in understanding the program logic, but they are not needed to run the program. The "Space Raid" program contains many REM statements that are called from other GOSUB statements. Please include as many REM statements as you want—the more the better. But please do not have any program branches go to the REM line.

I tried to enter the QUICK DRAW program in machine language using the S.C. Macro Assembler. It didn't work. The assembler indicated OPCODE ERROR at line 1930 "BGE SKIP." After several attempts, I gave up and simply entered the machine language program from the monitor.

I then tried to save it using the instructions on page 39:

```
BSAVE QUICK DRAW, A$800, L$89
```

It didn't work, so I then figured out the following, and it worked:

```
QUICK DRAW. OBJ,A$0803,L137
```

Peter V. Young
Ardmore, PA

Peter—We found that you were probably getting a FILE TYPE MISMATCH response because there is already an Applesoft file on the disk called Quick Draw. You might find success by instead using the file name QD. Any assembly language listings that contain BGE commands can be corrected by substituting BCS. Replacing BLT with BCC will also eliminate opcode errors when assembling.

Keep It Together

Enclosed find my check for \$20. Please renew my subscription for the next 12 issues of **HARDCORE COMPUTIST/CORE**.

I received my first issue of **CORE** today. I find it very informative, although I do have one gripe. I urge you to refrain from splitting articles and placing them at various locations. This not only leads to aggravation, but also causes mistakes while typing in programs. Most computer magazines try to keep this to a minimum, and for good reason.

George Pleau
St. Louis, MO

Double Header

Congratulations on a great new magazine. What a combination—**CORE** and **HARDCORE**—it will be a most difficult pair to beat! Enclosed is our check for \$15 to cover the **CORE** disk—another real bargain. Hope that you can keep this economical offer in the future.

Harry M. Randel
Scotch Plains, NJ

What Is It?

What in Heaven's name does the latest issue of **CORE** have to do with the magazine I subscribed to? I don't have the slightest interest in computer graphics, but am interested in encoding and decoding the materials on Apple II and III disks. Please hold the graphics stuff.

Professor E.J. Blawie
University of Santa Clara
School of Law

*Dear Mr. Blawie—Publishing a magazine is much like designing an academic curriculum—every issue will not titillate every reader just as every lecture will not thrill every student. But if you bear with us, you will find that **HARDCORE** will continue to delve into the same problems as always—Apple II and III programs. Every third month you will receive **CORE**, which will explore in-depth topics, such as graphics, or in the next two editions, utilities and games. So each year you will still receive eight issues of **HARDCORE**, packed with valuable coding-encoding information. You might want to look at the **CORE** issues as a bonus to your **HARDCORE** subscription.*

Bob's Graphic Needlework

Although I'm less interested in the graphic aspect of Apple programming, I enjoyed vol.1/no.1 very much. I can't wait for no.2, considering your plans, if it's even half as good as this!

I was intrigued by the use of graphics commands to format the text screen. The idea had never occurred to me before, and it does seem to be rather more spiffy.

After playing around with the idea for a bit, I came up with a little variation. The enclosed listing explains how to use the lo-res color characteristic charts on page 17 to alter lines 30 and 40 to vary the box. (I chose an inverse "x" because the appearance is more like old needlework.)

Bob Curtin
Rochester, NY

```

5  REM BOX THREE
6:
7  REM INSPIRED BY TWO SIMI-
  LAR UTILITIES IN CORE 1-1
8:
9:
10 TEXT : HOME : CLEAR
15 WW = 75
20 GR

```

```

30 COLOR = 8: GOTO 50
40 COLOR = 1
50 HLIN 0,39 AT X: X = X + 1
60 IF X = 40 THEN 90
70 IF X / 2 = INT (X / 2) THEN 30
80 GOTO 40
90 POKE 32,1: POKE 33,38:
  POKE 34,1: POKE 35,19:
  HOME
100 POKE - 16303,0
110 MS$ = "YOUR MESSAGE
  HERE"
120 FOR V = 18 TO 2 STEP - 1:
  VTAB V: HTAB (19- LEN (MS$) /
  2): PRINT MS$: VTAB V + 1:
  CALL - 868: FOR W = 1 TO
  WW: NEXT W: NEXT V: HOME
  : GOTO 120

```

Savage Comments

You need more hard in your Core and an update of Castle Wolfenstein.

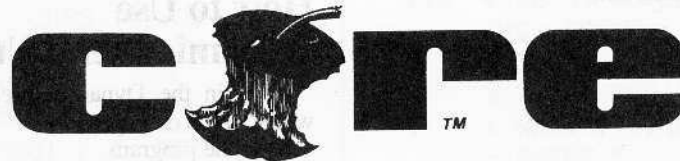
Thomas C. Savage
Sacramento, CA

Graphic Explanation

Your readers might be interested in a relatively new user's group devoted specifically to graphics for the Apple family of computers. The High Resolution Picture Library is concerned with compilation and dissemination of public domain graphics software. We collect hi-res pictures, shape tables, and fonts.

Anyone who would like to join the HRPL and receive the HRPL software currently available can send a diskette in a returnable mailer to HRPL. Return postage in the form of stamps is required. Any software suitable for contribution to the library (programs, shapes, fonts, etc.) may be included on the diskette. Send all correspondence to: HRPL, c/o Paul Pritchard; 2353 S. 8th St.; Omaha, NE 68109. Membership is free.

Paul Pritchard
Omaha, NE



Publisher/Editor:
Charles R. Haight

Managing Editor:
Dennis Bloomquist

Associate Editor:
Julie Joringdal

Technical Editing:
Gary Peterson

Assistants:
Dave Thompson
Ray Darrah

Advertising:
Tanyalee Erwin
CORE Advertising
3710 100th St. SW
Tacoma, WA 98499

Circulation:
Valerie Robinson

Illustrators:
Kris Miskimens

Doug West
Sara Willeston
Steve West

Cover Art:
Stanley Shaw

Typeset via Modem:
Graphic Services
Tacoma, WA

Printing:
Grange Printing, Inc.
Seattle, WA

Publishing:
SoftKey Publishing
P.O. Box 44549
Tacoma, WA 98444
USA

Entire contents copyright 1983 by SoftKey Publishing. All rights reserved. Copying done for other than personal or internal reference (without express written permission from the publisher) is prohibited.

The editorial staff assumes no liability or responsibility for the products advertised in the magazine. Any opinions expressed by the authors are not necessarily those of CORE magazine or SoftKey Publishing.

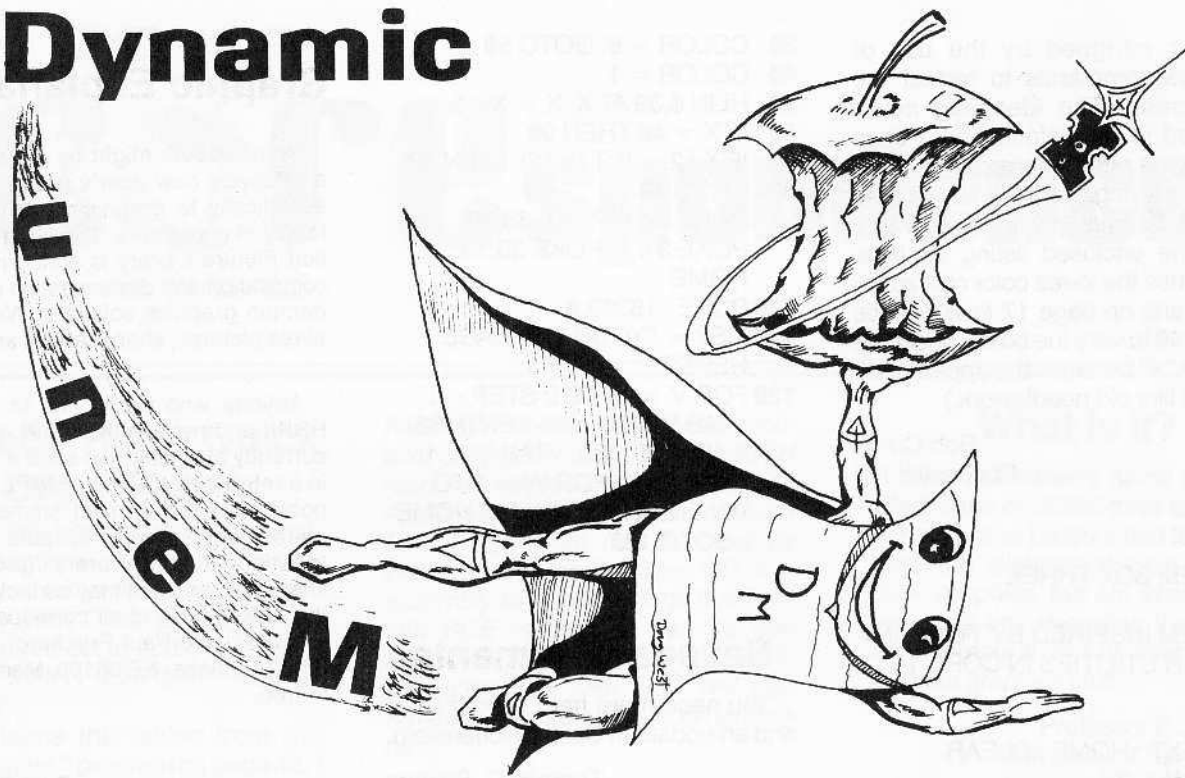
Address all advertising and editorial copy to the proper departments, CORE, 3710 100th St. SW, Tacoma, WA 98499. All subscription inquiries should be directed to CORE, P.O. Box 44549, Tacoma, WA 98444.

Return postage must accompany all manuscripts, drawings, photos, disks, or tapes if they are to be returned. No responsibility can be assumed for unsolicited manuscripts. We suggest you send only copies.

DOMESTIC DEALER RATES sent upon request, or call (206) 581-6038.

Apple usually refers to the Apple II or II Plus computer and is a trademark of Apple Computers, Inc.

Dynamic



By Brent Millirans

Requirements:

Blank, initialized disk
48K Applesoft in ROM
At least one disk drive

The Dynamic Menu Utility uses a push-button approach to writing a menu routine. It creates a text file from .5 to perhaps 4K of Applesoft code which will place a RUNable menu in memory with the touch of only a few keys.

Using the Dynamic Menu Utility, it is quite simple to generate a menu of your choice in about five minutes.

Once the menu has been saved to disk and EXECed back into memory, renumbering, further editing and interfacing to other program routines can take place. The time saved is substantial, especially if a large number of selections is necessary.

The Dynamic Menu Utility sets up a dummy menu from which to work. This will be referred to as the "information block." The only time an information block is not available is when the edit mode is entered and memory is empty.

There are four areas within the program (each will be explained in detail later in the article):

- 1) The Main Menu
- 2) The Director Screen
- 3) The Edit Screen
- 4) The Fix Screen

A short tutorial will provide the user with a functional method of exploring each area and of handling the program as a whole for the first time. Practice, as always, makes perfect.

How to Use Dynamic Menu Utility

Type in the Dynamic Menu Utility Applesoft listing which starts on page 5.

Save the program.

Running the Dynamic Menu Utility brings you to the main menu. Although various functions are available at this point, select number 1, "Develop Menu."

You will be asked to select either double (D) or single (S) spacing. Type "D" for now. Next, enter the number of lines that the proposed menu requires. Type "4" and press return. The program will go to the director screen with a dummy menu and title line. This is what is used to construct the real menu.

To enter the edit mode, press "E". The dummy menu, or information block, will be transferred to the edit screen. At the bottom, note the command line (<< Command) and the prompt "Line Symbol." Press "L" for line and "Line=" will appear. Enter "A" for line A of the alphabetic scaler, which is where the title will be placed. The line will disappear awaiting your entry. Type in "Apple II Tests" and press return.

The new title line will appear and the prompt "Title In Inverse or Normal" will be displayed. Press "I" for inverse. Now the information block should have the inverse title "Apple II Tests" plus four lines, each with the word "selection" in normal video.

Use the "Line" command again, placing in each line (C,E,G,H) the selections "RAM Card," "ROM Card,"

"Memory Test" and "Integer Card." When the prompt "Line Symbol" returns, press "S" for symbol. Next press "L" for letter followed by "I" for inverse. The information block should now look like this:

APPLE II TESTS

A RAM CARD

B ROM CARD

C MEMORY TEST

D INTEGER CARD

At this point, exit by pressing "X" and return to the director. The information block will follow.

Type "F" for fix, and you are back in a similar screen area but in the fix mode.

The next step is to position the block for the most pleasing look. Type "P" for position and the prompt will read "Title or Item." Press "T" for title and then "C" for center. The title should now be centered, but the rest of the block should not have moved.

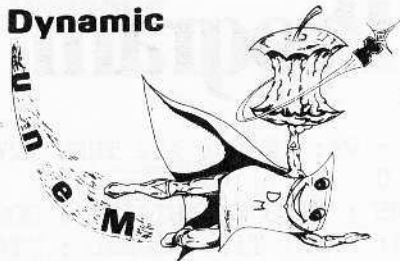
Now press "P" again, followed by "I" for item. Again, entering "C" should center the remainder of the information block. The column command could have been used here as well. Try it if you like and then return the information block to center.

The menu is now ready to be positioned vertically. By pressing "V" for vertical, the next entry will position the title and the menu body using the alphabetic scaler in the left margin. The whole menu is repositioned with the title to prevent overlapping.

Enter "C" and watch the information block reposition from line C on down. Now press "G" and the menu body will position downward separately from the title.

At this point you may wish to play around with different

Dynamic



Program

```

10 ONERR GOTO 1760
20 D$ = CHR$(13) + CHR$(4):G$
   = CHR$(7):ST = 1:TZ$ = "N
   ORMAL": GOTO 120
30 H% = 20 - LEN(OB$) / 2: RETURN
40 H1% = 19 - LEN(OB$) / 2: RETURN
50 T = PEEK(36) + 1: HTAB T: PRINT
   A$;G$;: HTAB T: CALL - 868:
   HTAB T: PRINT A$;: RETURN
60 FOR I = 1 TO 500: NEXT: RETURN

70 FOR I = 1 TO LI: POKE - 1633
   6, PEEK(- 16336): NEXT: RETURN
80 IF I < 10 AND NN = 1 THEN PRINT
   S$;: RETURN
90 RETURN
100 IF TZ$ = "INVERSE" THEN INVERSE
   : RETURN
110 FOR I = 1 TO 100: NEXT: RETURN
120 DIM IT$(16),V$(24),SY$(16),L
   M$(22): IF TG THEN RETURN
130 REM *****
140 REM ** **
150 REM ** FIRST PAGE **
160 REM ** **
170 REM *****

```

```

180 TEXT : HOME :V% = 3:S$ = " "
   :LI = 7:TT$ = ": DYNAMIC MEN
   U UTILITY :": FOR I = 1 TO 5
   : READ SE$(I): NEXT: RESTORE
190 VTAB 3: INVERSE : HTAB 9: PRINT
   TT$: PRINT : VTAB 7: FOR I =
   1 TO 4: HTAB 14: PRINT I;: NORMAL
   : PRINT SE$(I): INVERSE : PRINT
   : NEXT : VTAB 21: HTAB 9: INVERSE
   : PRINT ": PROGRAM AIDE
   :": NORMAL
200 GOSUB 70: GOSUB 110: GOSUB 7
   0: GOSUB 110: GOSUB 70: PRINT
   : FLASH : HTAB 14: VTAB 17: PRINT
   "E";: NORMAL : PRINT SE$(5);
   : HTAB 14: GET AN$: INVERSE
   : PRINT AN$: NORMAL : GOSUB
   70: GOSUB 110: GOSUB 70:AN =
   VAL(AN$): IF AN < 1 OR AN >
   4 THEN 200
210 ON AN GOTO 280,590,700,220
220 VTAB 21:SG$ = ": PROGRAM IS
   COMPLETED :": HTAB 9: INVERSE
   : PRINT SG$: NORMAL : VTAB 2
   3: END
230 REM *****
240 REM ** PARAMETERS **
250 REM ** FOR DEVELOPER **
260 REM ** **
270 REM *****
280 VTAB 21: HTAB 14: INVERSE : PRINT

```

arrangements. But remember, going back to edit will always refile the information block to the edit position.

The next step is to save the finished menu to disk. This is accomplished by returning to the director screen (press "X") and pressing "S" for save. Add your file name (say "test-menu") and press return. If you mistakenly select save or load, entering "X" for a file name will allow an escape.

The drive will start as two files are saved to disk. The screen will display the files being manufactured and saved. One is a data file which, when read in under "load," will reinstall a menu for further editing. The other is an EXECutable text file which will install the menu Applesoft code in memory.

NOTE: The NEW command should be entered before EXECing any file because the EXEC command does not clear memory as RUN or BRUN does.

The rest is up to you. Reading the explanation of each area of the program and the corresponding commands should answer most questions.

Main Menu

Main Menu selections:

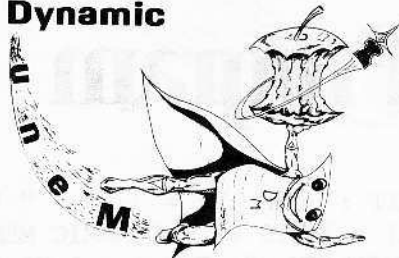
- 1) Develop Menu—Allows building a menu from scratch with full-time editing available.
- 2) Revise Files—Brings file into memory for further editing or revision.
- 3) Edit Program—Allows reentry to editing without loss of data in case of irrecoverable error.
- 4) Exit Program—Returns user to Applesoft.

The setup entry for the "Develop Menu" selection will require two initial inputs before allowing editing. You must specify: 1) single or double spacing, and 2) the number of items in the proposed menu.

As a programming hint, two or three shorter menus are much more user-friendly than one long one.

In "Revise Files" mode you will be asked for a file name, but only the user portion (i.e., "test"). Since the program assigns a primary file name (menu.mod.), the user provides only the latter portion (menu.mod.test). The display will say,

Dynamic



```

" DEVELOP MENU ": NORMAL : GOSUB
60: CLEAR :WA$ = "": TEXT : HOME
:MX = 16:S$ = " ":TG = 1: GOSUB
20:TI$ = " -DEVELOPEMENT- ":
  FOR I = 0 TO 16:IT$(I) = "S
ELECTION": NEXT
290 VTAB 2: HTAB 12: INVERSE : PRINT
TI$: NORMAL :LI = 2
300 VTAB 5: PRINT "SINGLE SPACES
OR DOUBLE ... D";: HTAB 29:
  GET PC$: INVERSE : PRINT PC
$: NORMAL : GOSUB 70: IF PC$
= "D" THEN SP = 1:MX = 8
310 IF PC$ < > "D" AND PC$ < >
"S" THEN 300
320 VTAB 7: PRINT "HOW MANY ENTR
IES .....": PRINT "(MAX
IMUM=";MX;")": VTAB 7: HTAB
29: INPUT "":N: GOSUB 70: IF
N < 1 OR N > MX THEN VTAB 8
:A$ = " -INVALID- ": GOSUB 5
0: GOSUB 60: PRINT : GOTO 32
0
330 V1% = 2:V% = 2:H% = 3:H1% = 2
:TI$ = "TITLE LINE": IF SP THEN
ST = 2:N = N * 2
340 GOTO 460

```

Program

```

350 V2% = V%: IF V1% > 2 THEN V2%
= 0
360 HOME : VTAB V%: HTAB H%: GOSUB
100: PRINT TI$: NORMAL : VTAB
V1% + V2%: FOR I = 1 TO N STEP
ST: IF FH THEN FLASH : HTAB
H1%: PRINT SY$(I);: NORMAL :
  GOSUB 80: PRINT S$;IT$(I): GOTO
390
370 IF IV THEN INVERSE : HTAB H
1%: PRINT SY$(I);: NORMAL : GOSUB
80: PRINT S$;IT$(I): GOTO 39
0
380 HTAB H1%: GOSUB 80: PRINT SY
$(I);S$;IT$(I)
390 IF SP THEN PRINT
400 NEXT : RETURN
410 REM *****
420 REM **
430 REM ** THE DIRECTOR **
440 REM **
450 REM *****
460 HOME : POKE 32,2: POKE 33,38
: VTAB 22: FOR I = 2 TO 38: PRINT
"-";: NEXT : HTAB 1: INVERSE
: PRINT "<DIRECTOR>";: NORMAL
: HTAB 24: PRINT "-('X' TO E
XIT)"
470 VTAB 23: HTAB 1: PRINT "[E]D
IT";: HTAB 8: PRINT "[F]IX";
: HTAB 14: PRINT "[S]AVE";: HTAB

```


"Add file name: menu.mod." The user will simply type "test" or whatever file name is chosen.

With "Edit" selection, if memory has a file in it, you will be returned to the edit mode. If memory is empty you will be returned to the director. A file may be loaded from disk just as in the revise mode.

Director Screen

While working from the director screen (as noted at the bottom left corner) there are four basic commands available. Any selection from the main menu will be processed through the director screen, except the edit mode selection if a file is in memory.

Edit—Places the information block in the edit mode using a formatting grid. While in this mode the text of each line of the information block may be altered.

Fix—Places the information block in the fix mode using a formatting grid. While in this mode the vertical and horizontal position of the information block may be changed. It is important that all editing work be completed before "fixing" the

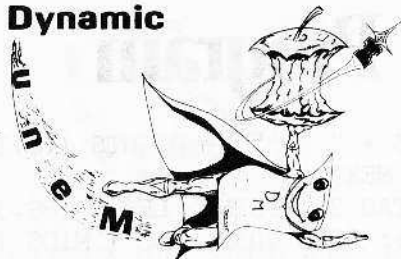
position of the information block. The reason for this is that the edit mode automatically fixes the information block so the program can accurately locate the line currently being altered.

Save—Asks for the name under which you wish to save the finished menu. The name chosen should be short, as it is appended to an internally selected file name. You will be asked to "add file name: menu.mod." Your chosen name, say "testmenu," would result in a file name of "menu.mod.testmenu". The reason for this is that the Dynamic Menu Utility writes two files. One is executable and one is for data only. Two files would result on disk from the above example, "menu.mod.menuitest" and "menu.val.menuitest."

Load—Asks for the name to be appended to the internal file name "menu.val." This is the data file and memory will be loaded with proper data resulting in an editable information block. You are then returned to the director screen.

X—Exit to main menu. In the edit and fix modes, additional commands are available. The edit mode is described first as this is the usual starting point.

Dynamic



```

21: PRINT "[L]OAD";
480 HTAB 28: PRINT "<< SELECT";:
      POKE 35,21: GOSUB 570: GOSUB
550: VTAB 23: HTAB 37: GET S
E$: GOSUB 70
490 IF SE$ = "E" THEN FX = 0:V% =
2:V1% = 4:H% = 2:H1% = 2: GOTO
770
500 IF SE$ = "F" THEN 1050
510 IF SE$ = "S" THEN 1410
520 IF SE$ = "L" THEN 650
530 IF SE$ = "X" THEN 180
540 GOTO 480
550 IF WA$ < > "" THEN VTAB 10
:OB$ = WA$: GOSUB 30: HTAB H
%: PRINT WA$
560 RETURN
570 IF NOT EX THEN GOSUB 350
580 RETURN
590 LI = 2: VTAB 21: HTAB 14: INVERSE
: PRINT " REVISE FILES ": NORMAL
: GOSUB 60:EX = 1: TEXT : HOME
:WA$ = "LOAD FILE TO BE MODI
FIED":OB$ = WA$: GOSUB 30: GOTO
460

```

Program

```

600 REM *****
610 REM ** **
620 REM ** LOAD ROUTINE **
630 REM ** **
640 REM *****
650 HOME : VTAB 3: HTAB 14: INVERSE
: PRINT ": FILE LOAD ": NORMAL
: POKE 34,4: VTAB 7:XF = 1:F
L$ = "MENU.VAL.": HTAB 15: PRINT
FL$: HTAB 1: PRINT "ADD FIL
ENAME: ";: HTAB 24: INPUT ""
;NA$: GOSUB 70:FL$ = FL$ + N
A$
660 IF NA$ = "" THEN HOME :A$ =
"NO ENTRY - RESELECT": VTAB
10: HTAB H%: GOSUB 50: VTAB
23: GOSUB 60: GOTO 480
670 IF NA$ = "X" THEN 180
680 PRINT : PRINT D$;"MONICO": PRINT
D$;"OPEN";FL$: PRINT D$;"REA
D";FL$: INPUT FH: INPUT IV: INPUT
TI$: INPUT H%: INPUT H1%: INPUT
V%: INPUT N: INPUT SP: INPUT
TZ$:ST = 1: IF SP THEN ST =
2
690 FOR I = 1 TO N STEP ST: INPUT
SY$(I): INPUT IT$(I): NEXT :
INPUT TY$: INPUT V1%: INPUT
V2%: PRINT D$;"CLOSE": PRINT
D$;"NOMONICO": POKE 34,0: HOME
:WA$ = "":EX = 0: GOTO 460: END

```

Edit Screen

When entering the edit mode, the Apple will beep and "<<Command" will appear in the right bottom corner. This simply draws attention to the command queries. You will be prompted by the words "Line Symbol." Pressing the key corresponding to the first letter of either command will provide the desired operation.

Line—Asks for which line to edit using the left margin scale A through S. Line "T" is reserved for the prompt. Enter the wording desired, but keep it short (under 30 characters). Use the left arrow key as usual to backspace and correct mistakes. You may, of course, reselect the line. When selecting Line A (Line = A), you will always be altering the title. After typing in the title of the menu being built (or leaving it blank) you will be asked whether you wish the title in normal or inverse video. Select by keypress as desired. You will be returned to the command line and prompted with "Line Symbol."

Symbol—Refers to the "keypress symbol" that you wish to use in your menu. You will be prompted to select "letter or number" symbols. The letters A through Z (if

necessary) will be assigned to the left of each selection entry if letters are selected. Numbers may otherwise be assigned. You will afterward be prompted to choose whether the assigned symbols will be displayed normally, or in flashing or inverse video. There is no interference between the Line or Symbol commands. Either may be selected at any time.

X—At the point of completion in the edit mode the X command will return you to the director screen.

NOTE: The Dynamic Menu Utility will automatically assign a "prompt line" for your menu in a complementary style to the finished information block. This is only visible when running the finished product.

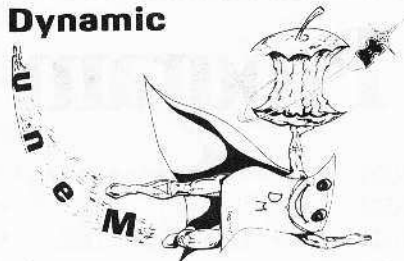
Fix Screen

The fix mode would be the next step in building the menu. Depending on the size of the menu in progress, its position on the screen will need to be changed.

The command line will feature the prompt "Position Column Vertical." Again, pressing a key corresponding to the first letter of each word will produce the desired operation.

Position—Asks for the desired horizontal position of the

Dynamic



```

700 LI = 2: VTAB 21: HTAB 14: INVERSE
   : PRINT " EDIT PROGRAM ": NORMAL
   : GOSUB 60:WA$ = "":EX = 0: TEXT
   : HOME : IF IT$(1) = "" THEN
WA$ = "NO FILE IN MEMORY": GOTO
460
710 SE$ = "E": GOTO 490
720 REM *****
730 REM **
740 REM ** EDIT MENU **
750 REM **
760 REM *****
770 TEXT : HOME : HTAB 22: FLASH
   : PRINT "EDIT";: NORMAL : PRINT
   " ('X' TO EXIT)": POKE 34,1
780 FOR I = 1 TO 21:LM$(I) = CHR$(
   (I + 64): PRINT LM$(I) + "-"
   : NEXT : PRINT ;: POKE 32,2:
   POKE 33,38: VTAB 22: PRINT
   "--^";: FOR IR = 1 TO 7: FOR
   I = 1 TO 4: PRINT "-";: NEXT
   I: PRINT "^";: NEXT IR
790 VTAB 23: HTAB 2: PRINT "C0";
   : FOR I = 10 TO 40 STEP 5:I$
   = STR$(I): PRINT SPC(3)
   + "C" + LEFT$(I$,1);: NEXT
   :B$ = " 5": FOR I = 10 TO 4
0 STEP 5:I$ = STR$(I):B$ =

```

Program

```

B$ + " " + RIGHT$(I$,1)
: NEXT
800 VTAB 24: PRINT LEFT$(B$,37
   );: POKE 2039, ASC ( MID$(B
   $,38,1)) + 128: VTAB 1: POKE
   35,21
810 GOSUB 350: VTAB 21: HTAB 29:
   PRINT "<<COMMAND";: VTAB 21
   : HTAB 1: POKE 33,38: GOSUB
   60: IF FX THEN 1060
820 HTAB 1: PRINT SPC(27);: HTAB
   1: PRINT "[L]INE [S]YMBOL
   <<COMMAND";: HTAB
   28: GET CM$: IF CM$ < > "L"
   AND CM$ < > "S" AND CM$ <
   > "X" THEN 820
830 GOSUB 1220: IF CM$ < > "L" THEN
   910
840 V% = ASC (LN$) - 63: IF V% >
   24 THEN V% = 2: TEXT : HOME
   : GOTO 460
850 IF V% < 1 OR V% > 22 THEN 82
   0
860 VTAB V%: HTAB 3: IF LN$ = "A
   " THEN HTAB 1:TI$ = "": PRINT
   SPC(30);: HTAB 3: INPUT ""
   ;TI$: GOSUB 70: GOTO 880
870 LN = ASC (LN$) - 66: HTAB 1:
   PRINT SPC(30);: HTAB 4: INPUT
   "";IT$(LN): GOSUB 70
880 IF LN$ = "A" THEN VTAB V%: HTAB
   H$: PRINT SPC(25);: HTAB H

```


title or the item of the information block. Enter "T" for title or "I" for item. One of the prompts, flush left, flush right or centered, will then be offered. A single keypress will automatically fix the horizontal position of the selected part of the information block.

Column—Should the position command not be adequate for horizontal positioning, the column command uses the numeric scale along the bottom of the screen. The entry range is 1 through 40, and will HTAB the information block accordingly. This command also moves the title and the item block separately. Only the first item of the information block is aligned. Other items following are keyed to the first item.

Vertical—Asks for the vertical position of the information block using the left margin alphabetic scale. This is done in two parts. When first selected, you will be prompted for the vertical position of the title line (even though blank). Enter the letter corresponding to the vertical position you wish to use for the title only. Although the entire information block will move, a second prompt will ask for the vertical position of the initial selection entry (item). The body of the menu will then

be moved separately, leaving the title as it was originally positioned.

X—Exits the fix mode and returns control to the director screen.

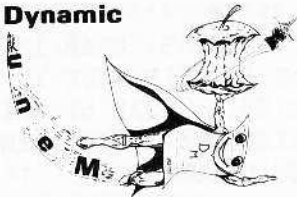
At this point the menu should be edited and the position fixed. However, the edit mode could be reentered if desired. Note again that the information block will have to be "refixed" if additional editing is done. The SAVE command will allow you to save the generated menu to disk (see SAVE command).

Improving the Program

By now you may have several ideas for program improvements. One suggestion is to make it possible for the Menu program to be incremented by any number of lines, starting at any line number. This would allow the menu to be placed conveniently in other programs.



Dynamic



```

%: GOSUB 100: PRINT TI$: NORMAL
: VTAB 21: PRINT "TITLE IN "
;: PRINT "[I]NVERSE OR [N]OR
MAL ?";: CALL - 868: HTAB 3
2: GET TZ$: GOSUB 70
890 IF TZ$ = "I" THEN TZ$ = "INV
ERSE": GOTO 900
900 V% = 2: GOSUB 350: VTAB 21: GOTO
820
910 IF SY$ = "L" THEN GOSUB 127
0:SY$ = ""
920 IF SY$ = "N" THEN GOSUB 130
0:SY$ = ""
930 IF CM$ = "T" THEN 950
940 CM$ = "T": GOSUB 1250
950 IF TY$ = "F" THEN FH = 1:TY$
= ""
960 IF TY$ = "I" THEN IV = 1:FH =
0:TY$ = ""
970 IF TY$ = "N" THEN FH = 0:IV =
0
980 GOTO 810
990 TEXT : GOTO 460
1000 REM *****
1010 REM ** **
1020 REM ** FIX POSITIONS **
1030 REM ** **
1040 REM *****
1050 TEXT : HOME : HTAB 23: FLASH

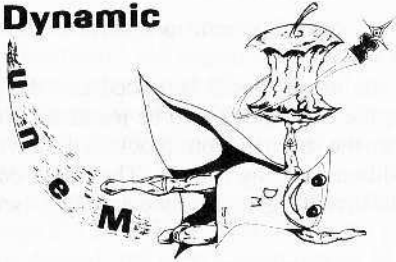
```

```

: PRINT "FIX";: NORMAL : PRINT
" ('X' TO EXIT)": POKE 34,1:
FX = 1: GOTO 780
1060 VTAB 21: HTAB 1: PRINT SPC(
37);: HTAB 1: PRINT "[P]OSIT
ION [C]OLUMN [V]ERTICAL ";: GET
CM$: IF CM$ < > "X" AND CM$
< > "P" AND CM$ < > "C" AND
CM$ < > "V" THEN 1060
1070 GOSUB 1330
1080 IF CM$ = "V" THEN V% = ASC
(VT$) - 63:V1% = V% + 2: GOSUB
350: GOSUB 1380:V1% = ASC (
V1$) - 63: GOSUB 350: GOTO 1
060
1090 IF CM$ = "P" THEN 1130
1100 IF QL$ = "I" THEN H1% = VAL
(CL$) - 2: GOSUB 1200
1110 IF QL$ = "T" THEN H% = VAL
(CL$) - 2: GOSUB 1200
1120 GOSUB 350: GOTO 1060
1130 IF AN$ = "L" AND QL$ = "T" THEN
H% = 1
1140 IF AN$ = "L" AND QL$ = "I" THEN
H1% = 1
1150 IF AN$ = "R" AND QL$ = "T" THEN
H% = 38 - LEN (TI$)
1160 IF AN$ = "R" AND QL$ = "I" THEN
H1% = 36 - LEN (IT$(1))
1170 IF AN$ = "C" AND QL$ = "T" THEN
OB$ = TI$: GOSUB 30
1180 IF AN$ = "C" AND QL$ = "I" THEN
OB$ = IT$(1): GOSUB 40
1190 GOSUB 350: GOTO 1060
1200 IF H% < 1 OR H% > 38 THEN A
$ = "-INVALID-": VTAB 3: HTAB

```

Dynamic

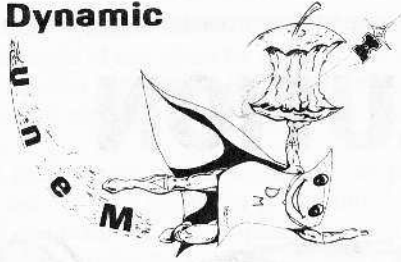


```
27: GOSUB 50: VTAB 21: POP :
  GOTO 1060
1210 RETURN : END
1220 IF CM$ = "X" THEN 990
1230 VTAB 21: IF CM$ = "L" THEN
  HTAB 1: PRINT SPC( 20);: HTAB
  1: PRINT "LINE=";: GET LN$:
  PRINT LN$;: RETURN
1240 VTAB 21: HTAB 1: IF CM$ = "
S" THEN PRINT "SYMBOL= [L]E
TTER [N]UMBER ";: GET SY$: GOSUB
70: PRINT SY$;: RETURN
1250 VTAB 21: HTAB 1: PRINT SPC(
37);: HTAB 1: IF CM$ = "T" THEN
  PRINT "TYPE=";: PRINT "[F]
LASH [I]NVERSE [N]ORMAL ";: GET
TY$: GOSUB 70: PRINT TY$;: RETURN
1260 GOTO 1220
1270 RL = 16:NN = 0:R = 0: IF SP THEN
  RL = 8
1280 FOR I = 1 TO RL:SY$(I + R) =
  CHR$( I + 64): IF SP THEN R
  = R + 1
1290 NEXT : RETURN
1300 NN = 1:RL = 16:R = 0: IF SP THEN
  RL = 8:NN = 0
1310 FOR I = 1 TO RL:SY$(I + R) =
  STR$( I): IF SP THEN R = R +
  1
1320 NEXT : RETURN
1330 IF CM$ = "X" THEN 990
1340 IF CM$ = "P" THEN GOSUB 14
  00
1350 VTAB 21: HTAB 1: IF CM$ = "
P" THEN PRINT "POS.=FLUSH "
;: PRINT "[L]EFT [R]IGHT OR
[C]ENTER";: GET AN$: GOSUB 7
  0: RETURN
1360 IF CM$ = "V" THEN PRINT SPC(
37);: HTAB 1: PRINT "VTAB TI
TLE AT LINE=";: GET VT$: GOSUB
70: RETURN
1370 IF CM$ = "C" THEN GOSUB 14
  00: HTAB 1: PRINT SPC( 26):
  HTAB 1: PRINT "COLUMN NUMBE
R=";: HTAB 16: INPUT "";CL$:
  GOSUB 70: RETURN
```

Program

```
1380 IF CM$ = "V" THEN VTAB 21:
  HTAB 1:V2% = 0: PRINT "VTAB
  FIRST ENTRY AT=";: GET V1$:
  : GOSUB 70: RETURN
1390 GOTO 1330
1400 HTAB 1: VTAB 21: PRINT "[T]
  ITLE OR [I]TEM POSITION ";:
  CALL - 868: GET QL$: GOSUB
  70: RETURN
1410 FI$ = "MENU.MOD.":FL$ = "MEN
  U.VAL."
1420 REM *****
1430 REM ** **
1440 REM ** GET FILE NAME **
1450 REM ** **
1460 REM *****
1470 HOME : VTAB 3: HTAB 13: INVERSE
  : PRINT ": FILE SAVE ": NORMAL
  : POKE 34,4: VTAB 6: PRINT "
  ADD FILENAME: ";FI$;: INPUT
  "";NA$: GOSUB 70
1480 GOSUB 1490: GOSUB 1510:FI$ =
  FI$ + NA$:FL$ = FL$ + NA$: VTAB
  8: PRINT "DISK READY PLEASE
  ...DRIVE 1": VTAB 10: PRINT
  "PRESS RETURN ";: GET RT$: GOSUB
  70: GOTO 1530
1490 IF NA$ = "" THEN POP : VTAB
  8: PRINT "NO FILE NAME ...":
  VTAB 10: PRINT "RE-ENTER OR
  TYPE 'X' TO EXIT ";G$: GOSUB
  60: GOSUB 60: GOTO 1470
1500 RETURN
1510 IF NA$ = "X" THEN POKE 34,
  0: HOME : POP : GOTO 460
1520 RETURN
1530 IF FH THEN TY$ = "FLASH"
1540 IF IV THEN TY$ = "INVERSE"
1550 REM *****
1560 REM ** **
1570 REM ** SAVE ROUTINE **
1580 REM ** **
1590 REM *****
1600 PRINT D$;"MONICO": PRINT D$
  ;"OPEN";FI$: PRINT D$;"DELET
  E";FI$: PRINT D$;"OPEN";FI$:
  PRINT D$;"WRITE";FI$: PRINT
  "1000 TEXT:HOME": PRINT "100
  5 VTAB";V%:";HTAB";H%:";";TZ
  $;":PRINT"; CHR$( 34);TI$; CHR$(
  34);":NORMAL"
```


Dynamic



```
1605 END
1610 IF SP THEN FOR I = 1 TO N STEP
2: GOTO 1630
1620 FOR I = 1 TO N
1630 PRINT "10" + STR$ ((I * 5)
+ 5) + "VTAB"; (V1% - 1) + I
;":HTAB";H1%;":":TY$;":PRINT
"; CHR$ (34);SY$(I); CHR$ (3
4);";:NORMAL:PRINT"; CHR$ (3
4);S$ + IT$(I); CHR$ (34): NEXT
1640 PRINT "1200 VTAB 21:HTAB";H
1%;":":TY$;":PRINT"; CHR$ (3
4);"S"; CHR$ (34);";: NORMAL
: PRINT "; CHR$ (34);" SELE
CT..."; CHR$ (34);";: HTAB "
;H1%;": GET SE$: PRINT SE$":
IF SP THEN FOR I = 1 TO N STEP
2: GOTO 1660
1650 FOR I = 1 TO N
1660 PRINT "12" + STR$ ((I * 5)
+ 5) + " IF SE$="; CHR$ (34
);SY$(I); CHR$ (34);"THEN 14
" + STR$ ((I * 5) + 5): NEXT
: PRINT "1390 GOTO 1200": IF
SP THEN FOR I = 1 TO N STEP
2: GOTO 1680
1670 FOR I = 1 TO N
1680 PRINT "14" + STR$ ((I * 5)
+ 5) + " VTAB 23:PRINT"; CHR$
(34);IT$(I);" ROUTINE"; CHR$
(34);": END ": NEXT : PRINT
"1500 END": PRINT D$;"CLOSE"
: PRINT D$;"OPEN";FL$: PRINT
D$;"DELETE";FL$: PRINT D$;"O
PEN";FL$: PRINT D$;"WRITE";F
L$
1690 PRINT FH: PRINT IV: PRINT T
I$: PRINT H%: PRINT H1%: PRINT
V%: PRINT N: PRINT SP: PRINT
TZ$:ST = 1: IF SP THEN ST =
2
1700 FOR I = 1 TO N STEP ST: PRINT
SY$(I): PRINT IT$(I): NEXT :
PRINT TY$: PRINT V1%: PRINT
V2%: PRINT D$;"CLOSE": PRINT
D$;"NOMONICO": GOTO 180
```

Program

```
1710 REM *****
1730 REM ** ERROR TRAPPER **
1750 REM *****
1760 TEXT : HOME : VTAB 12: HTAB
10: INVERSE : PRINT "-IRRECO
VERABLE ERROR-": HTAB 13: PRINT
"-AUTO RE-START-": NORMAL : PRINT
G$G$G$: GOSUB 60: GOSUB 60:T
M = TM + 1: IF TM = 3 THEN RUN
1770 V% = 2:V1% = 2:V2% = 2: GOTO
460
1780 REM * * * * *
1800 REM * MENU-GENERATOR *
1820 REM * BY *
1840 REM * BRENT A.MILLIRANS *
1860 REM * *
1880 REM * * * * *
1890 DATA " DEVELOP MENU"," RE
VISE FILES"," EDIT PROGRAM",
" EXIT PROGRAM"," ENTER ITEM
#"
```

DISK DRIVES

\$199 *First time under \$200!*

At last you can afford to buy another drive!!

- Siemens drive mechanism (uses lead screw instead of rotating cam, for very quiet, reliable operation)
- Works with your present controller
- Can be used as 35 or 40 track drive
- Improved self-centering of diskettes
- Externally-adjustable speed control
- Handles synchronized tracks
- Manufacturer's 1-year warrantee

These superb ready-to-use drives come with standard size and color casings. *Disk controller cards available: \$65.00*

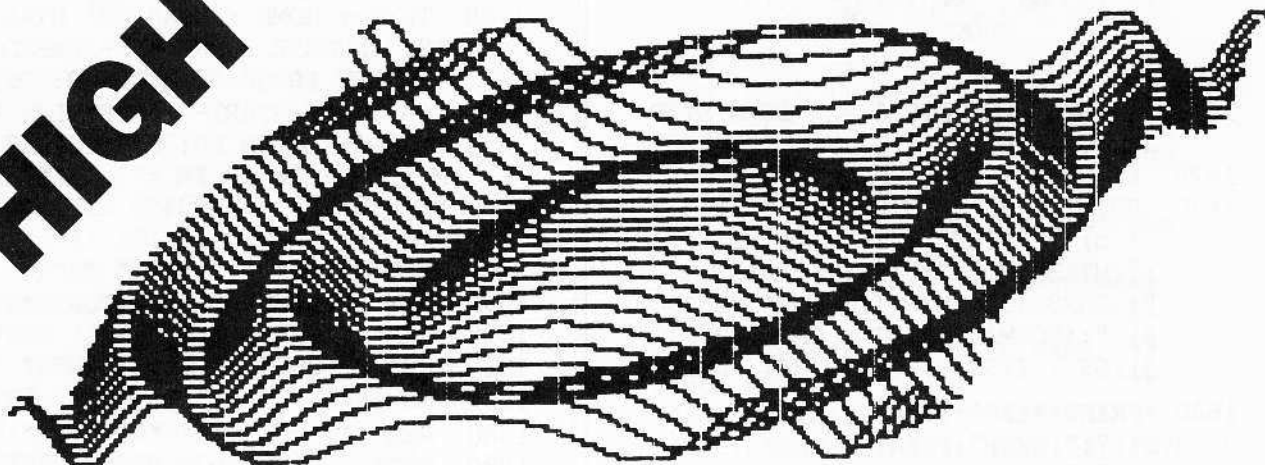
No charge cards or COD orders. Add \$5 for shipping and handling. Send money order or certified check for immediate shipment, \$4 extra for 2nd day delivery
Ask for free 74LS125 chip with order.

VF ASSOCIATES

6327 Western Ave, N.W., Wash., D.C. 20015
(202) 363-1313

RESOLUTION

HIGH



By Michael Patrick Scanlin

Requirements:

Blank, initialized disk
48K, Applesoft in ROM
One disk drive

In one smooth motion the screen phased from white on black into black on white. The inverse routine in Juggler was my motivation to devise an original one.

But just inverting the screen proved unchallenging, so I added a few twists to the program. Besides the normal inverse routine, I created routines for inverting the screen from top to bottom and from right to left, and a programmable delay to be used with any of these functions.

Inverting the screen in various ways was a bigger challenge, but then I decided to add more routines which would transpose and flip the screen. Next, I added a separate program which simulates the game Ultima by scrolling hi-res page one up, down, right or left. The result of all this is a complete set of hi-res utilities.

ROUTINES PROGRAM

The inverse, flip, and transpose routines are part of the hi-res utility program ROUTINES. To use the program:

- 1) Type in the ROUTINES listing on page 16.
- 2) **BSAVE ROUTINES, A \$9000, L\$1FB**
- 3) **BLOAD ROUTINES**
CALL 36864

This sets up a table of base addresses that the routines use at

\$9200 in memory. To access the inverse routines described earlier simply make the appropriate CALL.

Routine	CALL
fast	36997 (\$9085)
top to bottom w/delay	37024 (\$90A0)
right to left w/delay	37058 (\$90C2)

Transpose Routine

The transpose routine included with the inverse package is used to move hi-res page two onto page one while displaying page one. There are three ways to move the pages, which correspond to the inverse sequence.

Routine	CALL
fast	36874 (\$900A)
top to bottom w/delay	36908 (\$902C)
right to left w/delay	36947 (\$9053)

Flip Routine

Two flip routines are included with this package. They can be used to turn hi-res page one upside down or to turn it right to left.

Routine	CALL
top to bottom w/delay	37107 (\$90F3)
right to left w/delay	37172 (\$913A)

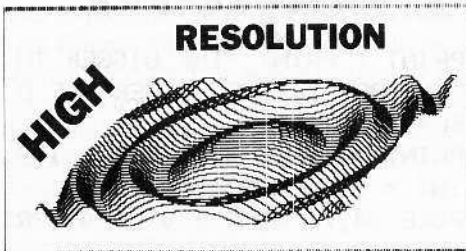
Delay Routine

The delay routine used is the monitor's (\$FCA8). To change the delay for all of the routines, POKE 254,X where X is any number from 1 to 255. The smaller the number, the shorter the delay.

How the Routines Work

The inverse routines use the logical EOR function, which allows the status of any given bit or set of bits in a byte to be changed. All that is necessary is to EOR the bit (or bits) with one. This will change it to the opposite of what the bit was originally.

A byte on the hi-res screen contains seven pixels, each of which is one bit. The eighth bit is the color bit. In order to invert all the pixels on the graphics page, EOR the lower seven bits (preserving the color bit) of every byte with one, from memory locations \$2000 to \$3FFF (the hi-res screen).



```
10 REM
20 REM ** VARIABLES SET TO SUBROUTINES **
30 REM
40 IRL = 37058:ITB = 37024:INV = 36997
50 FTB = 37107:FRL = 37172:SETUP = 36864
60 TNS = 36874:TRL = 36947:TTB = 36908
70 DLAY = 254
80 TEXT : HOME
90 VTAB 1: HTAB 1: PRINT "INVERSE & FLIP DEMONSTRATION": NORMAL : POKE 34,3
100 HOME : HTAB 1: PRINT "LOADING ROUTINES"
110 PRINT CHR$(4)"BLOAD ROUTINES"
120 HIMEM: 36864
130 CALL SETUP
140 PRINT : PRINT "DO YOU HAVE A HI-RES PICTURE TO LOAD?"; CHR$(8);
150 GET A$: IF A$ < > "Y" THEN GOSUB 820: GOTO 190
160 HTAB 1: PRINT "NAME OF PICTURE =>" TAB( 39)
170 HTAB 19: INPUT "":A$: PRINT : PRINT "LOADING ":A$
180 PRINT CHR$(4);"BLOAD";A$;" ,A$4000"
190 CALL TNS
200 GOTO 260
210 POKE - 16304,0: POKE - 16302,0: POKE - 16300,0: POKE - 16297,0: RETURN
```

The transpose routines aren't too difficult to understand. They are memory move commands which take place in a definite order to give special effects. The only one that requires any explanation is the right-to-left routine, which uses the logical AND function. This function can be thought of like multiplication. If you AND zero and one, you get zero (because $0 * 1 = 0$). If you AND one and one, you get one ($1 * 1 = 1$).

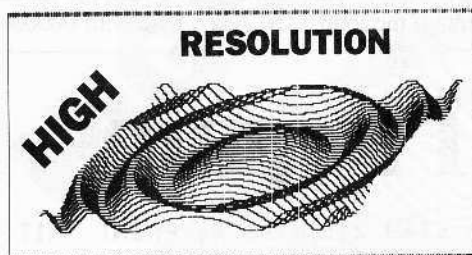
So, in order to get the smooth right to left motion, create a loop that will show one more bit of each byte each time it goes through the loop. The first time through, AND each byte in a column with 00000001. This will cut off all but the first bit. Next time through the loop, AND each byte with 00000011

Program

```
220 VTAB 23: HTAB 8: PRINT "HIT ANY KEY TO CONTINUE": POKE - 16368,0
230 IF PEEK ( - 16384) < 128 THEN 230
240 POKE - 16368,0: RETURN
250 GOSUB 220: GOSUB 210: GOSUB 220: POKE DLAY,WT: CALL VAR: GOSUB 220: POKE - 16303,0: RETURN
260 REM
270 REM ** MAIN PROGRAM **
280 REM
290 HOME : VTAB 5
300 PRINT : PRINT "THESE ROUTINES CAN BE USED FROM WITHIN"
310 PRINT : PRINT "A BASIC PROGRAM TO DO THE FOLLOWING:" : PRINT : PRINT
320 PRINT "INVERSE SCREEN => MAKE EVERY WHITE DOT"
330 PRINT TAB( 19)"A BLACK ONE AND EVERY"
340 PRINT TAB( 19)"BLACK DOT A WHITE ONE": PRINT
350 PRINT "FLIP SCREEN"; TAB( 16 );"=> TURN SCREEN UPSIDE"
360 PRINT TAB( 19)"DOWN OR TURN SCREEN"
370 PRINT TAB( 19)"RIGHT TO LEFT": PRINT
380 PRINT "TRANSPOSE"; TAB( 16); "=> MOVE HIRES PAGE TWO"
390 PRINT TAB( 19);"ON TO PAGE ONE": PRINT : PRINT
400 GOSUB 220
410 REM
420 REM ** INVERSE EXAMPLES **
430 REM
440 HOME : VTAB 6: PRINT "THERE ARE THREE DIFFERENT WAYS TO"
```

and so on until you are transferring the entire byte by ANDing it with 11111111.

The flip routines are about as straightforward as any. To flip the screen from top to bottom, all you have to do is set up a loop that takes any given line in the top half of the screen, stores it somewhere, and gets its matching symmetrical line on the lower half of the screen. The loop must then move the lower line to the top line, and move the top line (which was stored somewhere) to the lower line.



```

450 PRINT "INVERSE THE HI-RES SC
REEN:": PRINT : PRINT
460 INVERSE : PRINT "TOP TO BOTT
OM": NORMAL
470 PRINT " WITH PROGRAMMABLE DE
LAY": PRINT
480 INVERSE : PRINT "RIGHT TO LE
FT": NORMAL
490 PRINT " WITH PROGRAMMABLE DE
LAY": PRINT
500 INVERSE : PRINT "FAST": NORMAL

510 PRINT " (NO DELAY - FADES IN
)": PRINT
520 GOSUB 220: POKE 34,0
530 HOME : VTAB 6: PRINT "FOR AL
L OF THESE EXAMPLES, FIRST R
EAD"
540 PRINT "WHAT IT SAYS, THEN HI
T ANY KEY TO TURN"
550 PRINT "ON THE GRAPHICS PAGE,
HIT ANY KEY AGAIN TO START
THE ROUTINE, AND HIT ANY KEY
A"
560 PRINT "THIRD TIME TO GET BAC
K TO THE TEXT PAGE"
570 GOSUB 220
580 HOME : VTAB 6: PRINT "EXAMPL
E: INVERSE - FAST"
590 VAR = INV: GOSUB 250: HOME
600 VTAB 6: PRINT "EXAMPLE: INVE
RSE - TOP TO BOTTOM W/DELAY"
610 PRINT : PRINT "IN EACH ROUTI
NE WHERE A DELAY IS"
620 PRINT "POSSIBLE, ALL YOU HAV
E TO DO IS": PRINT : PRINT
630 PRINT "POKE 254,X (WHERE X I
S BETWEEN 1-255)": PRINT : PRINT

```

These are the steps used in flipping row 0 and row 191 (the first and last rows on the screen):

- 1) Move row 0 into the buffer.
- 2) Move row 191 to row 0.
- 3) Move the buffer to row 191.

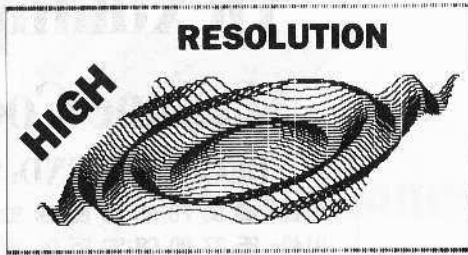
To flip right to left, a similar process is used (but right to left instead of top to bottom). You must flip the sequence of the lower seven bits (or pixels) in order to maintain the picture. For example, if the bit pattern 10110101 were present in the

Program

```

640 PRINT : PRINT "THE BIGGER TH
E NUMBER X, THE LONGER THE D
ELAY": PRINT
650 PRINT "HERE'S X=1":VAR = ITB
:WT = 1: GOSUB 250
660 POKE 34,7: HOME : VTAB 9: PRINT
"HERE'S X=60":WT = 60: GOSUB
250
670 POKE 34,0: HOME
680 VTAB 6: PRINT "EXAMPLE: INVE
RSE - RIGHT TO LEFT W/DELAY"
: PRINT
690 PRINT "HERE'S X=1":WT = 1:VA
R = IRL: GOSUB 250: POKE 34, 7
700 VTAB 9: PRINT "HERE'S X=80":
WT = 80: GOSUB 250
710 POKE 34,0: HOME
730 REM ** FLIP EXAMPLES **
740 REM
750 VTAB 6: PRINT "EXAMPLE: FLIP
- TOP TO BOTTOM W/DELAY": VTAB
9
760 PRINT "HERE'S X=1":WT = 1:VA
R = FTB: GOSUB 250: POKE 34, 7
770 HOME : VTAB 9: PRINT "HERE'S
X=100":WT = 100: GOSUB 250
780 POKE 34,0: HOME : VTAB 6: PRINT
"EXAMPLE: FLIP - RIGHT TO LE
FT W/DELAY"
790 VTAB 9: PRINT "HERE'S X=1":V
AR = FRL:WT = 1: GOSUB 250: POKE
34,7
800 HOME : VTAB 9: PRINT "HERE'S
X=90":WT = 90: GOSUB 250
810 TEXT : HOME : PRINT "END OF
DEMONSTRATION": END
820 HTAB 1: PRINT TAB( 39):: PRINT
: PRINT "CREATING A PICTURE"
: POKE 230,64
830 CALL 62450: HCOLOR= 3: FOR X
= 0 TO 279 STEP 2.3
840 HPLLOT 260,191 TO X,20 + 20 *
SIN (X / 7): NEXT
850 RETURN

```

byte at \$2000 (the first position in the first row), you would want to store 11010110 at \$2027 (the last position on the first row).

In order to preserve the color, the MSB (Most Significant Bit) is not flipped. But, before \$2000 could be moved you would have to store the byte which is currently at \$2027 in a buffer area so as not to lose it. After you finish with the \$2000 byte, flip it and then store it at \$2027. The steps involved with moving the first pair of bytes would be:

- 1) Move \$2027 to the buffer.
- 2) Load \$2000, and flip its bits.
- 3) Store it at \$2027.
- 4) Get the byte from the buffer and flip it.
- 5) Store it at \$2000.

In the case of the flip routines, the stack is used as the buffer (since only one byte will be there at any given time).

SCROLL PROGRAM

This program makes it possible to scroll hi-res page one up, down, right or left (seven pixels at a time), similar to the effect achieved in the game Ultima.

- 1) Type in the SCROLL listing on page 18.

- 2) **BSAVE SCROLL, A\$9380, L\$219**
- 3) Load the picture to be scrolled into hi-res page one.
- 4) **BRUN SCROLL**

The following commands move the picture.

- A Up
- Z Down
- <- Left
- > Right
- ESC Exit program

If you want to use the routines from within a BASIC program to scroll the hi-res screen:

- 1) **BLOAD SCROLL**
- 2) **CALL 38207**
- 3) Set **HIMEM:37376**
- 4) Make the appropriate **CALL**

Scroll Direction

up
down
right
left

CALL

37833 (\$93C9)
37900 (\$940C)
38106 (\$94DA)
38139 (\$94FB)

NOTE: If used in conjunction with ROUTINES, set **HIMEM:36864**.

The hi-res utilities included in the programs ROUTINES and SCROLL show what can be achieved by learning from imitation. What began as a simple attempt to copy the effects of somebody else's program resulted in a unique set of hi-res utilities. Now they can become a new addition to your software library, and may give you ideas for your own programs.



COLLEGIATE MICROCOMPUTER

A JOURNAL DEVOTED TO ALL ASPECTS OF MICROCOMPUTERS IN THE UNDERGRADUATE CURRICULA

COLLEGIATE MICROCOMPUTER is a forum for the exchange of ideas on microcomputers in all areas of college and university life - microcomputers in teaching, research, classroom, laboratory, library, studio, office, planning, athletics, and recreation.

Articles include reviews and accounts of hardware and software uses - descriptions of topics, units and courses using microcomputers - results of research using microcomputers - analyses of experiments in microcomputer uses - student projects - suggestions and tips - experiences with microcomputer consulting and workshops - microcomputer use in office work and material preparation - and - reviews of software, hardware, peripherals, products, and literature.

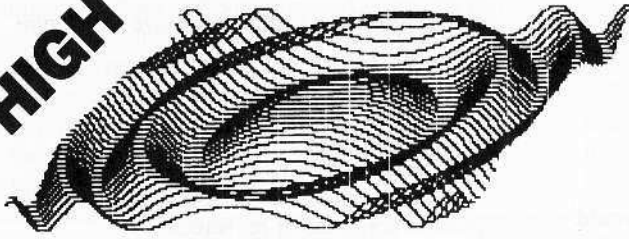
COLLEGIATE MICROCOMPUTER is the only journal of its type, reaching college and university professionals and libraries. Readers are interested in the applications of microcomputers to their special uses in the undergraduate environment.

Volume One, Number One is February 1983. Subscription rates are \$28.00/year for US and \$36.00/year for non-US with non-US AIR MAIL available for \$60.00/year.

COLLEGIATE MICROCOMPUTER, Rose-Hulman Institute of Technology, Terre Haute IN 47803.

HIGH

RESOLUTION



Hexdump Machine Code

BEG: 9000; END: 91FA

```

9140- 00 05 FD A6 FD 00 02 $9E6F
9148- 05 07 0D C0 92 05 06 A0 $5600
9150- 00 B1 06 48 04 FF A9 27 $6A43
9158- 38 E5 FF A0 B1 06 20 05 $7FA4
9160- 91 A4 FF 91 06 A9 27 38 $1009
9168- E5 FF A0 60 20 05 91 91 $5178
9170- 06 E6 FF A4 FF C0 14 D0 $722B
9178- D8 20 04 90 E6 FD A5 FD $5320
9180- C9 C0 D0 BF 60 A2 00 06 $5A4C
9188- FC 0A 00 A2 06 0A 90 09 $5409

```

```

9190- 48 0D 00 03 05 FC 05 FC $F030
9198- 68 CA 10 F1 A5 FC 0A 20 $9240
91A0- 6A 60 A9 00 A0 05 06 A9 $06FF
91A8- 04 05 00 A5 06 A2 00 99 $0AFC
91B0- C0 92 C0 CA D0 F9 18 69 $8CAE
91B8- 00 90 F2 C6 00 D0 EC A5 $2C5D
91C0- 06 69 27 05 06 C9 78 D0 $0004
91C8- DE A9 03 05 FC A0 00 A9 $88C3
91D0- 00 05 06 A9 02 05 00 A5 $6543
91D8- 06 18 69 20 99 00 92 38 $9139

```

```

91E0- E9 20 C8 18 69 04 C9 20 $23EA
91E8- 30 EF C6 00 D0 E9 E6 06 $055E
91F0- A5 06 C9 04 D0 D0 C6 FC $5B47
91F8- D0 D5 60 $E394

```

```

9000- 20 A2 91 60 A5 FE 20 A0 $2806
9008- FC 60 A9 00 05 06 05 00 $8AF0
9010- A9 20 05 07 A9 40 05 09 $80D3
9018- A0 00 B1 00 91 06 C8 D0 $CE38
9020- F9 E6 07 E6 09 A5 07 C9 $48C8
9028- 40 D0 ED 60 A2 00 D0 C0 $D03C
9030- 92 05 06 05 00 D0 00 92 $9475
9038- 05 07 18 69 20 05 09 A0 $2540
9040- 00 B1 00 91 06 C8 C0 28 $726A
9048- D0 F7 20 04 90 E8 E0 C0 $870D

```

```

90A0- A2 00 D0 00 92 05 07 D0 $D3E9
90A8- C0 92 05 06 A0 00 B1 06 $C9FC
90B0- 49 7F 91 06 C8 C0 28 D0 $80FA
90B8- F5 20 04 90 E8 E0 C0 D0 $3C8F
90C0- E1 60 A0 00 A9 01 05 FF $2223
90C8- A2 00 D0 00 92 05 07 D0 $4763
90D0- C0 92 05 06 B1 06 45 FF $CE94
90D8- 91 06 E8 E0 C0 D0 EB 20 $919D
90E0- 04 90 18 26 FF 26 FF D0 $80E7
90E8- 04 66 FF D0 D0 C8 C0 28 $3D4A

```

```

9050- D0 DC 60 A0 00 A9 01 05 $ED26
9058- FF A2 00 D0 00 92 05 07 $5C83
9060- 10 69 20 05 09 D0 C0 92 $9B0F
9068- 05 06 05 00 B1 00 25 FF $F6C3
9070- 91 06 E8 E0 C0 D0 E4 20 $8075
9078- 04 90 30 26 FF 90 DA C8 $2251
9080- C0 20 D0 D1 60 A9 20 05 $CF21
9088- 07 A9 00 05 06 A0 B1 06 $2FDA
9090- 49 7F 91 06 C8 D0 F7 E6 $21D0
9098- 07 A5 07 C9 40 D0 EF 60 $F689

```

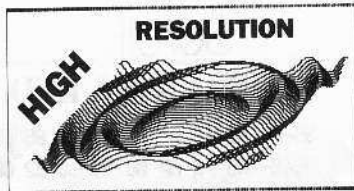
```

90F0- D0 D2 60 A9 00 05 FF A6 $FA9B
90F8- FF D0 00 92 05 07 D0 C0 $ED1E
9100- 92 05 06 A9 BF 30 E5 FF $079F
9108- AA D0 00 92 05 09 D0 C0 $3290
9110- 92 05 00 A0 00 B1 00 80 $E23C
9118- 00 03 B1 06 91 00 A0 00 $C3E9
9120- 03 91 06 C8 C0 28 D0 ED $ED82
9128- 20 04 90 E6 FF A5 FF C9 $0551
9130- 60 D0 C4 60 A9 01 A2 06 $D01B
9138- 9D 00 03 0A CA 10 F9 A9 $7C61

```

Checksums

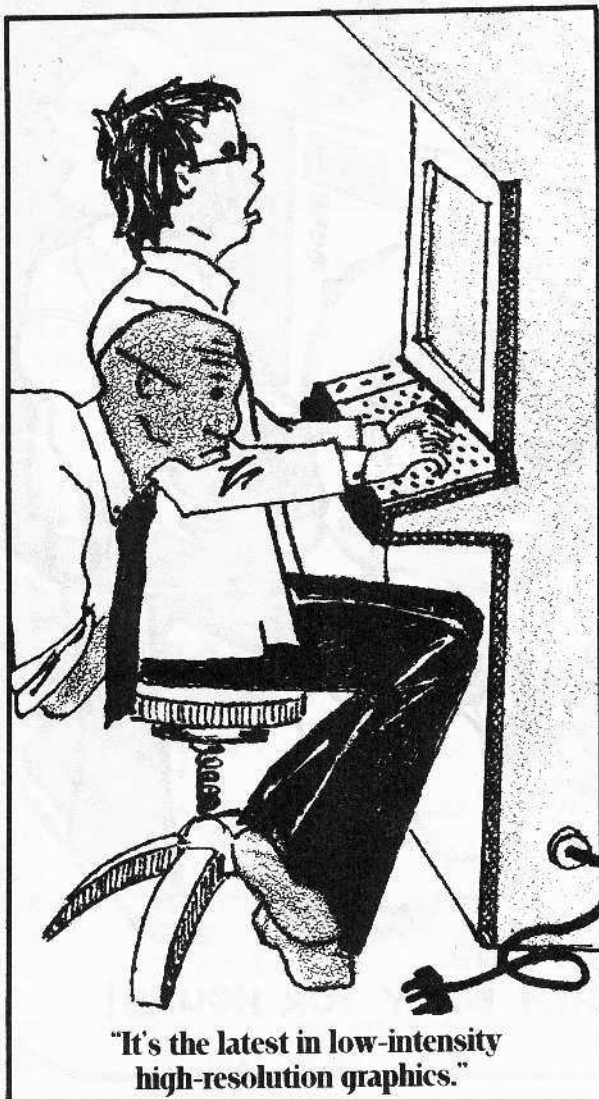
10	- \$BADD	210	- \$094F	410	- \$048B	610	- \$3BA8	810	- \$6301
20	- \$9B13	220	- \$8861	420	- \$A7C0	620	- \$E681	820	- \$3C62
30	- \$4D3B	230	- \$C307	430	- \$F068	630	- \$4915	830	- \$8013
40	- \$197D	240	- \$B01B	440	- \$52F2	640	- \$2882	840	- \$6E17
50	- \$80B0	250	- \$3559	450	- \$0918	650	- \$A59A	850	- \$C52E
60	- \$A51B	260	- \$3581	460	- \$28A1	660	- \$1669		
70	- \$A172	270	- \$A520	470	- \$C721	670	- \$A4DB		
80	- \$738A	280	- \$5895	480	- \$C662	680	- \$2A6D		
90	- \$E74C	290	- \$11A7	490	- \$0457	690	- \$8F99		
100	- \$9BDD	300	- \$3097	500	- \$DE1D	700	- \$66AD		
110	- \$AFAD	310	- \$9EE3	510	- \$342F	710	- \$1847		
120	- \$A357	320	- \$1A05	520	- \$08E2	720	- \$6CE5		
130	- \$0154	330	- \$9AC5	530	- \$BA6F	730	- \$CE14		
140	- \$D9CB	340	- \$00F7	540	- \$6BB7	740	- \$13BF		
150	- \$7997	350	- \$4E9D	550	- \$423B	750	- \$4D7B		
160	- \$C90C	360	- \$157E	560	- \$E7E4	760	- \$61AB		
170	- \$D998	370	- \$30AD	570	- \$8D02	770	- \$05EF		
180	- \$5C21	380	- \$A93B	580	- \$E619	780	- \$1FCF		
190	- \$1706	390	- \$94F7	590	- \$EFE4	790	- \$362F		
200	- \$9906	400	- \$3AFC	600	- \$C601	800	- \$8006		



Program

Scroll

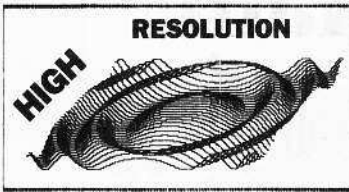
Checksums	
10	- \$BADD
20	- \$9B13
30	- \$4D3B
40	- \$2E52
50	- \$0A8A
60	- \$E7FE
70	- \$E3FA
80	- \$DDA7
90	- \$5E46
100	- \$3F87
110	- \$6DC1
120	- \$B7E0
130	- \$5BAE
140	- \$697E
150	- \$31D5
160	- \$F3E8
170	- \$5C94
180	- \$E00E
190	- \$929A
200	- \$D884
210	- \$EC78
220	- \$577D
230	- \$50EA
240	- \$393F
250	- \$60E9
260	- \$E734
270	- \$994C
280	- \$8E38
290	- \$912C
300	- \$179A



```

10 REM
20 REM ** SCROLL DEMO **
30 REM
40 TEXT : HOME
50 VTAB 2: HTAB 1: PRINT "SCROLL
  DEMONSTRATION"
60 POKE 34,3: VTAB 4
70 PRINT "LOADING SCROLL": PRINT
  CHR$(4)"BLOAD SCROLL"
80 HIMEM: 37375
90 PRINT : PRINT "DO YOU HAVE A
  HI-RES PICTURE TO LOAD ?"; CHR$(
  8);
100 GET A$: IF A$ < > "Y" THEN
  GOSUB 270: GOTO 140
110 HTAB 1: PRINT "NAME OF PICTU
  RE =>"; TAB( 39)
120 HTAB 19: INPUT "";A$: PRINT
  : PRINT "LOADING "A$
130 PRINT CHR$(4);"BLOAD";A$;"
  ,A$2000"
140 HOME
150 PRINT "THIS SCROLL ROUTINE S
  IMULATES THE"
160 PRINT "EFFECTS OF THE GAME U
  LTIMA. YOU CAN"
170 PRINT "SCROLL THE SCREEN UP,
  DOWN, RIGHT AND LEFT."
180 PRINT : PRINT "THE COMMANDS
  ARE:": PRINT : PRINT
190 PRINT " A = UP": PRINT "
  Z = DOWN"
200 PRINT " -> = RIGHT": PRINT
  " <- = LEFT"
210 PRINT " ESC = EXIT PROGRAM"

220 VTAB 23: HTAB 11: PRINT "HIT
  ANY KEY TO BEGIN"
230 POKE - 16368,0
240 IF PEEK ( - 16384) < 128 THEN
  240
250 POKE - 16368,0: TEXT : HOME
260 CALL 37760
270 HTAB 1: PRINT TAB( 39);: PRINT
  : PRINT "CREATING A PICTURE"
280 POKE 230,32: CALL 62450: HCOLOR=
  3: FOR X = 20 TO 259 STEP 2,
  3
290 HPLLOT 140,171 TO X,20 + 20 *
  SIN ((X - 20) / 25.5): NEXT
300 RETURN
  
```



Hexdump

Scroll
BEG: 9380; END: 9597

9380- 2C 50 C0 2C 52 C0 2C 54 \$7D8B
 9380- C0 2C 57 C0 20 3F 95 20 \$592A
 9390- BE 93 C9 C1 D0 03 20 C9 \$3807
 9390- 93 C9 DA D0 03 20 0C 94 \$4A49
 93A0- C9 00 D0 03 20 DA 94 C9 \$02A8
 93A0- 95 D0 03 20 FB 94 C9 9B \$77FD
 93B0- D0 D0 20 58 FC 2C 51 C0 \$AA32
 93B0- 2C 10 C0 4C D0 03 AD 10 \$C9E6
 93C0- C0 AD 00 C0 C9 00 90 F9 \$C57B
 93C0- 60 A9 00 00 53 94 00 A7 \$037A

93D0- 94 A9 18 00 5F 94 00 B3 \$5989
 93D0- 94 A9 69 00 60 94 00 B4 \$7511
 93E0- 94 A9 38 00 6E 94 00 C5 \$094C
 93E0- 94 A9 E9 00 6F 94 00 C6 \$F440
 93F0- 94 A9 B9 00 7F 94 00 B3 \$DC3E
 93F0- 94 A9 C0 00 A2 94 A9 07 \$4ADE
 9400- 00 D6 94 A9 E8 00 7D 94 \$1AA7
 9400- 20 4F 94 60 A9 BF 00 53 \$CFA6
 9410- 94 A9 B9 00 A7 94 A9 38 \$6AA1
 9410- 00 5F 94 00 B3 94 A9 E9 \$1400

9420- 00 60 94 00 B4 94 A9 18 \$1D89
 9420- 00 6E 94 00 C5 94 A9 69 \$6587
 9430- 00 6F 94 00 C6 94 A9 06 \$8E71
 9430- 00 7F 94 A9 00 00 B3 94 \$1050
 9440- A9 07 00 A2 94 A9 00 00 \$D15C
 9440- D6 94 A9 CA 00 7D 94 20 \$471F
 9450- A6 94 A2 00 00 00 92 05 \$F3E2
 9450- FD 00 C0 92 05 FC 0A 18 \$C5A7
 9460- 69 07 AA 00 00 92 05 FF \$A344
 9460- 00 C0 92 05 FE 0A 38 E9 \$7800

9470- 07 AA A0 00 B1 FE 91 FC \$7690
 9470- C0 C0 20 D0 F7 E8 E0 B9 \$3040
 9480- D0 D2 A2 B9 00 00 92 05 \$5232
 9480- FD 10 69 20 05 FF 00 C0 \$61EB
 9490- 92 05 FC 05 FE A0 00 B1 \$F485
 9490- FE 91 FC C0 C0 20 D0 F7 \$03A8
 94A0- E8 E0 C0 D0 DF 60 A2 00 \$C97F
 94A0- 00 00 92 05 FD 00 C0 92 \$8616
 94B0- 05 FC 0A 18 69 B9 AA 00 \$3C10
 94B0- 00 92 18 69 20 05 FF 00 \$340A

94C0- C0 92 05 FE 0A 38 E9 B9 \$B45E
 94C0- AA A0 00 B1 FC 91 FE C0 \$7C5E
 94D0- C0 20 D0 F7 E8 E0 07 D0 \$0EEA
 94D0- CF 60 A9 00 00 25 95 A9 \$658A
 94E0- C0 00 2F 95 00 30 95 00 \$5F11

94E0- 29 95 A9 20 00 32 95 A9 \$C368
 94F0- 00 00 2C 95 00 35 95 20 \$F20A
 94F0- 18 95 60 A9 27 00 25 95 \$5DE0
 9500- A9 00 00 2F 95 00 30 95 \$142C
 9500- 00 29 95 A9 C0 00 2C 95 \$0C9C

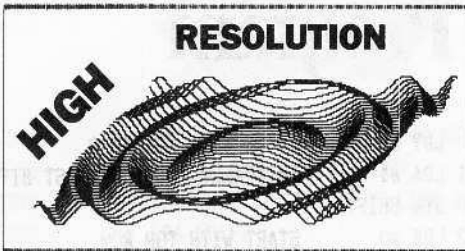
9510- 00 35 95 A9 FF 00 32 95 \$6801
 9510- A2 00 00 00 92 05 FD 00 \$A490
 9520- C0 92 05 FC A0 00 B1 FC \$AE0A
 9520- 40 C0 B1 FC 00 91 FC C0 \$739F
 9530- C0 C0 20 D0 F5 00 60 91 \$CC0E
 9530- FC E0 E0 C0 D0 DC 60 A9 \$C3FB

9540- 00 A0 05 FC A9 04 05 FE \$250E
 9540- A5 FC A2 00 99 C0 92 C0 \$DF04
 9550- CA D0 F9 18 69 00 90 F2 \$AA31
 9550- C6 FE D0 EC A5 FC 69 27 \$0ED9

9560- 05 FC C9 70 D0 DE A9 03 \$972C
 9560- 05 FB A0 00 A9 00 05 FC \$3C0C
 9570- A9 02 05 FE A5 FC 10 69 \$A151
 9570- 20 99 00 92 30 E9 20 C0 \$1300
 9580- 10 69 04 C9 20 30 EF C6 \$7E0A
 9580- FE D0 E9 E6 FC A5 FC C9 \$7D40
 9590- 04 D0 D0 C6 FB D0 D5 60 \$743E



IT'S A NEW ONE...
KEEPS HIM BUSY FOR HOURS!



Program

```

1000 *****
1010 *                               *
1020 *   HI-RES ROUTINES           *
1030 *                               *
1040 *   By: Mike Scanlin          *
1050 *                               *
1060 *****
1080 START   .EQ #9000
1100 PTR1    .EQ #06   2 BYTE POINTER
1110 PTR2    .EQ #08   2 BYTE POINTER
1120 GEN1    .EQ #FC   GENERAL STORAGE
1130 GEN2    .EQ #FD   GENERAL STORAGE
1140 DELAY   .EQ #FE
1150 SHIFT   .EQ #FF
1160 TABLE  .EQ #0300  TO BE USED BY FLIP (R TO L)
1170 HIRESH  .EQ #9200  TABLE OF HI-RES HIBYTES
1180 HIRESL   .EQ HIRESH+#C0 TABLE OF HI-RES LOBYTES
1190 WAIT    .EQ #FCA8
1210 *
1220 * SET UP HI-RES BASE ADDRESS TABLE
1230 *
1250 SETUP   JSR CALCLO  CALC BASE HI-RES ADDRESSES
1260         RTS
1280 * DELAY LOOP
1300 DELAYLP LDA DELAY
1310         JSR WAIT
1320         RTS
1340 *
1350 * TRANSPOSE - FAST
1360 *
1380 TRANS   LDA #0
1390         STA PTR1    LOBYTE OF PAGE1 PTR
1400         STA PTR2    LOBYTE OF PAGE2 PTR
1410         LDA #20
1420         STA PTR1+1  HIBYTE OF PAGE1 PTR
1430         LDA #40
1440         STA PTR2+1  HIBYTE OF PAGE2 PTR
1460 TLOOP   LDY #0
1470 TLP2    LDA (PTR2),Y LOAD FROM PAGE2
1480         STA (PTR1),Y STORE ON PAGE1
1490         INY
1500         BNE TLP2    MOVE #FF BYTES
1520         INC PTR1+1  INC HIBYTE OF PTR1 & PTR2
1530         INC PTR2+1
1540         LDA PTR1+1
1550         CMP #40     DONE WITH PAGE?
1560         BNE TLOOP   IF NOT, MOVE

```

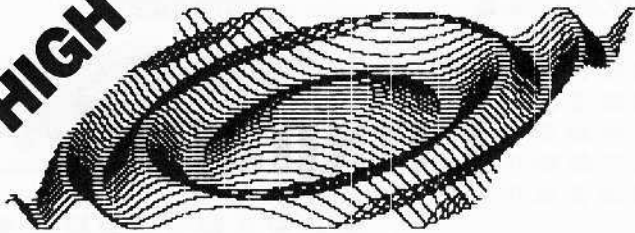
```

1570         RTS
1590 *
1600 * TRANSPOSE - TOP TO BOTTOM W/DELAY
1610 *
1630 TRANSTB LDX #0      START WITH TOP LINE
1640 TRTBLP1 LDA HIRESL,X GET LOBYTE
1650         STA PTR1
1660         STA PTR2
1670         LDA HIRESH,X GET HIBYTE OF LINE (PAGE1)
1680         STA PTR1+1
1690         CLC
1700         ADC #20     GET HIBYTE OF LINE (PAGE2)
1710         STA PTR2+1
1730         LDY #0     MOVE ONE LINE (#28 BYTES)
1740 TRTBLP2 LDA (PTR2),Y
1750         STA (PTR1),Y
1760         INY
1770         CPY #28
1780         BNE TRTBLP2
1800         JSR DELAYLP DELAY LOOP
1820         INX       INCREASE LINE NUMBER
1830         CPX #C0   DONE WITH SCREEN?
1840         BNE TRTBLP1 IF NOT, MOVE NEXT LINE
1850         RTS
1870 *
1880 * TRANSPOSE - RIGHT TO LEFT W/DELAY
1890 *
1910 TRANSRL LDY #0      START WITH HORZ. COLUMN 0
1920 TRRLLP1 LDA #1      SET LSB
1930         STA SHIFT
1940 TRRLLP2 LDX #0     START WITH VERT. LINE 0
1960 TRRLLP3 LDA HIRESH,X GET HIBYTE (PAGE1)
1970         STA PTR1+1
1980         CLC
1990         ADC #20     GET HIBYTE (PAGE2)
2000         STA PTR2+1
2010         LDA HIRESL,X GET LOBYTE
2020         STA PTR1
2030         STA PTR2
2050         LDA (PTR2),Y GET BYTE AT ROW X,COL Y (PAGE2)
2060         AND SHIFT  CUT OFF PART OF BYTE
2070         STA (PTR1),Y PUT ON PAGE1
2080         INX       INC ROW COUNTER
2090         CPX #C0   DONE WITH COLUMN?
2100         BNE TRRLLP3 IF NOT, DO NEXT BYTE
2120         JSR DELAYLP DELAY LOOP
2140         SEC

```


RESOLUTION

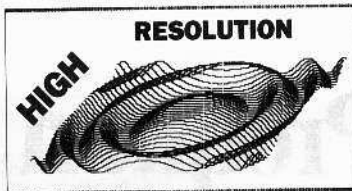
HIGH



```
2150      ROL SHIFT      INC NUMBER OF PIXELS TO SHOW IN
2160      BCC TRRLP2     CURRENT COLUMN
2180      INY             IF=8 THEN INC COLUMN COUNTER
2190      CPY #S28       DONE WITH SCREEN?
2200      BNE TRRLP1     IF NOT, START NEXT COLUMN
2210      RTS
2230 *
2240 * INVERSE - FAST
2250 *
2270 INVERSE LDA #S20     HIBYTE OF PTR
2280      STA PTR1+1
2290      LDA #0          LOBYTE OF PTR (PTR=$2000)
2300      STA PTR1
2320      TAY             START WITH Y=0
2330 INVLP  LDA (PTR1),Y  LOAD BYTE
2340      EOR #S7F        FLIP ALL BITS EXCEPT MSB
2350      STA (PTR1),Y    STORE INVERSED BYTE
2360      INY
2370      BNE INVLP       DONE WITH SFF BYTES?
2390      INC PTR1+1     IF YES, INC HIBYTE OF PTR
2400      LDA PTR1+1
2410      CMP #S40        DONE WITH PAGE?
2420      BNE INVLP       IF NOT, INVERSE ANOTHER SFF BYTES
2430      RTS
2450 *
2460 * INVERSE - TOP TO BOTTOM W/DELAY
2470 *
2490 INVTB  LDX #0        START WITH TOP ROW
2500 ITBLP1 LDA HIRESH,X  GET HIBYTE
2510      STA PTR1+1
2520      LDA HIRESL,X    GET LOBYTE
2530      STA PTR1
2550      LDY #0
2560 ITBLP2 LDA (PTR1),Y  LOAD BYTE IN ROW X, COL Y
2570      EOR #S7F        FLIP ALL BITS EXCEPT MSB
2580      STA (PTR1),Y    STORE INVERSED BYTE
2590      INY
2600      CPY #S28        DONE WITH ROW?
2610      BNE ITBLP2     IF NOT, DO NEXT BYTE
2630      JSR DELAYLP    DELAY LOOP
2650      INX             INC ROW COUNTER
2660      CPX #SCO        DONE WITH SCREEN?
2670      BNE ITBLP1     IF NOT, DO NEXT ROW
2680      RTS
2700 *
2710 * INVERSE RIGHT TO LEFT W/DELAY
2720 *
```

Program

```
2740 INVRL  LDY #0        START WITH COLUMN 0
2750 INVRLP1 LDA #1        INITIALLY INVERSE FIRST BIT
2760      STA SHIFT
2770 INVRLP2 LDX #0        START WITH TOP ROW
2780 INVRLP3 LDA HIRESH,X  GET HIBYTE
2790      STA PTR1+1
2800      LDA HIRESL,X    GET LOBYTE
2810      STA PTR1
2830      LDA (PTR1),Y    GET BYTE AT ROW X, COL Y
2840      EOR SHIFT       FLIP SOME BITS
2850      STA (PTR1),Y    STORE BYTE
2860      INX             INC ROW COUNTER
2870      CPX #SCO        DONE WITH COLUMN?
2880      BNE INVRLP3     IF NOT, DO NEXT ROW
2900      JSR DELAYLP    DELAY LOOP
2920      CLC
2930      ROL SHIFT       INCR NUMBER OF FLIPPED BITS
2940      ROL SHIFT
2950      BCS IRLLP5
2960      ROR SHIFT
2970      BNE INVRLP2
2980 IRLLP5  INY             INC COLUMN COUNTER
2990      CPY #S28        DONE WITH SCREEN?
3000      BNE INVRLP1     IF NOT, DO NEXT COLUMN
3010      RTS
3030 *
3040 * FLIP - TOP TO BOTTOM W/DELAY
3050 *
3070 FLIPTB LDA #0        START WITH VERT. LINE 0
3080      STA SHIFT
3090 FLOOP  LDX SHIFT     GET VERT. LINE NUMBER
3100      LDA HIRESH,X    GET HIBYTE OF ORG ROW
3110      STA PTR1+1
3120      LDA HIRESL,X    GET LOBYTE OF ORG ROW
3130      STA PTR1
3150      LDA #SBF        CALC DEST ROW NUMBER
3160      SEC
3170      SBC SHIFT
3180      TAX
3190      LDA HIRESH,X    GET HIBYTE OF DEST ROW
3200      STA PTR2+1
3210      LDA HIRESL,X    GET LOBYTE OF DEST ROW
3220      STA PTR2
3240      LDY #0          MOVE ORG ROW -> TABLE
3250 FLP2   LDA (PTR2),Y  DEST ROW -> ORG ROW
3260      STA TABLE     TABLE -> DEST ROW
3270      LDA (PTR1),Y
3280      STA (PTR2),Y
3290      LDA TABLE
3300      STA (PTR1),Y
3310      INY
3320      CPY #S28        DONE WITH ROW?
```



Program

```

3330 BNE FLP2 IF NOT, MOVE NEXT BYTE
3350 JSR DELAYLP DELAY LOOP
3370 INC SHIFT INC VERT LINE NUMBER
3380 LDA SHIFT
3390 CMP #960 FLIPPED TOP HALF OF SCREEN?
3400 BNE FLOOP IF NOT, FLIP NEXT PAIR OF ROWS
3410 RTS
3430 *
3440 * FLIP - RIGHT TO LEFT W/DELAY
3450 *
3470 FLIPRL LDA #1 SET UP FLIP BIT TABLE
3480 LDX #6
3490 FLP3 STA TABLE,X
3500 ASL
3510 DEX
3520 BPL FLP3
3540 LDA #0 START AT VERT. LINE 0
3550 STA GEN2
3560 FLMAIN LDX GEN2 GET VERT. LINE NUMBER
3570 LDA HIRESH,X GET HIBYTE
3580 STA PTR1+1
3590 LDA HIRESL,X GET LOBYTE
3600 STA PTR1
3620 LDY #0 START WITH LEFT MOST BYTE IN LINE
3630 FLP4 LDA (PTR1),Y
3640 PHA STORE LEFT BYTE
3650 STY SHIFT SAVE OFFSET LOCATION OF LEFT BYTE
3660 LDA #27 CALC ITS SYMMETRICAL POSITION
3670 SEC
3680 SBC SHIFT
3690 TAY
3700 LDA (PTR1),Y GET BYTE FROM THIS POSITION
3710 JSR FLIPBITS FLIP ITS BITS
3720 LDY SHIFT GET ORG HORZ. POSITION
3730 STA (PTR1),Y STORE FLIPPED BYTE
3740 LDA #27 CALC WHERE SECOND BYTE CAME FROM
3750 SEC
3760 SBC SHIFT
3770 TAY
3780 PLA GET ORIGINAL BYTE
3790 JSR FLIPBITS FLIP ITS BITS
3800 STA (PTR1),Y STORE FLIPPED BYTE
3810 INC SHIFT INC HORZ. COUNTER
3820 LDY SHIFT
3830 CPY #14 FLIPPED HALF OF ROW?
3840 BNE FLP4 IF NOT, FLIP NEXT PAIR
3860 JSR DELAYLP DELAY LOOP
3880 INC GEN2 GET VERT. LINE NUMBER
3890 LDA GEN2
3900 CMP #960 DONE WITH SCREEN?
3910 BNE FLMAIN IF NOT, FLIP NEXT ROW
3920 RTS

```

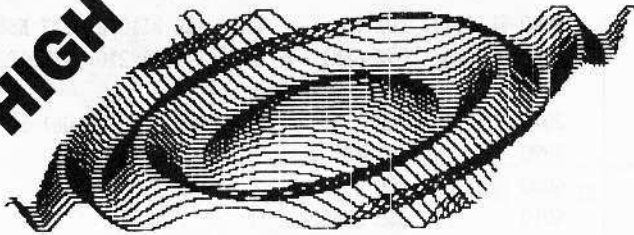
```

3940 FLIPBITS LDX #0 FLIP ALL BITS EXCEPT MSB
3950 STX GEN1 BITS: 76543210 -> 70123456
3960 ASL
3970 PHP PRESERVE MSB (COLOR) IN CARRY
3990 LDX #6
4000 FLB1 ASL
4010 BCC FLB2
4020 PHA
4030 LDA TABLE,X
4040 ORA GEN1
4050 STA GEN1
4060 PLA
4070 FLB2 DEX
4080 BPL FLB1
4100 LDA GEN1
4110 ASL
4120 PLP RECALL MSB IN CARRY
4130 ROR
4140 RTS
4160 *
4170 * ROUTINE TO GENERATE HIBYTE AND LOBYTE TABLE OF
4180 * HI-RES PAGE1 BASE ADDRESSES
4190 *
4210 CALCLO LDA #0
4220 TAY
4230 STA PTR1
4250 CL1 LDA #4
4260 STA PTR2
4270 CL2 LDA PTR1
4280 CL3 LDX #8
4290 CL4 STA HIRESL,Y
4300 INY
4310 DEX
4320 BNE CL4
4330 CLC
4340 ADC #980
4350 BCC CL3
4360 DEC PTR2
4370 BNE CL2
4380 LDA PTR1
4390 ADC #27
4400 STA PTR1
4410 CMP #978
4420 BNE CL1
4440 CALCHI LDA #3 CALC HIBYTE FOR HI-RES BASE ADDR
4450 STA GEN1
4470 LDY #0
4480 CH1 LDA #0
4490 STA PTR1
4500 CH2 LDA #2
4510 STA PTR2
4520 CH3 LDA PTR1

```

HIGH

RESOLUTION



Program

4530 CH4	CLC	4630	DEC PTR2
4540	ADC #20	4640	BNE CH3
4550	STA HIRESH,Y	4650	INC PTR1
4560	SEC	4660	LDA PTR1
4570	SBC #20	4670	CMP #4
4580	INY		
4590	CLC	4680	BNE CH2
4600	ADC #4	4690	DEC GEN1
4610	CMP #20	4700	BNE CH1
4620	BMI CH4	4710	RTS

Highest Quality, Lifetime Guarantee!

DISKETTES

\$1.65 5 1/4" soft sectored

- Our own custom-manufactured brand
- Certified for Single & Double Density
- Can be used double-sided on Apple, Atari, IBM, and many other computers
- Hub ring, tight seams, loose-fitting jackets, white Tyvek sleeves

- * order in any quantity from 1 up
- * \$155 for 100 - \$149 for bulk pak of 100 (sleeves & labels separate, write-protect notch on one side)
- * add \$2 to total order for shipping

Hard plastic stand-up 10-diskette carrying cases \$2.75 each, 4 for \$10 (specify choices: beige, black, blue, green, grey, red, yellow)

Smoked-plastic jumbo-size flip-top 75-diskette file cases \$19.50 each

VF ASSOCIATES

6327 Western Ave., N.W., Wash., D.C. 20015
(202) 363-1313

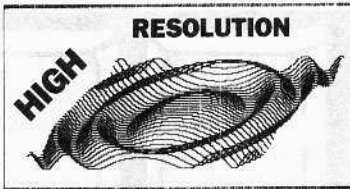
Program

Scroll

```

1000 *****
1010 * *
1020 * HI-RES SCROLL *
1030 * *
1040 * By: Mike Scanlin *
1050 * *
1060 *****
1080 START .OR $9380
1100 GEN1 .EQ #FB
1110 PTR1 .EQ #FC
1120 PTR2 .EQ #FE
1130 HOME .EQ #FC58
1140 KYBD .EQ #C000
1150 STROBE .EQ #C010
1170 * TABLE OF HI-RES PAGE1 BASE ADDRESSES
1180 * HIBYTES AT $9200, LOBYTES AT $92C0
1200 HIRESH .EQ #9200
1210 HIRESL .EQ HIRESH+#C0
1230 BIT #C050 SELECT GRAPHICS MODE
1240 BIT #C052 SELECT FULLSCREEN MODE
1250 BIT #C054 SELECT PAGE1
1260 BIT #C057 SELECT HI-RES
1280 * CALCULATE HI-RES PAGE1 BASE ADDRESSES
1300 JSR CALCLO
1320 *
1330 * PROGRAM STARTS HERE
1340 *
1360 CK0 JSR GETAKEY
1370 CMP #C1 'A'=UP
1380 BNE CK1
1390 JSR SCRUP
1400 CK1 CMP #SDA 'Z'=DOWN
1410 BNE CK2
1420 JSR SCRDOWN
1430 CK2 CMP #S88 '<'=RIGHT
1440 BNE CK3
1450 JSR SCRRIGHT
1460 CK3 CMP #S95 '->'=LEFT
1470 BNE CK4
1480 JSR SCRLEFT
1490 CK4 CMP #S9B 'ESC' EXITS
1500 BNE CK0
1510 JSR HOME
1520 BIT #C051 SELECT TEXT NODE
1530 BIT STROBE CLEAR KEYBOARD STROBE
1540 JMP #03D0 EXIT TO BASIC
1560 GETAKEY LDA STROBE CLEAR KEYBOARD STROBE
1570 GLOOP LDA KYBD CHECK KEYBOARD
1580 CMP #S80 KEY PRESSED?

```

Program

Scroll

```

1590      BCC GLOOP      IF NOT, GO BACK
1600      RTS
1620 *
1630 * ALL FOUR ROUTINES USE SELF MODIFYING CODE
1640 *
1660 SCRUP   LDA #0      SCROLL UP
1670      STA VERT+4
1680      STA SAVETOP+1
1690      LDA #918      OPCODE FOR 'CLC'
1700      STA CCARRY1
1710      STA CCARRY2
1720      LDA #969      OPCODE FOR 'ADC #'
1730      STA ADD1
1740      STA ADD2
1750      LDA #938      OPCODE FOR 'SEC'
1760      STA SCARRY1
1770      STA SCARRY2
1780      LDA #9E9      OPCODE FOR 'SBC #'
1790      STA SUB1
1800      STA SUB2
1810      LDA #9B9
1820      STA OP1+1
1830      STA OP2+1
1840      LDA #9C0
1850      STA OP3+1
1860      LDA #7
1870      STA OP4+1
1880      LDA #9E8      OPCODE FOR 'INX'
1890      STA INX
1900      JSR VERT
1910      RTS
1930 SCRDNW LDA #9BF      SCROLL DOWN
1940      STA VERT+4
1950      LDA #9B9
1960      STA SAVETOP+1
1970      LDA #938      OPCODE FOR 'SEC'
1980      STA CCARRY1
1990      STA CCARRY2
2000      LDA #9E9      OPCODE FOR 'SBC #'
2010      STA ADD1
2020      STA ADD2
2030      LDA #918      OPCODE FOR 'CLC'
2040      STA SCARRY1
2050      STA SCARRY2
2060      LDA #969      OPCODE FOR 'ADC #'
2070      STA SUB1
2080      STA SUB2
2090      LDA #6
2100      STA OP1+1

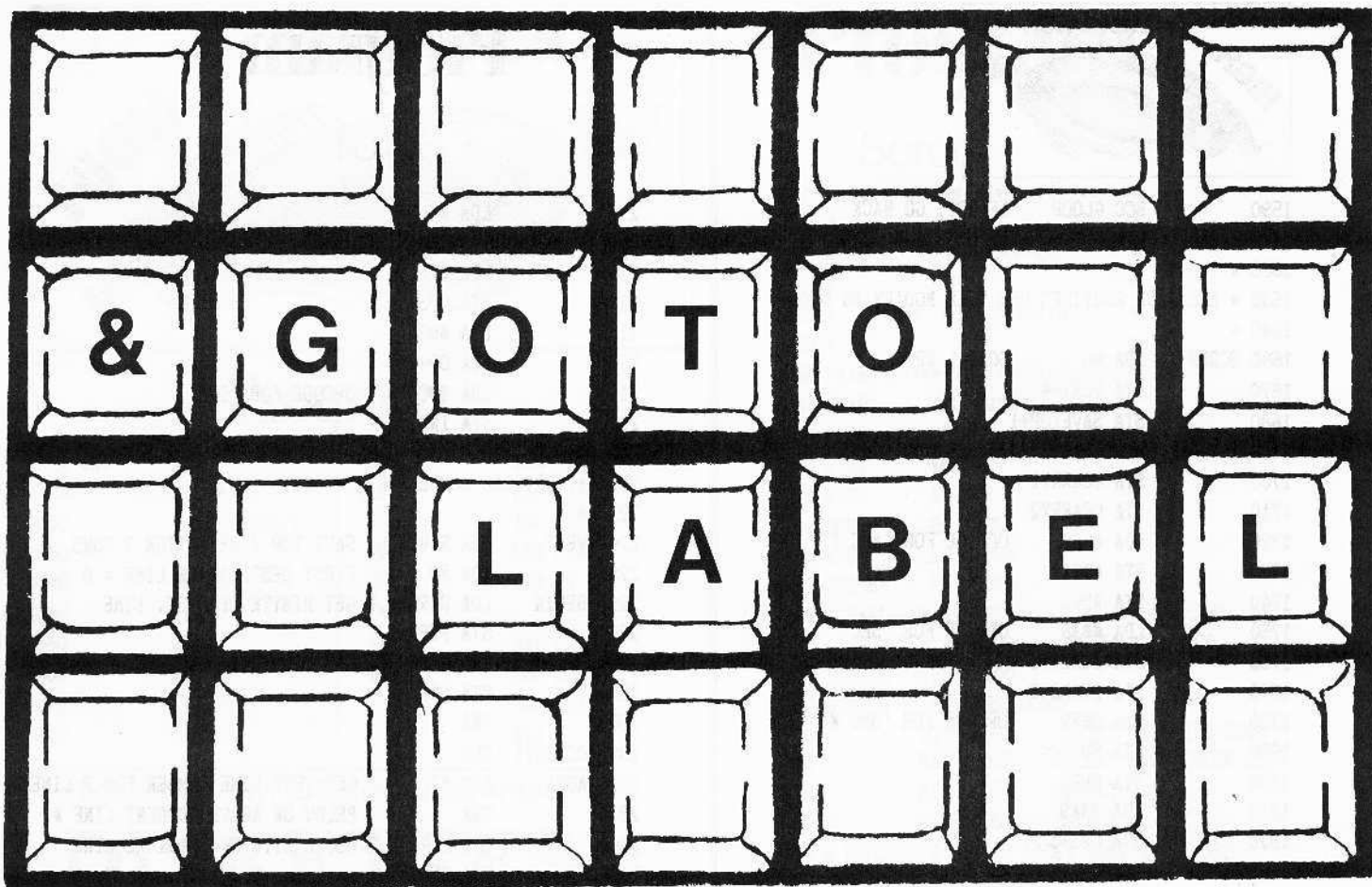
```

```

2110      LDA #0
2120      STA OP2+1
2130      LDA #7
2140      STA OP3+1
2150      LDA #9C0
2160      STA OP4+1
2170      LDA #9CA      OPCODE FOR 'DEX'
2180      STA INX
2200 *
2210 * VERTICAL SCROLLING ROUTINE
2220 *
2240 VERT   JSR SAVETOP  SAVE TOP 7 OR BOTTOM 7 ROWS
2250      LDX #0          FIRST DESTINATION LINE = 0
2260 BEGIN  LDA HIRESH,X GET HIBYTE OF DEST. LINE
2270      STA PTR1+1
2280      LDA HIRESL,X GET LOBYTE OF DEST. LINE
2290      STA PTR1
2300      TXA
2310 CCARRY1 CLC
2320 ADD1   ADC #7          GET VERT LINE NUMBER FOR 7 LINES
2330      TAX              BELOW OR ABOVE CURRENT LINE #
2340      LDA HIRESH,X GET HIBYTE OF ORIGINAL LINE
2350      STA PTR2+1
2360      LDA HIRESL,X GET LOBYTE OF ORIGINAL LINE
2370      STA PTR2
2380      TXA
2390 SCARRY1 SEC
2400 SUB1   SBC #7
2410      TAX
2420      LDY #0
2430 LP3   LDA (PTR2),Y LOAD FROM ORIGINAL LINE
2440      STA (PTR1),Y STORE IN DESTINATION LINE
2450      INY
2460      CPY #928        DONE WITH LINE?
2470      BNE LP3         IF NOT, GET NEXT BYTE
2480 INX   INX              INC VERT LINE COUNTER
2490 OP1   CPY #9B9        DONE WITH SCREEN?
2500      BNE BEGIN      IF NOT, DO THE NEXT LINE
2520 OP2   LDX #9B9        RECALL TOP 7 OR BOTTOM 7 LINES
2530 RCLOOP1 LDA HIRESH,X FROM PAGE2
2540      STA PTR1+1
2550      CLC
2560      ADC #920
2570      STA PTR2+1
2580      LDA HIRESL,X
2590      STA PTR1
2600      STA PTR2
2610      LDY #0
2620 RCLOOP2 LDA (PTR2),Y

```

continued on page 39



By Robb Canfield

Requirements:

- Apple II with 48K
- At least one disk drive
- Blank, initialized disk

GOTO Label is a utility which provides the user with the capacity to use true labels instead of line numbers. It was developed out of the frustration caused by constantly having to remember, by line number, the location of routines in long programs. (SOUND at 10000, GET INPUT at 20000, and COLLISION at 15060, etc.).

GOTO Label allows labels to be defined and branched to through the use of the ampersand command. (A branch in Applesoft is any command that alters the flow: GOTO, GOSUB and ON.)

The companion program, Replace, will substitute normal line numbers for all of the labels, thus converting the program to a normal Applesoft file for later compilation.

Entering GOTO Label

To place GOTO Label in memory, follow these steps:

- 1) Boot the 3.3 master disk.
 - 2) Clear the Applesoft program from memory.
- FP

- 3) Enter the monitor.
CALL -151
- 4) Enter the hexadecimal listing (if you have an assembler, enter the source code listing).
- 5) Return to BASIC.
3D0G
- 6) Save GOTO Label.
BSAVE GOTO LABEL,A\$803,L\$143
- 7) Return to BASIC.
3D0G
- 8) Save the program.

How to Use GOTO Label

To activate GOTO Label, simply BRUN GOTO LABEL. Any Applesoft program in memory at this time will be destroyed. When the prompt reappears (I), the program is ready. All of the normal Applesoft commands are available, with the additional advantage of labels.

There are two terms for labels: the source label and the target label. The source label follows a branch command (GOSUB, GOTO, or ON). The target label is where the program should branch. In other words, the source label is the FROM location and the target label is the TO location.

Source Labels

The source label must be enclosed in quotes and must follow an ampersand branch command (&GOTO, &GOSUB and &ON). When the program finds a quote, all characters, including spaces, are accepted until the end of the line is reached or until a control character is encountered. If no quote is found after the branch command, Applesoft will handle it normally. Below are some examples.

10 &GOTO "HELP"

This line will GOTO the label "HELP".

20 &GOSUB 23

In this line, the program will GOSUB to line 23. Since it is not enclosed in quotes, it is not a label.

30 &GOTO SOUND

A SYNTAX ERROR will be generated by this line because it does not contain a line number and is not enclosed in quotes.

40 &ON A-4 GOTO 23,"HELLO",45,"SOUND"

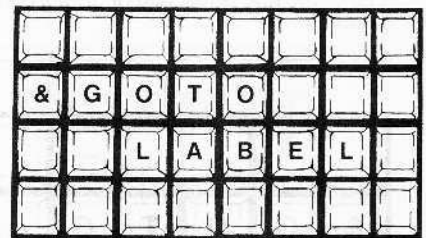
Line 40 will GOTO any of the following: line 23, the "HELLO" label, line 45, or the "SOUND" label. As this example shows, line numbers and labels can be freely mixed within an ON statement. Only the ON is preceded by an ampersand; the branch type (GOTO/GOSUB) is never preceded by an ampersand when inside an ON statement.

Program

```

1000 *-----
1010 * HANDLES THE AMPERSAND COMMANDS
1020 * FOR &GOTO, &GOSUB AND &ON
1030 *-----
1040 *-----
1050 * APPLESOFT TOKENS.
1060 *-----
1070 .OR $803
1080 .TF GOTO LABEL1
00AB- 1090 GOTO.T .EQ $AB GOTO COMMAND
00B0- 1100 GOSUB.T .EQ $B0 GOSUB COMMAND
00B4- 1110 ON.T .EQ $B4 THE ON COMMAND
0022- 1120 QUOTE .EQ $22
00B2- 1130 REM.T .EQ $B2
00AF- 1140 AMPER.T .EQ $AF
003A- 1150 EOL .EQ $3A (A COLON ":")
0021- 1160 LABEL.T .EQ $21 DETERMINES LEGAL LABELS
0020- 1170 SPACE.T .EQ $20 A SPACE
002C- 1180 COMMA .EQ $2C A COMMA ","
1190 *-----
1200 * PAGE ZERO EQUATES.
1210 *-----
009B- 1220 LOWTR .EQ $9B
009D- 1230 FAC .EQ $9D TEMP VARIABLE
00A0- 1240 FIRST.OFFSET .EQ FAC+3
0067- 1250 TXTTAB .EQ $67
00B8- 1260 TXTPTR .EQ $B8
00A0- 1270 TOKEN.FOUND .EQ FAC+3
0076- 1280 CURLIN .EQ $76 CURR. LINE #
1290 *-----
1300 * ROUTINES USED BY APPLESOFT.
1310 *-----
00B1- 1320 CHRGET .EQ $B1 GET NEXT CHAR.
D959- 1330 GO2 .EQ $D959 BASIC ROUTINE
DEC9- 1340 SYNERR .EQ $DEC9 ? SYNTAX ERROR
D9A6- 1350 REMN .EQ $D9A6
D64B- 1360 SCRATCH .EQ $D64B CLR VAR & PTRS
D93E- 1370 GOTO.FP .EQ $D93E NORM GOTO CMND
D828- 1380 DO.NORMAL .EQ $D828 DO CMD IN A-REG
D7D2- 1390 NEWSTT .EQ $D7D2 DO NEW STATEMENT
D3D6- 1400 CHKMEM .EQ $D3D6 GOSUBS TOO DEEP?
D9F8- 1410 ONCNT .EQ $D9F8 NORM ON...CMD
E6F8- 1420 GETBYTE .EQ $E6F8 EVAL FORMULA
D43C- 1430 RESTART .EQ $D43C ENTER APPLESOFT
03F5- 1440 AMPER .EQ $3F5 AMPERSAND VECTOR
1450 *-----
1460 INITIALIZE.AMPERSAND
1470 *-----
0803- AD F5 03 1480 LDA AMPER TRANSFER OLD &
0806- 8D 53 08 1490 STA OLD.AMPER
0809- AD F6 03 1500 LDA AMPER+1
080C- 8D 54 08 1510 STA OLD.AMPER+1
080F- AD F7 03 1520 LDA AMPER+2
0812- 8D 55 08 1530 STA OLD.AMPER+2
0815- A9 4C 1540 LDA #$4C A JUMP COMMAND
0817- 8D F5 03 1550 STA AMPER
081A- A9 32 1560 LDA #AMPERSAND.CONTROL
081C- 8D F6 03 1570 STA AMPER+1
081F- A9 08 1580 LDA /AMPERSAND.CONTROL
0821- 8D F7 03 1590 STA AMPER+2
0824- A9 42 1600 LDA #END.PROGRAM+1 RESET PTRS
0826- 85 67 1610 STA TXTTAB

```



Hexdump

BEG: 803; END: 944

```

0803- AD F5 03 RD 53 $8F2F
0808- 08 AD F6 03 8D 54 08 AD $C9A9
0810- F7 03 8D 55 08 A9 4C 8D $8404
0818- F5 03 A9 32 8D F6 03 A9 $AB43
0820- 08 8D F7 03 A9 42 85 67 $3B9F
0828- A9 09 85 68 20 4B D6 4C $D0AE
0830- 3C D4 C9 B4 F0 46 C9 AB $01B8
0838- F0 04 C9 B0 D0 15 48 A0 $1687
0840- 01 B1 B8 C9 22 F0 04 68 $8138
0848- 4C 28 D8 68 C9 AB F0 06 $DC45

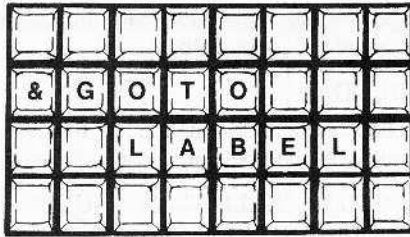
0850- 4C 5F 08 4C FF FF 20 B1 $243A
0858- 00 20 CF 08 4C 5C D9 20 $490A
0860- B1 00 A9 03 20 D6 D3 A5 $A21D
0868- B9 48 A5 B8 48 A5 77 48 $8ACE
0870- A5 76 48 A9 B0 48 20 59 $9048
0878- 08 4C D2 D7 20 B1 00 20 $7B29
0880- F8 E6 48 C9 B0 F0 07 C9 $32B5
0888- AB F0 03 4C C9 DE C6 A1 $A205
0890- D0 12 A0 01 B1 B8 C9 22 $2B55
0898- F0 03 4C FC D9 68 C9 B0 $FA4C

08A0- F0 BD D0 B2 20 B1 00 F0 $6C2F
08A8- 24 C9 22 F0 0B 20 B1 00 $8DCF
08B0- F0 1B C9 2C D0 F7 F0 D6 $85FD
08B8- 20 B1 00 F0 10 C9 22 D0 $ACF3
08C0- F7 20 B1 00 F0 07 C9 2C $4275
08C8- F0 C4 4C C9 DE 68 60 A0 $D23B
08D0- 00 C8 B1 B8 F0 F4 C9 22 $2CAC
08D8- F0 F0 C9 21 90 F3 88 84 $D284
08E0- A0 A5 67 85 9B A5 68 85 $163A
08E8- 9C A0 00 B1 9B 48 C8 B1 $B9B3

08F0- 9B 48 D0 04 68 68 18 60 $40E0
08F8- C8 C8 C8 B1 9B C9 B2 F0 $AC6D
0900- 09 68 85 9C 68 85 9B 4C $32CA
0908- E9 08 C8 B1 9B F0 F2 C9 $9264
0910- 21 90 F7 88 A6 A0 C8 E8 $301E
0918- B1 9B C9 20 90 0F 84 9E $9E4F
0920- 86 9F A4 9F D1 B8 D0 D9 $3418
0928- A4 9E 4C 16 09 8A A8 B1 $7778
0930- B8 C9 22 F0 04 C9 20 B0 $54FA
0938- C8 68 85 9C 68 85 9B 38 $5E86

0940- 60 00 00 00 3F $DD99

```

Target Labels

The target label will always follow a remark statement and should not be enclosed in quotes. The program handles target labels by ignoring all characters until a normal ASCII character is found (punctuation and upper- and lower-case letters). Then all characters, including spaces, are accepted until the end of the line is reached or until a control character is encountered.

Various labels are shown below. The superscripted C signifies a letter as being a control character.

```
10 REM JCJC PRINT NAMEJCJC
```

This line will have a label of "PRINT NAME".

```
10 REM JCJC PRINT JCNAME
```

In this example, the label will be "PRINT" because the J^C cancels the rest of the line.

```
10 REM "PRINT NAME"
```

The last label will not be acknowledged since it is enclosed in quotes.

Active Memory

GOTO Label has another feature transparent to the user — any ampersand program in memory when GOTO Label is



INTRODUCING

The Bit Copier's Friend....

TRACK INSPECTOR

- A replacement cover for all Apple Disk II drives.
- Window allows visual inspection of the booting process.
- Locate tracks (& half tracks) to be copied - FAST.
- Locate bad track copies - FAST.
- Adjust drive speed WITHOUT removing drive cover.
- Instructions for easy labeling of tracks inside drive.

--- Track Inspector ---

\$ 24.95 (plus UPS)

CT MFG., INC.
224 EAST BROAD STREET
GADSDEN, ALABAMA 35903
(205) 547-2717

Mastercharge, Visa or check.



```
0803- C9 B4 1580
0805- D0 03 1590
0807- 4C 49 08 1600
080A- 48 1610 .1
080B- A0 01 1620
080D- B1 B8 1630
080F- C9 22 1640
0811- F0 04 1650
0813- 68 1660
0814- 4C 28 D8 1670
0817- 68 1690
0818- C9 AB 1700
081A- F0 07 1710
081C- C9 B0 1720
081E- F0 0C 1730
0820- 4C C9 DE 1740 .1
1750
1760
1770
1780
1790
1800
1810
0823- 20 B1 00 1820
1830
0826- 20 9C 08 1840
0829- 4C 5C D9 1850
1860
1870
1880
1890
1900
1910
082C- 20 B1 00 1920
1930
082F- A9 03 1940
0831- 20 D6 D3 1950
0834- A5 B9 1960
0836- 48 1970
0837- A5 B8 1980
0839- 48 1990
083A- A5 77 2000
083C- 48 2010
083D- A5 76 2020
083F- 48 2030
0840- A9 B0 2040
0842- 48 2050
0843- 20 26 08 2060
0846- 4C D2 D7 2070
2080
2090
2100
2110
2120
2130
0849- 20 B1 00 2140
084C- 20 F8 E6 2150
```

```
CMP #ON.T IS IT THE ON... STATEMENT?
BNE .1
JMP ON YES SO EXECUTE
PHA SAVE TOKEN
LDY #S01 LOOK FOR QUOTE AFTER THE STATEMENT
LDA (TXTPTR),Y
CMP #QUOTE
BEQ DO.SPECIAL.COMMAND
PHA RESTORE STACK
JME DO.NORMAL DO A NORMAL COMMAND
DO.SPECIAL.COMMAND
PLA RESTORE OLD TOKEN
CMP #GOTO.T IS IT A GOTO?
BEQ GOTO
CMP #GOSUB.T IS IT A GOSUB?
BEQ GOSUB
JMP SYNERR

-----
* HANDLES GOTO, GOSUB AND ON
* GOTO/GOSUBS TO A LABEL OR LINE
* NUMBER, LINKED THRU THE & KEY.
-----

GOTO JSR CHRGET POINT TO THE QUOTE
GOTO2 JSR SEARCH FIND LINE
JMP GO2+3

-----
* THIS ROUTINE HANDLES THE
* GOSUB LABEL COMMAND.
-----

GOSUB JSR CHRGET POINT TO THE QUOTE
GOSUB2 LDA #S3
JSR CHKMEM
LDA TXTPTR+1 PUSH THE TXTPTR ONTO THE STACK
PHA
LDA TXTPTR
PHA
LDA CURLIN+1 PUSH CURRENT LINE ONTO THE STACK
PHA
LDA #GOSUB.T SAVE THE COMMAND ONTO THE STACK
PHA
JSR GOTO2 LOOK FOR THE LINE
JMP NEWST EXECUTE A NEW STATEMENT

-----
* ON GOTO,GOSUB FOR LABEL.
-----

ON JSR CHRGET POINT TO THE FORMULA
JSR GETBYTE EVAL. FORMULA
```

BRUN is left active (as long as it does not use the same memory space). This allows RENUMBER from the Apple master disk to reside in memory and remain accessible to the user at the same time as GOTO Label. To have both programs in memory at once, RUN RENUMBER, then BRUN GOTO LABEL. All the features of RENUMBER will still be active.

Replacing Labels with Line Numbers

The Replace program is GOTO Label's counterpart. It will scan through an Applesoft program to replace the ampersand GOTOs, GOSUBs and ONs with normal GOTOs, GOSUBs and ONs, and will replace the source labels with line numbers. This allows the Applesoft program to be run on any Apple (GOTO Label need not be in memory, since the Applesoft program is standard), or even compiled.

To enter Replace in memory, follow these steps:

- 1) Boot the 3.3 master disk.
- 2) Insert the blank disk in the drive.
- 3) Clear the Applesoft program from memory.
FP
- 4) Enter the monitor.
CALL -151
- 5) Type the Hex dump for REPLACE.
- 6) Return to BASIC.
3D0G
- 7) Save the program.
BSAVE REPLACE,A\$803,L\$360

Replace will overwrite GOTO Label. Because there is no need to have both in memory at the same time, this presents no major problem. Replace will also destroy any Applesoft program currently in memory, so the program to be converted must be saved to the disk. Replace should be BRUN.

Program

```

084F- 4B      2160      PHA          SAVE COMMAND TYPE
0850- C9 B0   2170      CMP #GOSUB.T IS IT A GOSUB
0852- F0 07   2180      BEQ .1
0854- C9 AB   2190      CMP #GOTO.T  IS IT A GOTO
0856- F0 03   2200      BEQ .1
0858- 4C C9 DE 2210      JMP SYNERR  NEITHER,SO PRINT SYNTAX ERROR
                                2220
085B- C6 A1   2230      .1      DEC FAC+4   DECREMENT ON LOOP
085D- D0 12   2240      BNE .3
085F- A0 01   2250      LDY #S01
0861- B1 B8   2260      LDA (TXTPTR),Y IS IT A QUOTE?
0863- C9 22   2270      CMP #QUOTE
0865- F0 03   2280      BEQ .2
0867- 4C FC D9 2290      JMP ONCNT+4 PROCESS COMMAND NORMALLY
                                2300
086A- 68      2310      .2      PLA          GET COMMAND
086B- C9 B0   2320      CMP #GOSUB.T IS IT A GOSUB COMMAND
086D- F0 BD   2330      BEQ GOSUB
086F- D0 B2   2340      BNE GOTO    MUST BE A GOTO THEN
                                2350
0871- 20 B1 00 2360      .3      JSR CHRGET  GET CHARACTER
0874- F0 24   2370      BEQ END.LINE IF END THEN EXIT
0876- C9 22   2380      CMP #QUOTE  IS IT A QUOTE (ENDING)
0878- F0 0B   2390      BEQ .10    USE SEPERATE QUOTE COUNT ROUTINE
087A- 20 B1 00 2400      .20     JSR CHRGET  GET NEXT CHARACTER
087D- F0 1B   2410      BEQ END.LINE
087F- C9 2C   2420      CMP #COMMA  REACH THE END YET?
0881- D0 F7   2430      BNE .20
0883- F0 D6   2440      BEQ .1
0885- 20 B1 00 2450      .10     JSR CHRGET  GET NEXT CHARACTER
0888- F0 10   2460      BEQ END.LINE
088A- C9 22   2470      CMP #QUOTE  END OF QUOTE?
088C- D0 F7   2480      BNE .10
088E- 20 B1 00 2490      JSR CHRGET  MOVE TO NEXT POSITION IN THE LINE
0891- F0 07   2500      BEQ END.LINE
0893- C9 2C   2510      CMP #COMMA  IS IT A COMMA?
0895- F0 C4   2520      BEQ .1      CONTINUE WITH ON COMMAND
                                2530
                                2540 SYNTAX.ERROR
0897- 4C C9 DE 2550      JMP SYNERR  OTHERWISE IT'S A SYNTAX ERROR
                                2560
                                2570 END.LINE
089A- 68      2580      PLA          RESTORE STACK
089B- 60      2590      RTS          AND EXIT
                                2600
                                2610 *-----*
                                2620 * SEARCH FOR THE LINE REQUESTED.
                                2630 *-----*
                                2640
                                2650 SEARCH
089C- A0 00   2660      LDY #S00    SCAN FOR A LEGAL LABEL
                                2670 .0
089E- C8      2680      INY
089F- B1 B8   2690      LDA (TXTPTR),Y
08A1- F0 F4   2700      BEQ SYNTAX.ERROR BAD BRANCH,PRINT SYNTAX ERROR
08A3- C9 22   2710      CMP #QUOTE  FOUND ANOTHE QUOTE?
08A5- F0 F0   2720      BEQ SYNTAX.ERROR

```

Hexdump

Replace

```

0803- A9 5B 85 67 A9      $F3A5
0808- 0B 85 68 20 4B D6 A9 4C $E8F9
0810- 8D F5 03 A9 20 8D F6 03 $2050
0818- A9 08 8D F7 03 4C 3C D4 $2CD6
0820- A5 67 85 B8 A5 68 85 B9 $D413
0828- A5 AF E9 03 85 AF B0 02 $73BA
0830- C6 AF 20 DA 09 A5 B8 85 $CBB1
0838- 75 A5 B9 85 76 20 DA 09 $A8F9
0840- 20 E7 09 B0 03 4C BF 09 $BAA5
0848- A0 00 B1 B8 C9 AF F0 24 $3532

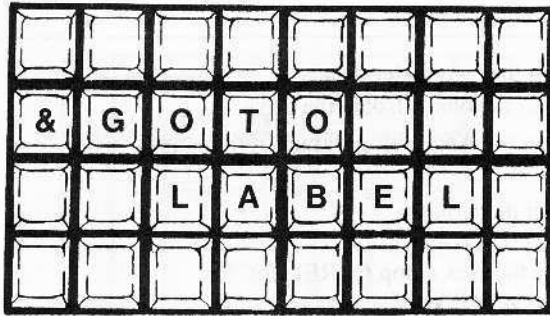
0850- C9 AD D0 11 C8 B1 B8 F0 $907D
0858- 0C C9 C4 D0 F7 C8 20 98 $7262
0860- D9 A0 00 F0 E5 20 A3 D9 $D2D1
0868- B1 B8 08 C8 20 98 D9 28 $6041
0870- D0 D6 F0 BE C8 B1 B8 C9 $BAEF
0878- B0 F0 08 C9 AB F0 04 C9 $8146
0880- B4 D0 E2 88 8C 58 0B A0 $6FEF
0888- 01 A2 02 B1 B8 C9 22 F0 $47D1
0890- 07 88 91 B8 C8 C8 D0 F3 $79C0
0898- CA D0 F6 88 A9 20 91 B8 $C192

08A0- AC 58 0B 20 F1 09 90 12 $7FFB
08A8- 20 A3 D9 A0 00 B1 B8 C8 $0193
08B0- C9 2C F0 06 20 98 D9 4C $4988
08B8- 32 08 8C 58 0B A5 B8 85 $5B93
08C0- 77 A5 B9 85 78 A5 67 85 $BF70
08C8- B8 A5 68 85 B9 20 DA 09 $53C1
08D0- 20 DA 09 20 E7 09 B0 03 $6C1B
08D8- 4C E5 0A A0 00 B1 B8 C9 $2EC7
08E0- B2 F0 0A 20 A6 D9 C8 20 $EC6B
08E8- 98 D9 4C CD 08 C8 B1 B8 $58B6

08F0- D0 03 4C 29 09 C9 21 90 $211F
08F8- F4 8C 57 0B AC 58 0B C8 $B820
0900- B1 77 F0 25 C9 21 90 F7 $9E06
0908- 8C 56 0B AC 56 0B B1 77 $D4A0
0910- F0 36 C9 22 F0 32 C9 20 $6E2A
0918- 90 2E AC 57 0B D1 B8 D0 $23C9
0920- 08 EE 56 0B EE 57 0B D0 $367F
0928- E2 C9 20 D0 11 B1 B8 F0 $5F1E
0930- 17 C9 20 90 13 D0 03 C8 $7F03
0938- D0 F3 C9 22 F0 0A 20 A6 $3442

0940- D9 C8 20 98 D9 4C CD 08 $8019
0948- AC 57 0B B1 B8 F0 0F C9 $09AC
0950- 22 F0 0B C9 20 90 07 F0 $1F00
0958- 02 B0 CE C8 D0 ED 20 A6 $EADD
0960- D9 C8 20 98 D9 20 DA 09 $2A4F
0968- 20 E7 09 B0 03 4C 01 0B $B94C
0970- A0 02 B1 B8 C9 B2 F0 E6 $718F
0978- A0 00 B1 B8 8D 53 0B C8 $93A3
0980- B1 B8 8D 54 0B 20 91 0A $828C
0988- A5 77 85 B8 A5 78 85 B9 $5AC9

```



Conversion

After Replace is BRUN, the labels can be converted to line numbers. Load the Applesoft program to be converted, press the ampersand (&) key, and wait a few seconds. When the Applesoft prompt appears, the conversion is complete. The ampersands have been removed and the program has become standard Applesoft.

The conversion program will always point the branch to the next non-REM statement after the label. In this way no branches to a remark are made.

There are two error messages which can be encountered during the conversion process.

1) LABEL NOT FOUND IN LINE X

This means that there is a branch to a label that does not exist.

2) NO LINE AFTER LABEL CALLED FROM LINE X

This appears if there was no non-REM statement after the label.

Both of these errors cause the conversion process to halt, leaving the remaining portion of the program unconverted. These errors must be corrected before a conversion can be attempted again.

NOTE: Always save the original Applesoft program (the one with all the labels in it), since there is no way the labels can be restored after a conversion is made.

Program

```

08A7- C9 21      2730      CMP #LABEL.T IS IT A LEGAL LABEL?
08A9- 90 F3      2740      BCC .0      NO SO CONTINUE
08AB- 88         2750      DEY         BACK UP ONE
08AC- 84 A0      2760      STY FIRST.OFFSET
                2770
08AE- A5 67      2780      LDA TXTTAB  GET BEGINNING OF THE PROGRAM
08B0- 85 9B      2790      STA LOWTR
08B2- A5 68      2800      LDA TXTTAB+1
08B4- 85 9C      2810      STA LOWTR+1
                2820
08B6- A0 00      2830      LDY #$00
08B8- B1 9B      2840      LDA (LOWTR),Y GET OFFSET TO NEXT LINE
08BA- 48         2850      PHA
08BB- C8         2860      INY
08BC- B1 9B      2870      LDA (LOWTR),Y
08BE- 48         2880      PHA
08BF- D0 04      2890      BNE .6
08C1- 68         2900      PLA
08C2- 68         2910      PLA
08C3- 18         2920      CLC
08C4- 60         2930      RTS
08C5- C8         2940      INY          SKIP LINE#
08C6- C8         2950      INY
08C7- C8         2960      INY
08C8- B1 9B      2970      LDA (LOWTR),Y
08CA- C9 B2      2980      CMP #REM.T  IS IT A REM (MAYBE A LABEL)
08CC- F0 09      2990      BEQ .3
08CE- 68         3000      PLA
08CF- 85 9C      3010      STA LOWTR+1 GOTO NEXT LINE
08D1- 68         3020      PLA
08D2- 85 9B      3030      STA LOWTR
08D4- 4C B6 08   3040      JMP .1      ALWAYS
                3050
08D7- C8         3060      INY
08D8- B1 9B      3070      LDA (LOWTR),Y GET NEXT CHARACTER IN LINE
08DA- F0 F2      3080      BEQ .2
08DC- C9 21      3090      CMP #LABEL.T IS IT A LEGAL LABEL?
08DE- 90 F7      3100      BCC .3
08E0- 88         3110      DEY         SET Y-REG TO PT ONE BEHIND LEG LAB
08E1- A6 A0      3120      LDY FIRST.OFFSET OFFSET TO GOTO/GOSUB/ON LABEL
                3130
08E3- C8         3140      INY
08E4- E8         3150      INX
08E5- B1 9B      3160      LDA (LOWTR),Y COMPARE LABEL
08E7- C9 20      3170      CMP #SPACE.T LOOK FOR SPACE OR CONTROL CHAR.
08E9- 90 0F      3180      BCC .5      IF FOUND THEN LABEL IS DONE
08EB- 84 9E      3190      STY FAC+1
08ED- 86 9F      3200      STX FAC+2      TRANSFER X-REG TO Y-REG
08EF- A4 9F      3210      LDY FAC+2
08F1- D1 B8      3220      CMP (TXTPTR),Y
08F3- D0 D9      3230      BNE .2
08F5- A4 9E      3240      LDY FAC+1      RESTORE Y-REG
08F7- 4C E3 08   3250      JMP .4      CONTINUE WITH COMPARISON
                3260
08FA- 8A         3270      TXA         TRANSFER X-REG TO Y-REG
08FB- A8         3280      TAY
08FC- B1 B8      3290      LDA (TXTPTR),Y MAKE SURE GOTO LABEL IS FINISHED
08FE- C9 22      3300      CMP #QUOTE
0900- F0 04      3310      BEQ .7
0902- C9 20      3320      CMP #SPACE.T

```

Hexdump Replace

```

0990- A0 FF 20 F1 09 AE 50 02 $E03B
0998- BD 51 02 91 B8 CA F0 15 $12B9
09A0- C8 B1 B8 F0 04 C9 22 D0 $A11F
09A8- EF 20 98 D9 8A A8 20 51 $5746
09B0- 0A A0 00 F0 E3 C8 20 98 $47FB
09B8- D9 20 02 0A 4C AB 08 A0 $6742
09C0- 04 A9 00 91 AF 88 10 FB $2130
09C8- A5 B0 85 6A A5 AF 18 69 $D3E9
09D0- 04 85 69 90 02 E6 6A 4C $B697
09D8- F2 D4 E6 B8 D0 02 E6 B9 $8278

09E0- E6 B8 D0 02 E6 B9 60 A5 $54A5
09E8- AF 38 E5 B8 A5 B0 E5 B9 $C8F1
09F0- 60 C8 B1 B8 F0 0A C9 3A $C6C9
09F8- F0 06 C9 22 D0 F3 18 60 $0305
0A00- 38 60 A0 FF C8 B1 B8 F0 $59A4
0A08- 09 C9 3A F0 05 C9 22 D0 $E318
0A10- F3 C8 8C 55 0B A2 03 A5 $FEE2
0A18- B8 8D 28 0A 8D 2B 0A A5 $8201
0A20- B9 8D 29 0A 8D 2C 0A B9 $1D55
0A28- FF FF 8D FF FF D0 10 CA $451B

0A30- D0 0F 38 A5 AF ED 55 0B $8CB5
0A38- 85 AF B0 02 C6 B0 60 A2 $C885
0A40- 03 EE 28 0A EE 2B 0A D0 $D0F3
0A48- DE EE 29 0A EE 2C 0A D0 $F92C
0A50- D6 A5 AF 8D 62 0A 8D 65 $0B77
0A58- 0A A5 B0 8D 63 0A 8D 66 $C1B3
0A60- 0A AD FF FF 99 FF FF AD $56EA
0A68- 62 0A D0 06 CE 63 0A CE $0012
0A70- 66 0A CE 62 0A CE 65 0A $5D5F
0A78- AD 62 0A C5 B8 B0 E2 AD $4BDB

0A80- 63 0A C5 B9 D0 DB 98 18 $5695
0A88- 65 AF 85 AF 90 02 E6 B0 $8148
0A90- 60 A9 00 A8 8D 51 02 8D $63C1
0A98- 50 02 A9 00 8D 59 0B 8D $C09B
0AA0- 5A 0B A2 10 18 2E 53 0B $FD0B
0AA8- 2E 54 0B 2E 59 0B 2E 5A $DB80
0AB0- 0B 38 AD 59 0B E9 0A A8 $32F5
0AB8- AD 5A 0B E9 00 0A 06 C8 $2D72
0AC0- 59 0B 8D 5A 0B CA D0 DD $0A46
0AC8- 2E 53 0B 2E 54 0B AD 59 $FFB8

0AD0- 0B 09 30 EE 50 02 AC 50 $43D6
0AD8- 02 99 51 02 AD 53 0B 0D $2EFF
0AE0- 54 0B D0 B6 60 A0 1A B9 $DB79
0AE8- 0F 0B 20 ED FD 88 10 F7 $75E3
0AF0- A0 00 B1 75 A4 C8 B1 75 $509B
0AF8- 20 24 ED 20 8E FD 4C BF $8B9F
0B00- 09 A0 28 B9 2A 0B 20 ED $478F
0B08- FD 88 10 F7 4C F0 0A A0 $4DA7
0B10- C5 CE C9 CC A0 CE C9 A0 $EB30
0B18- C4 CE D5 CF C6 A0 D4 CF $D8DE

```


How GOTO Label Works

GOTO Label makes use of the normal Applesoft branches. A description of how the normal GOSUB, GOTO and ON work will help explain how the program operates. Then these branches can be compared to the special ampersand branches used in GOTO Label.

GOSUB (\$D921 to \$D93B)

First makes certain there is enough room on the stack to execute another GOSUB.

- a) If there isn't, generates an "out of memory" error.
- b) If there is, saves the position in this line and the token for GOSUB. Enters step 1 of the GOTO routine.

GOTO (\$D93E to \$D968)

1) Converts the tokenized line number to a two-byte hexadecimal number.

2) Finds out if the line number being branched to is greater than 256 plus the current line number.

- a) If so, starts the search from this line number.
- b) If not, starts the search from the beginning of the program.

3) Searches for the line number. If it is found, moves the pointers there and executes the line, otherwise give an ?UNDEF'D STATEMENT ERROR.

ON (\$D9EC to \$DA0B)

1) Evaluates the formula following the ON command.

2) Saves the type of branch (GOTO/GOSUB) on the stack. If the branch is not a GOTO or GOSUB, generates a SYNTAX ERROR.

Program

```

093A- 85 9C   3340   STA LOWTR+1
093C- 68     3350   PLA
093D- 85 9B   3360   STA LOWTR
093F- 38     3370   SEC           SET FOR BORROW
0940- 60     3380   RTS           RTS TO CALLER
           3390   END PROGRAM
0941- 00 00 00 3400   .HS 000000
    
```

Hexdump QUICK COPY

```

1200- 4C 6B 12 4C A8 12 4C C5 $35FA
1208- 12 4C 24 13 4C 2F 13 4C $6981
1210- 50 12 00 60 01 00 00 00 $797D
1218- 23 12 00 14 00 00 00 00 $3CE7
1220- 00 60 01 00 01 EF D8 A9 $9E52
1228- 0C 85 24 A9 08 20 5B FB $6F41
1230- AE 1E 12 B0 3A 13 20 ED $4143
1238- FD AD 16 12 20 DA FD A9 $02CD
1240- 2E 20 ED FD AD 17 12 20 $2E08
1248- DA FD 2C 83 C0 2C 83 C0 $7428
    
```

```

1250- A9 00 8D 1F 12 A9 12 A0 $5A67
1258- 12 20 B5 B7 08 2C 81 C0 $DEA6
1260- A9 00 85 48 28 B0 03 BD $6CEF
1268- 1F 12 60 A9 00 8D 1A 12 $00B4
1270- 8D 1B 12 85 00 8D 15 12 $F80A
1278- A9 22 85 01 85 03 A9 0F $DF5F
1280- 85 04 85 02 A9 B7 8D 14 $36CF
1288- 13 8D 83 C0 8D 83 C0 A2 $D41E
1290- 0F BD 3E 13 9D F0 FF DD $F8A4
1298- F0 FF D0 08 CA 10 F2 A9 $E4AB
    
```

```

12A0- FF 8D 14 13 8D 81 C0 60 $04E8
12A8- A9 01 8D 1E 12 A5 01 8D $5C96
12B0- 16 12 A5 02 8D 17 12 20 $F3D9
12B8- E9 12 AD 17 12 85 02 AD $40B3
12C0- 16 12 85 01 60 A9 02 8D $3301
12C8- 1E 12 A5 03 8D 16 12 A5 $FD36
12D0- 04 8D 17 12 20 E9 12 AD $806E
12D8- 16 12 85 03 AD 17 12 85 $1F5B
12E0- 04 8D 01 60 A9 FF 85 00 $B637
12E8- 60 A9 14 8D 1B 12 20 27 $1AA0
    
```

```

12F0- 12 9D 09 68 8D 4E 13 68 $2478
12F8- 8D 4F 13 60 CE 17 12 10 $E3CC
1300- 0C A9 0F 8D 17 12 CE 16 $52C2
1308- 12 10 02 38 60 EE 1B 12 $A300
1310- AD 1B 12 C9 FF F0 08 C9 $D1A7
1318- B7 D0 D3 A9 D0 8D 1B 12 $580B
1320- D0 CC 18 60 AD 4F 13 48 $20AE
1328- AD 4E 13 48 4C EE 12 AD $4304
1330- 4F 13 48 AD 4E 13 48 4C $1639
1338- FC 12 53 52 57 49 83 7F $631D
    
```

```

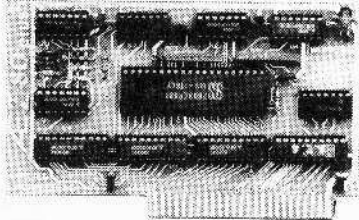
1340- 5C CC B5 FC 17 17 F5 03 $6880
1348- FB 03 59 FF 86 FA 00 00 $0605
    
```

Replace

```

0B20- CE A0 CC C5 C2 C1 CC 8D $1BC0
0B28- 8D 87 A0 C5 CE C9 CC A0 $ADB0
0B30- CD CF D2 C6 A0 C4 C5 CC $D170
0B38- CC C1 C3 8D AE CC C5 C2 $7D91
0B40- C1 CC A0 D2 C5 D4 C6 C1 $931A
0B48- A0 C5 CE C9 CC A0 CF CE $C140
0B50- 8D 8D 87 00 00 00 00 $77E6
0B58- 00 00 00 00 00 00 00 $106B
    
```

Z-80 PLUS!



- **Totally** compatible with all CP/M software, there are no exceptions!
- Fully compatible with Microsoft disks, most other Z-80 cards are not.
- Does everything the other Z-80 cards do plus supports Z-80 interrupts.
- Runs on any Apple IIe, II or II+ with at least 48K memory, works with any 80 column card but does not require one.
- A special "CP/M Detector" ROM eliminates many I.C.'s for the lowest power consuming Z-80 card available. (We use the Z-80A at a fast 3.58 MHZ).
- The Z-80 Plus runs: dBase II, WordStar, Spell Star, Fortran-80, Peachtree and all CP/M based software.
- Complete documentation included .
- High quality P.C. board, gold plated connector, all I.C.'s in high quality sockets, with mil. spec. components used throughout.
- Two year warranty.

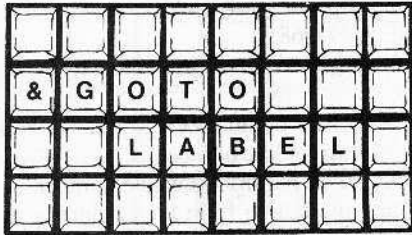
Price \$139.00

Add \$10.00 If Outside U.S.A.

Send Check or Money Order to:

APPLIED ENGINEERING
P.O. Box 470301, Dallas, TX 75247
All Orders Shipped Same Day

To Order By Phone Call (214) 492-2027
7am to 11pm 7 days a week
MasterCard & Visa Welcome
Texas Residents Add 5% Sales Tax



GOTO Label Ampersand Commands

Compare the GOTO Label commands to the normal Applesoft branches.

&GOSUB

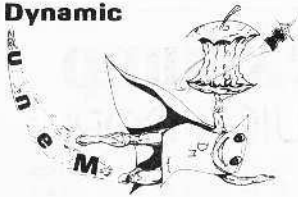
- 1) Duplicates step 1 from Applesoft.
- 2) Falls into &GOTO.

&GOTO

- 1) Searches for a label. Is there a quote following the &GOTO?
 - a) If no quote is found, there can't be a label, so the routine treats it as a normal GOTO and lets Applesoft handle it.
 - b) If a quote is found, a label was present. Falls into step 2.

- 3) Enters a loop to find the proper statement to CALL.
 - a) Decrements the results from the calculation made in step 1.
 - b) If at zero, exits to step 4.
 - c) Skips to the next line number. If there are no more line numbers, restores the stack and continues with the next statement.
 - d) Loops back to step a.
- 4) Retrieves branch type and perform the branch specified.

Dynamic



Checksums

10	-	\$E1BE
20	-	\$5F0C
30	-	\$B4FA
40	-	\$C19F
50	-	\$1484
60	-	\$26C5
70	-	\$4AE3
80	-	\$F571
90	-	\$C402
100	-	\$676E
110	-	\$A891
120	-	\$7C77
130	-	\$F1FE
140	-	\$0C4E
150	-	\$FC1E
160	-	\$B4F9
170	-	\$A74E
180	-	\$B055
190	-	\$376D
200	-	\$6FAF
210	-	\$7BDD
220	-	\$BF20
230	-	\$7265
240	-	\$1331
250	-	\$376D
260	-	\$C1E3
270	-	\$EFC0
280	-	\$7CF0
290	-	\$BEC4
300	-	\$21D0
310	-	\$C13A
320	-	\$0B81
330	-	\$8205
340	-	\$59E8
350	-	\$BC7D
360	-	\$ABEF
370	-	\$D6F7
380	-	\$E600
390	-	\$5E1A
400	-	\$E6DC

Program Replace

```

1000
1010 *-----
1020 *      REPLACE GOTO LABEL
1030 *
1040 *
1050 *      BY
1060 *      ROBB S. CANFIELD
1070 *
1080 *-----
1090
1100 *-----
1110 * ROUTINE TO SEARCH FOR
1120 * AMPERSANDS AND GOTO, GOSUB, ON
1130 *-----
1140
1150      .OR $803      PLACE BELOW BASIC PROGRAM
1160      .TF REPLACE
1170
1180
1190
0067- 1200 AP.START .EQ $67,68  POINTS TO WHERE APPLESOFT STARTS
0075- 1210 CURLIN  .EQ $75,76   START OF CURRENT LINE
00AF- 1220 AP.END   .EQ $AF,B0   POINTS TO END OF BASIC PROGRAM
0069- 1230 LOMEM   .EQ $69       POINTER TO END+1
00B8- 1240 CURRENT.LOC .EQ $B8,B9  POINTER TO CURRENT LOCATION
0077- 1250 TEMP    .EQ $77,78    POINTER TO SEARCH LOCATION
1260
0250- 1270 LENGTH  .EQ $250      LENGTH OF LINE IN THE BUFFER
0251- 1280 BUFFER  .EQ $251      BUFFER TO STORE LINE NUMBER
03F5- 1290 AMPER.VECTOR .EQ $3F5   THE AMPERSAND JUMP VECTOR
1300
1310
1320 *-----
1330 * TOKENS FOR APP. COMMANDS
1340 *-----
1350
00AF- 1360 AMPER   .EQ $AF
00AD- 1370 IF     .EQ $AD
00C4- 1380 THEN   .EQ $C4
00B0- 1390 GOSUB  .EQ $B0
00AB- 1400 GOTO   .EQ $AB
00B2- 1410 REM    .EQ $B2
00B4- 1420 ON     .EQ $B4
0020- 1430 SPACE  .EQ $20      THE TOKEN FOR A SPACE
002C- 1440 COMMA  .EQ $2C
1450

```

2) Searches from the beginning of the program for a matching label.

- a) If one is found, executes that line.
- b) If not, generates an error.

&ON

- 1) Duplicates step 1 from the normal ON command.
- 2) Duplicates step 2 from the normal ON command.
- 3) Enters a loop to find the correct statement to execute.
 - a) Decrements the calculation made in step 1.
 - b) If at zero, goes to step 4.
 - c) Skips to the next line number/label. If at the end of the &ON command, executes the next statement.
 - d) Loops back to step a.
- 4) If the branch is to a line number, lets Applesoft handle it.

Otherwise goes to the proper ampersand routine.

The GOTO Label ampersand commands nearly duplicate the corresponding Applesoft commands, except that instead of searching for a line number, the routines search for a label. After directing the pointer to the number of the line in which the label was found, the process switches to the appropriate Applesoft routine.

GOTO Label will make your programs more user-friendly. Instead of having to remember that a particular routine started at line 10500, you can simply GOTO the name of that routine. This is especially useful when dealing with programs of epic length, which may contain enough routines to bewilder the most organized user. Program documentation will obviously become a less-tedious chore.

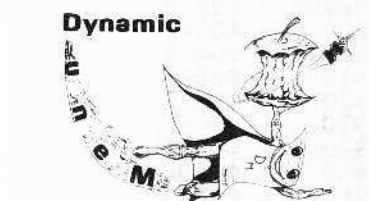


Program Replace

```

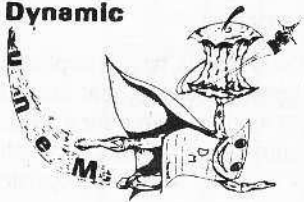
1460
1470 *-----
1480 * SUBROUTINES USED
1490 *-----
1500
D43C- 1510 RESTART .EQ $D43C   ENTER APPLESOFT
D64B- 1520 SCRATCH .EQ $D64B  EXECUTE A NEW COMMAND
D9A3- 1530 DATAN .EQ $D9A3   MOVE ONTO NEXT STATEMENT
D9A6- 1540 REMN .EQ $D9A6    GOTO NEXT LINE
D4F2- 1550 RELOCATE .EQ $D4F2  REBUILD POINTERS SUBROUTINE
ED24- 1560 LINPRT .EQ $ED24   PRINT THE LINE NUMBER
D998- 1570 ADDTXTPTR .EQ $D998  ADD Y-REG TO TXTPTR ($B8,$B9)
FD8E- 1580 COUT .EQ $FD8E    PRINT THE A-REG IN ASCII
FD8E- 1590 CROUT .EQ $FD8E   GENERATES A RETURN
1600
1610
1620 INIT
0803- A9 5B 1630 LDA #END.PROGRAM RESET BASIC PROGRAM POINTERS
0805- 85 67 1640 STA AP.START
0807- A9 0B 1650 LDA /END.PROGRAM
0809- 85 68 1660 STA AP.START+1
080B- 20 4B D6 1670 JSR SCRATCH EXECUTE THE NEW STATEMENT
080E- A9 4C 1680 LDA #$4C THE JUMP COMMAND
0810- 8D F5 03 1690 STA AMPER.VECTOR
0813- A9 20 1700 LDA #START
0815- 8D F6 03 1710 STA AMPER.VECTOR+1
0818- A9 08 1720 LDA /START
081A- 8D F7 03 1730 STA AMPER.VECTOR+2
081D- 4C 3C D4 1740 JMP RESTART AND ENTER APPLESOFT
1750
1780 START
0820- A5 67 1790 LDA AP.START INITIALIZE THE PROGRAM
0822- 85 B8 1800 STA CURRENT.LOC
0824- A5 68 1810 LDA AP.START+1
0826- 85 B9 1820 STA CURRENT.LOC+1
0828- A5 AF 1830 LDA AP.END POINT END OF PROGRAM POINTER TO
1840 * BYTE FOLLOWING LAST LINE
082A- E9 03 1850 SBC #$03
082C- 85 AF 1860 STA AP.END
082E- B0 02 1870 BCS LOOP
0830- C6 AF 1880 DEC AP.END
1890
1900 LOOP
0832- 20 DA 09 1910 JSR SKIP.TWO.BYTES
0835- A5 B8 1920 LDA CURRENT.LOC SAVE CURRENT LINE FOR ERRORS
0837- 85 75 1930 STA CURLIN
0839- A5 B9 1940 LDA CURRENT.LOC+1
083B- 85 76 1950 STA CURLIN+1

```



Checksums

410	-	\$4467
420	-	\$FB58
430	-	\$E874
440	-	\$B454
450	-	\$1CF1
460	-	\$A56E
470	-	\$4E29
480	-	\$7192
490	-	\$336E
500	-	\$9BE4
510	-	\$5A5B
520	-	\$8331
530	-	\$A80F
540	-	\$4FA5
550	-	\$AB74
560	-	\$88CD
570	-	\$B69F
580	-	\$F7A0
590	-	\$BBD2
600	-	\$95B4
610	-	\$F387
620	-	\$0404
630	-	\$5B10
640	-	\$3F8C
650	-	\$F5B9
660	-	\$3351
670	-	\$D1B3
680	-	\$492A
690	-	\$C15C
700	-	\$26F9
710	-	\$830E
720	-	\$D980
730	-	\$97BF
740	-	\$DCD4
750	-	\$6B74
760	-	\$AB78
770	-	\$2F46
780	-	\$3219
790	-	\$9E2F
800	-	\$6CD7

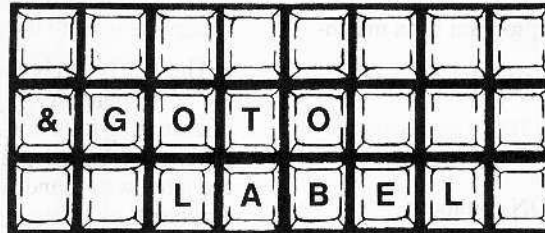


Checksums

- 810 - \$90F5
- 820 - \$1FEC
- 830 - \$A13A
- 840 - \$9A45
- 850 - \$5E95
- 860 - \$8207
- 870 - \$94D2
- 880 - \$232C
- 890 - \$0307
- 900 - \$00E5
- 910 - \$050B
- 920 - \$9E5A
- 930 - \$F0BF
- 940 - \$D84A
- 950 - \$B19F
- 960 - \$67F9
- 970 - \$B4BA
- 980 - \$4E7B
- 990 - \$52AD
- 1000 - \$9C3E

- 1010 - \$58D7
- 1020 - \$16C5
- 1030 - \$C116
- 1040 - \$BE1C
- 1050 - \$46FF
- 1060 - \$ED41
- 1070 - \$7115
- 1080 - \$57D5
- 1090 - \$A2C7
- 1100 - \$1906
- 1110 - \$8E06
- 1120 - \$C2C4
- 1130 - \$8A98
- 1140 - \$C17E
- 1150 - \$6088
- 1160 - \$7BF6
- 1170 - \$1C86
- 1180 - \$31F4
- 1190 - \$FE3E
- 1200 - \$3092

- 1210 - \$86C9
- 1220 - \$06F1
- 1230 - \$0726
- 1240 - \$BEE5
- 1250 - \$24A9
- 1260 - \$B10F
- 1270 - \$EC5E
- 1280 - \$68D1
- 1290 - \$0DB3
- 1300 - \$E8FA
- 1310 - \$0B92
- 1320 - \$D631
- 1330 - \$D8B7
- 1340 - \$8A9F
- 1350 - \$BD4D
- 1360 - \$27A4
- 1370 - \$5C11
- 1380 - \$46C0
- 1390 - \$AD38
- 1400 - \$551B



```

083D- 20 DA 09 1960      JSR SKIP.TWO.BYTES
0840- 20 E7 09 1970      JSR CHECK.END
0843- B0 03      1980      BCS LOOP.2
0845- 4C BF 09 1990      JMP THE.END
0848- A0 00      2000 LOOP.2 LDY #$00
                                2010 LOOP.3
084A- B1 B8      2020      LDA (CURRENT.LOC),Y
084C- C9 AF      2030      CMP #AMPER
084E- F0 24      2040      BEQ AMPER.FOUND
0850- C9 AD      2050      CMP #IF      IS IT AN IF THEN STATEMENT
0852- D0 11      2060      BNE LOOP.1
                                2070 LOOP.4
0854- C8      2080      INY
0855- B1 B8      2090      LDA (CURRENT.LOC),Y
0857- F0 0C      2100      BEQ LOOP.1
0859- C9 C4      2110      CMP #THEN    LOOK FOR A "THEN"
085B- D0 F7      2120      BNE LOOP.4
                                2130 LOOP.5
085D- C8      2140      INY      IS THE NEXT COMMAND AN "&"
085E- 20 98 D9 2150      JSR ADDTXTPTR
0861- A0 00      2160      LDY #$00
0863- F0 E5      2170      BEQ LOOP.3    ...ALWAYS
                                2180 LOOP.1
0865- 20 A3 D9 2190      JSR DATAN    GO TO NEXT LINE
0868- B1 B8      2200      LDA (CURRENT.LOC),Y
086A- 08      2210      PHP      SAVE STATUS (AT END OF A LINE?)
086B- C8      2220      INY
086C- 20 98 D9 2230      JSR ADDTXTPTR
086F- 28      2240      PLP
0870- D0 D6      2250      BNE LOOP.2
0872- F0 BE      2260      BEQ LOOP
                                2270 AMPER.FOUND
0874- C8      2280      INY
0875- B1 B8      2290      LDA (CURRENT.LOC),Y
0877- C9 B0      2300      CMP #GOSUB
0879- F0 08      2310      BEQ REPLACE
087B- C9 AB      2320      CMP #GOTO
087D- F0 04      2330      BEQ REPLACE
087F- C9 B4      2340      CMP #ON
0881- D0 E2      2350      BNE LOOP.1
                                2360
                                2370 REPLACE
0883- 88      2380      DEY
0884- 8C 58 0B 2390      STY TEMP.OFFSET
0887- A0 01      2400      LDY #S1      MOVE CODE AFTER & TILL SECOND QUOTE
0889- A2 02      2410      LDX #S02
088B- B1 B8      2420 .1    LDA (CURRENT.LOC),Y GET A CHARACTER
088D- C9 22      2430      CMP #' '    FOUND QUOTE
088F- F0 07      2440      BEQ .3
                                2450 .2
0891- 88      2460      DEY
0892- 91 B8      2470      STA (CURRENT.LOC),Y
0894- C8      2480      INY
0895- C8      2490      INY
0896- D0 F3      2500      BNE .1      ALWAYS
                                2510 .3
0898- CA      2520      DEX      ON SECONUD QUOTE
0899- D0 F6      2530      BNE .2
089B- B8      2540      DEY      REPLACE LAST LETTER WITH A SPACE
089C- A9 20      2550      LDA #SPACE
089E- 91 B8      2560      STA (CURRENT.LOC),Y
08A0- AC 58 0B 2570      LDY TEMP.OFFSET
08A3- 20 F1 09 2580      JSR SCAN.TO.QUOTE LOOK FOR A QUOTE
08A6- 90 12      2590      BCC REPLACE.2
08A8- 20 A3 D9 2600      JSR DATAN    GOTO THE END OF THE LINE
                                2610
                                2620 NEXT.LINE
08AB- A0 00      2630      LDY #$00
    
```

Program Replace

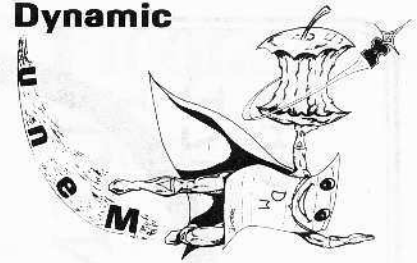
```

08AD- B1 B8 2640 LDA (CURRENT.LOC),Y CHECK FOR A COMMA (ON)
08AF- C8 2650 INY SKIP OVER THE EOL MARKER
08B0- C9 2C 2660 CMP #COMMA
08B2- F0 06 2670 BEQ REPLACE.2
08B4- 20 98 D9 2680 JSR ADDTXTPTR
08B7- 4C 32 08 2690 JMP LOOP CONTINUE THE SEARCH
2710
2720 REPLACE.2
08BA- 8C 58 0B 2730 STY TEMP.OFFSET SAVE THE OFFSET TO THE QUOTE
08BD- A5 B8 2740 LDA CURRENT.LOC SAVE CURRENT LOCATION
08BF- 85 77 2750 STA TEMP
08C1- A5 B9 2760 LDA CURRENT.LOC+1
08C3- 85 78 2770 STA TEMP+1
2780
08C5- A5 67 2790 LDA AP.START SEARCH FOR A LABEL
08C7- 85 B8 2800 STA CURRENT.LOC
08C9- A5 68 2810 LDA AP.START+1
08CB- 85 B9 2820 STA CURRENT.LOC+1
2840
2850 SEARCH.LABEL
08CD- 20 DA 09 2860 JSR SKIP.TWO.BYTES IGNORE NEXT LINE POINTER
08D0- 20 DA 09 2870 JSR SKIP.TWO.BYTES
08D3- 20 E7 09 2880 JSR CHECK.END
08D6- B0 03 2890 BCS .1
08D8- 4C E5 0A 2900 JMP LINE.NUMBER.NOT.FOUND
08DB- A0 00 2910 .1 LDY #00
08DD- B1 B8 2920 LDA (CURRENT.LOC),Y IS IT A REM???
08DF- C9 B2 2930 CMP #REM
08E1- F0 0A 2940 BEQ FOUND.LABEL YES!!!
08E3- 20 A6 D9 2950 JSR REMN NO, SO SKIP TO END OF LINE
08E6- C8 2960 INY
08E7- 20 98 D9 2970 JSR ADDTXTPTR CURRENT.LOC
08EA- 4C CD 08 2980 JMP SEARCH.LABEL
3000
3010 FOUND.LABEL
08ED- C8 3020 .2 INY SKIP THE REM STATEMENT
08EE- B1 B8 3030 LDA (CURRENT.LOC),Y GET A VALUE
08F0- D0 03 3040 BNE .1 AT THE END SO SKIP IT
08F2- 4C 29 09 3050 JMP NO.MATCH ILLEGAL LABEL, SKIP IT
08F5- C9 21 3060 .1 CMP #SPACE+1 IGNORE EVERYTHING =< THAN A SPACE
08F7- 90 F4 3070 BCC .2
08F9- 8C 57 0B 3080 STY OFFSET.2 SAVE THIS OFFSET
08FC- AC 58 0B 3090 LDY TEMP.OFFSET SKIP TO WITHIN THE QUOTES
08FF- C8 3100 .3 INY
0900- B1 77 3110 LDA (TEMP),Y FIND FIRST LEGAL CHARACTER
0902- F0 25 3120 BEQ NO.MATCH
0904- C9 21 3130 CMP #SPACE+1 LOOK FOR A LEGAL LABEL
0906- 90 F7 3140 BCC .3
0908- 8C 56 0B 3150 STY OFFSET.1 SAVE THE SECOUND OFFSET
3160
3170 CHECK.LABELS
090B- AC 56 0B 3180 .1 LDY OFFSET.1
090E- B1 77 3190 LDA (TEMP),Y GET BYTE FROM THE "TO" LOCATION
0910- F0 36 3200 BEQ FINAL.CHECK AT END OF LABEL?
0912- C9 22 3210 CMP #'"'
0914- F0 32 3220 BEQ FINAL.CHECK
0916- C9 20 3230 CMP #SPACE
0918- 90 2E 3240 BCC FINAL.CHECK
091A- AC 57 0B 3250 LDY OFFSET.2
091D- D1 B8 3260 CMP (CURRENT.LOC),Y
091F- D0 08 3270 BNE NO.MATCH
0921- EE 56 0B 3280 INC OFFSET.1
0924- EE 57 0B 3290 INC OFFSET.2
0927- D0 E2 3300 BNE .1
3320
3330 NO.MATCH
0929- C9 20 3340 CMP #SPACE
092B- D0 11 3350 BNE .1

```

continued on page 35

Dynamic



Checksums

```

1410 - $C8D9
1420 - $77D8
1430 - $E2A0
1440 - $0AE6
1450 - $3C33
1460 - $7544
1470 - $5515
1480 - $053D
1490 - $3C44
1500 - $7DCF
1510 - $E033
1520 - $86C3
1530 - $B93F
1540 - $116A
1550 - $CF45
1560 - $F49A
1570 - $679A
1580 - $C75E
1590 - $7BCF
1600 - $9DAA

```

```

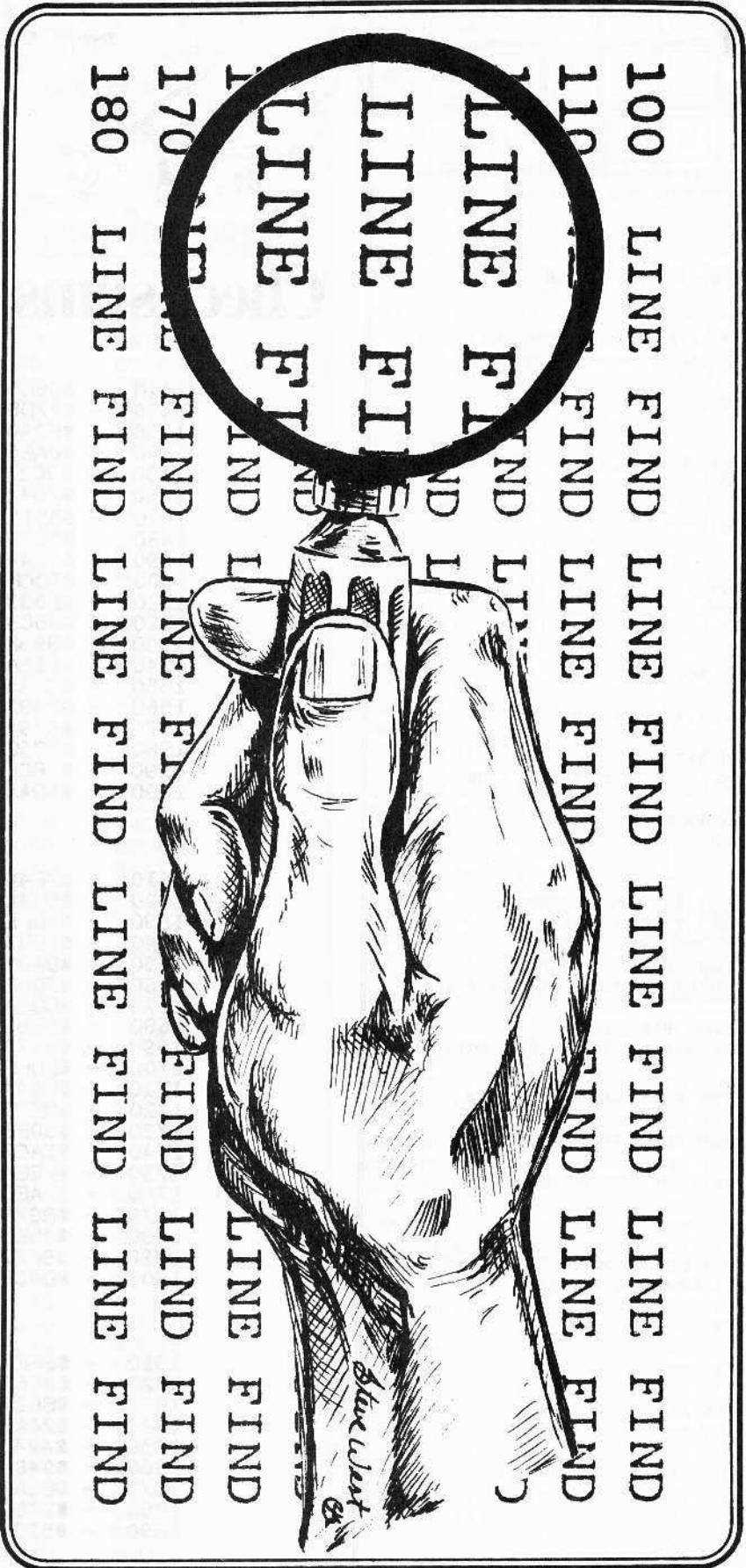
1610 - $7F4C
1620 - $5B4A
1630 - $ADFE
1640 - $3DB3
1650 - $DA05
1660 - $708C
1670 - $DD8A
1680 - $5E82
1690 - $697A
1700 - $E0FF
1710 - $064A
1720 - $0F79
1730 - $50FB
1740 - $1AC7
1750 - $EEEE4
1760 - $7AED
1770 - $B878
1780 - $35E5
1790 - $9C77
1800 - $D4CB

```

```

1810 - $BFE7
1820 - $4C65
1830 - $B637
1840 - $24A6
1850 - $494F
1860 - $9487
1870 - $BCAF
1880 - $2C50
1890 - $57CA

```



By Robb Canfield

Have you ever deleted what you thought was a useless line from your longest program and soon after witnessed the crash of the century? Maybe you were convinced the line was spare baggage, then found that other lines in the program needed to access that line.

Line Find was designed to prevent this problem before a critical line is deleted.

It is an ampersand routine which will reveal whether a particular line number is called by other lines in the program. If the answer is affirmative, Line Find lists which lines would be affected if the line in question were no longer present. If no dependent lines are found, the questionable line can be deleted. Otherwise, the modification should not be made.

Using Line Find

To place Line Find in memory, first type in the program listing. When finished, save Line Find with
BSAVE LINE FIND AS290,LS132

While Line Find is in memory, load the Applesoft program to be examined. When you find a line to delete, or just a line to which you have forgotten the calls, activate Line Find with the command

&L LINE NUMBER

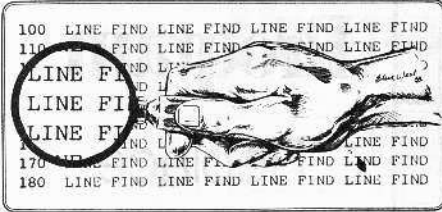
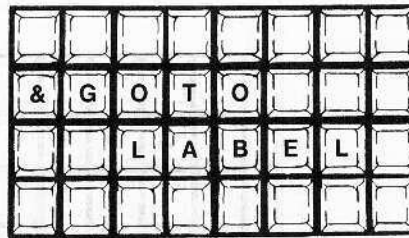
Line Find will search through your program and display in a column the numbers of all lines which use the chosen line number as a point of reference. To see any line, list the desired line number. If there are no calls to the chosen line, "NONE" will be printed.

The Line Find program will leave certain other ampersand features active in memory. For example, Line Find can be in memory at the same time as your favorite renumber program. This will allow you the advantage of using the programs simultaneously.



Program

Replace



Program

```

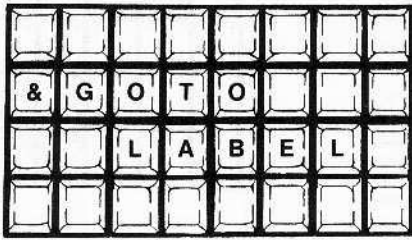
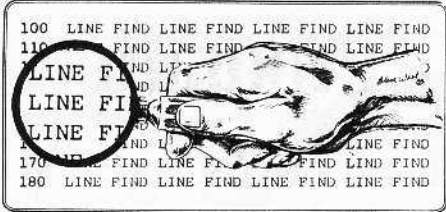
1000      .OR $29D
1010
1020
1030      .TF LINE FIND
1040      LDY #0
1050 NXCHAR LDA STMESS,Y
1060      BMI PRIT
1070      JMP INIT.LINE.SEARCH
1080 PRIT   JSR COUT1
1090      INY
1100      BNE NXCHAR
1110 STMESS .HS 8DBD
1120      .AS --"LINE FIND INSTALLED."
1130      .HS 8D
1140      .AT --"USE &L 'LINENUM' <RET> "
1150 *-----
1160 * FINDS A LINE NUMBER IN AN
1170 * APPLESOFT PROGRAM. USES THE
1180 * AMPERSAND COMMAND WITH AN
1190 * "L" PARAMETER.
1200 *-----
1210
1220 *-----
1230 * ZERO PAGE EQUATES
1240 *-----
1250 TXTPTR .EQ $B8,B9
1260 LOWTR  .EQ $9B,9C
1270 CURLIN .EQ $78,79
1280 AP.START .EQ $67,68
1290 NEXT.OFFSET .EQ $76,77
1300 ADDON  .EQ $D998   ADD Y TO TXTPTR
1310 *-----
1320 * TOKENS BEING USED
1330 *-----
1340 GOTO    .EQ $AB
1350 GOSUB   .EQ $B0
1360 THEN   .EQ $C4
1370 EOL    .EQ $3A
1380 SPACE  .EQ $20
1390
1400
1410 *-----
1420 * APPLESOFT ROUTINES USED
1430 *-----
1440 CHRGET  .EQ $B7
1450 WARM.BASIC .EQ $D43C
1460 REMN    .EQ $D9A6
1470 DATAN   .EQ $D9A3
1480 LINPRT  .EQ $ED24
1490 SYNERR  .EQ $DEC9
1500
1510
1520 *-----
1530 * MONITOR ROUTINES
1540 *-----
1550 CROUT   .EQ $FD8E
1560 AMPER   .EQ $3F5
1570 COUT1   .EQ $FDF0
1580 INIT.LINE.SEARCH

```

```

092D- B1 B8      3360 .2      LDA (CURRENT.LOC),Y
092F- F0 17      3370      BEQ FINAL.CHECK
0931- C9 20      3380      CMP #SPACE
0933- 90 13      3390      BCC FINAL.CHECK
0935- D0 03      3400      BNE .3
0937- C8        3410      INY
0938- D0 F3      3420      BNE .2      ...ALWAYS
093A- C9 22      3430 .3      CMP #' "
093C- F0 0A      3440      BEQ FINAL.CHECK
093E- 20 A6 D9   3450 .1      JSR REMN      POINT TO THE END OF THE LINE
0941- C8        3460      INY          PT TO FIRST BYTE OF NEXT LINE
0942- 20 98 D9   3470      JSR ADDTXTPTR
0945- 4C CD 08   3480      JMP SEARCH.LAB KEEP LOOKING FOR PROPER LAB
                3490
                3510
                3520 FINAL.CHECK
0948- AC 57 0B   3530      LDY OFFSET.2 MAKE SURE THE LABEL HAS ENDED
094B- B1 B8      3540 .0      LDA (CURRENT.LOC),Y
094D- F0 0F      3550      BEQ MATCH
094F- C9 22      3560      CMP #' "'
0951- F0 0B      3570      BEQ MATCH
0953- C9 20      3580      CMP #SPACE
0955- 90 07      3590      BCC MATCH
0957- F0 02      3600      BEQ .1
0959- B0 CE      3610      BCS NO.MATCH NO MATCH, SOCONTINUE SEARCH
095B- C8        3620 .1      INY
095C- D0 ED      3630      BNE .0      ...ALWAYS
                3640
                3650 MATCH
095E- 20 A6 D9   3660      JSR REMN      GO TO THE END OF THE LINE
0961- C8        3670      INY          AND ONE MORE
0962- 20 98 D9   3680      JSR ADDTXTPTR
0965- 20 DA 09   3690      JSR SKIP.TWO.BYTES SKIP THE LINE OFFSET
0968- 20 E7 09   3700      JSR CHECK.END
096B- B0 03      3710      BCS .1
096D- 4C 01 0B   3720      JMP NO.LINE.AFTER.REM
0970- A0 02      3730 .1      LDY #$2      MAKE SURE THIS IS NOT A REM
0972- B1 B8      3740      LDA (CURRENT.LOC),Y
0974- C9 B2      3750      CMP #REM     A REM?
0976- F0 E6      3760      BEQ MATCH   YES SO GOTO NEXT LINE
0978- A0 00      3770      LDY #$0     GET THE LINE NUMBER
097A- B1 B8      3780      LDA (CURRENT.LOC),Y
097C- 8D 53 0B   3790      STA LINE.NUMBER
097F- C8        3800      INY
0980- B1 B8      3810      LDA (CURRENT.LOC),Y
0982- 8D 54 0B   3820      STA LINE.NUMBER+1
0985- 20 91 0A   3830      JSR HEX.ASCII
0988- A5 77      3840      LDA TFMP    MOVE LINE TO CHGE TO CUR LOC
098A- 85 B8      3850      STA CURRENT.LOC
098C- A5 78      3860      LDA TEMP+1
098E- 85 B9      3870      STA CURRENT.LOC+1
                3880
0990- A0 FF      3890      LDY #$FF    SCAN FOR A QUOTE
0992- 20 F1 09   3900      JSR SCAN.TO.QUOTE
                3910
                3920 .3
                3930 *
                3940 *
0995- AE 50 02   3950      LDA #$20    WRITE OVER 1ST QUOTE W SPACE
0998- BD 51 02   3960 .4      STA (CURRENT.LOC),Y
099B- 91 B8      3970      LDX LENGTH LENGTH OF NUM TO SUB FOR LAB
099D- CA        3980      LDA BUFFER,X GET THE VALUE
099E- F0 15      3990      STA (CURRENT.LOC),Y AND SAVE IT
09A0- C8        4000      DEX
09A1- B1 B8      4010      BEQ DONE
09A3- F0 04      4020      INY
09A5- C9 22      4030      LDA (CURRENT.LOC),Y DO WE NEED MORE ROOM?
                4040      BEQ .5
                4050      CMP #' "'

```



Program Replace

```

1590 LDA AMPER+1
1600 STA OLD.AMPER+1
1610 LDA AMPER+2
1620 STA OLD.AMPER+2
1630 LDA #LINE.SEARCH
1640 STA AMPER+1
1650 LDA /LINE.SEARCH
1660 STA AMPER+2
1670 RTS
1680
1690
1700 OLD.AMPER
1710 JMP $FFFF
1720 PRMESS LDY #TBLLN
1730 NEXT LDA TXTBLE-1,Y
1740 JSR COUT1
1750 DEY
1760 BNE NEXT
1770 BEQ WARM
1780 TXTBLE .AS "-"
ENON"
1790
1800
1810 LINE.SEARCH
1820 CMP #'L IS IT A PROPER COMMAND
1830 BNE OLD.AMPER
1840 INIT.SEARCH
1850 LDX TXTPTR TRANSFER TXTPTR TO LOWTR
1860 INX
1870 STX LOWTR LOWTR IS LOC IN PAGE
1880 LDA TXTPTR+1 TWO THAT THE LINE NUMBER
1890 STA LOWTR+1 IS STORED AT.
1900 LDA AP.START
1910 STA NEXT.OFFSET NEXT.OFFSET IS LOC
1920 LDA AP.START+1 IN MEMORY WHERE THE WE
1930 STA NEXT.OFFSET+1 READING FROM PROG
1940
1950
1960 SEARCH.LINE
1970 LDA NEXT.OFFSET
1980 STA TXTPTR
1990 LDA NEXT.OFFSET+1
2000 STA TXTPTR+1
2010 LDY #03 MV INFO FR LINE TO BUFFERS
2020 .1 LDA (TXTPTR),Y
2030 STA NEXT.OFFSET,Y
2040 DEY
2050 BPL .1
2060 ORA NEXT.OFFSET+1 AT END OF PROGRAM?
2070 BNE WHAT
2080 LDA $FF
2090 BEQ PRMESS
2100 .2 LDA #0
2110 STA $FF
2120
2130 PLA PULL OFF RETURN ADDRESS
2140 PLA
2150 WARM JMP WARM.BASIC RETURN TO BASIC
2160
2170
2180 WHAT
2190 LDY #03
2200 SEARCH.LINE.2
2210 INY
2220 LDA (TXTPTR),Y
2230 BEQ END.LINE
2240 CMP #' " IF A QUOTE THEN SKIP TO END
2250 BNE .1

```

```

09A7- D0 EF 4040 BNE .4 NO, JUST CONT REPLACMENT
4050
4060 .5
09A9- 20 98 D9 4070 JSR ADDTXTPTR SET UP FOR THE MOVE
09AC- 8A 4080 TXA TRANSFER X TO Y
09AD- A8 4090 TAY INSERT Y BYTES INTO CODE
09AE- 20 51 0A 4100 JSR INSERT.Y.BYTES
09B1- A0 00 4110 LDY #00
09B3- F0 E3 4120 BEQ .4 ...ALWAYS
4130
4140
4150 DONE
09B5- C8 4160 INY GO PAST THE LINE NUMBER
09B6- 20 98 D9 4170 JSR ADDTXTPTR POINT B8 TO THIS SPOT
09B9- 20 02 0A 4180 JSR DELETE.TO.EOL
09BC- 4C AB 08 4190 JMP NEXT.LINE
4220
4230 *-----
4240 * UPDATE THE END OF THE PROGRAM
4250 * POINTER (RELOCATE USES LOMEM AS
4260 * THE END OF PROGRAM POINTER.
4270 *-----
4300
4310 THE.END
09BF- A0 04 4320 LDY #$4 STORE ZEROS AT
09C1- A9 00 4330 LDA #00 END OF PROG
09C3- 91 AF 4340 .1 STA (AP.END),Y
09C5- 88 4350 DEY
09C6- 10 FB 4360 BPL .1
09C8- A5 B0 4370 LDA AP.END+1
09CA- 85 6A 4380 STA LOMEM+1
09CC- A5 AF 4390 LDA AP.END
09CE- 18 4400 CLC
09CF- 69 04 4410 ADC #$4
09D1- 85 69 4420 STA LOMEM
09D3- 90 02 4430 BCC .2
09D5- E6 6A 4440 INC LOMEM+1
09D7- 4C F2 D4 4450 .2 JMP RELOCATE RESET
4470 LINE OFFSET POINTERS
4480 *
4490 * SKIP THE LINE OFFSET
4500 *
4510
4520 SKIP.TWO.BYTES
09DA- E6 B8 4530 INC CURRENT.LOC
09DC- D0 02 4540 BNE .1
09DE- E6 B9 4550 INC CURRENT.LOC+1
09E0- E6 B8 4560 .1 INC CURRENT.LOC
09E2- D0 02 4570 BNE .2
09E4- E6 B9 4580 INC CURRENT.LOC+1
09E6- 60 4590 .2 RTS
4640
4650 *
4660 * CHECK TO SEE IF PAST END
4670 * CC IF PAST END.
4680 *
4690
4700 CHECK.END
09E7- A5 AF 4710 LDA AP.END
09E9- 38 4720 SEC
09EA- E5 B8 4730 SBC CURRENT.LOC
09EC- A5 B0 4740 LDA AP.END+1
09EE- E5 B9 4750 SBC CURRENT.LOC+1
09F0- 60 4760 RTS
4770

```

Program

```

4790 *-----
4800 * LOOK FOR A QUOTE, THE CARRY
4810 * FLAG IS CLEARED IF ONE IS
4820 * FOUND, SET IF NOT.
4830 *-----
4850
4860 SCAN.TO.QUOTE
09F1- C8 4870 .1 INY LOOK FOR A QUOTE MARK
09F2- B1 B8 4880 LDA (CURRENT.LOC),Y
09F4- F0 0A 4890 BEQ .2
09F6- C9 3A 4900 CMP #' ':'
09F8- F0 06 4910 BEQ .2
09FA- C9 22 4920 CMP #' "'
09FC- D0 F3 4930 BNE .1 CONTINUE THE SEARCH
09FE- 18 4940 CLC NO ERROR
09FF- 60 4950 RTS
4960
0A00- 38 4970 .2 SEC NO QUOTE
0A01- 60 4980 RTS
5020
5030 *-----
5040 * ROUTINE TO DELETE BYTES
5050 * FROM AN APPLESOFT PROGRAM
5060 * ENTER WITH CURRENT.LOC
5070 * CONTAINING THE TO LOCATION
5080 * THE FROM LOCATION IS FOUND
5090 * BY SCANNING TO THE END OF THE
5100 * APPLESOFT LINE.
5110 *-----
5130
5140 DELETE.TO.EOL
0A02- A0 FF 5150 LDY #$FF FIND END OF STMT (: OR 00)
0A04- C8 5160 .1 INY
0A05- B1 B8 5170 LDA (CURRENT.LOC),Y
0A07- F0 09 5180 BEQ DELETE.Y.BYTES
0A09- C9 3A 5190 CMP #' ':'
0A0B- F0 05 5200 BEQ DELETE.Y.BYTES
0A0D- C9 22 5210 CMP #' "'
0A0F- D0 F3 5220 BNE .1
0A11- C8 5230 INY
5240
5250 DELETE.Y.BYTES
0A12- 8C 55 0B 5260 STY NUM.BYTES
0A15- A2 03 5270 LDX #$3 COUNTER FOR ZEROS
0A17- A5 B8 5280 LDA CURRENT.LOC GET TO LOCATION
0A19- 8D 28 0A 5290 STA GET1+1
0A1C- 8D 2B 0A 5300 STA PUT1+1
0A1F- A5 B9 5310 LDA CURRENT.LOC+1
0A21- 8D 29 0A 5320 STA GET1+2
0A24- 8D 2C 0A 5330 STA PUT1+2
5340
5350 GET1
0A27- B9 FF FF 5360 LDA $FFFF,Y GET A VALUE (OFFSET WITH
5370 PUT1 Y-REG)
0A2A- 8D FF FF 5380 STA $FFFF
0A2D- D0 10 5390 BNE .1 NOT A ZERO SO DON'T COUNT
0A2F- CA 5400 DEX COUNT THE ZERO
0A30- D0 0F 5410 BNE .11 NOT DONE, SO CONTINUE
0A32- 38 5420 SEC UPDATE END OF PR POINTER
0A33- A5 AF 5430 LDA AP.END
0A35- ED 55 0B 5440 SBC NUM.BYTES
0A38- 85 AF 5450 STA AP.END
0A3A- B0 02 5460 BCS .2 NO BORROW
0A3C- C6 B0 5470 DEC AP.END+1
0A3E- 60 5480 .2 RTS RETURN TO CALLER
5490
0A3F- A2 03 5500 .1 LDX #$3 RESET COUNTER
0A41- EE 28 0A 5510 .11 INC GET1+1 DO NOT RESET THE COUNTER
0A44- EE 2B 0A 5520 INC PUT1+1
0A47- D0 DE 5530 BNE GET1
0A49- EE 29 0A 5540 INC GET1+2

```

Program

```

2260 .0 INY
2270 LDA (TXTPTR),Y
2280 BEQ END.LINE
2290 CMP #' "
2300 BNE .0
2310 INY
2320 LDA (TXTPTR),Y
2330 .1 CMP #GOTO
2340 BEQ SEARCH
2350 CMP #GOSUB
2360 BEQ SEARCH
2370 CMP #THEN
2380 BNE SEARCH.LINE.2
2390
2400 SEARCH
2410 INY
2420 LDA (TXTPTR),Y
2430 CMP #' "
2440
2450 HERE BEQ SKIP.QUOTE
2460 .21 JSR ADDON
2470 LDY #$FF
2480 .4
2490 INY
2500 LDA (TXTPTR),Y NUMBER FOUND
2510 BEQ .41
2520 CMP #GOTO
2530 BEQ FIXBUG
2540 CMP #GOSUB
2550 BEQ FIXBUG
2560 CMP (LOWTR),Y
2570 BEQ .4
2580 CMP #EOL
2590 BEQ .41
2600 CMP #' , ON A ON STATEMENT?
2610 BNE NO.MATCH NO, SO THERE IS NO MATCH
2620
2630 .41
2640 LDA (LOWTR),Y
2650 BNE NO.MATCH
2660
2670
2680 MATCH
2690 LDX CURLIN
2700 LDA CURLIN+1
2710 JSR LINPRT
2720 LDA #1
2730 STA $FF
2740 JSR CROUT
2750 END.LINE
2760 JSR REMN MOVE TO END OF LINE
2770 JMP SEARCH.LINE
2780
2790 NO.MATCH
2800 LDA (TXTPTR),Y
2810 CMP #' , ARE WE IN A ON STATEMENT
2820 BEQ SEARCH YES SO CONTINUE SEARCH
2830 CMP #' 0 IS IT A DIGIT
2840 BCC .1
2850 CMP #$3A PAST DIGITS
2860 BCS .1
2870 INY LOOK FOR A COMMA
2880 BNE NO.MATCH
2890 .1
2900 JSR DATAN GOTO NEXT STATEMENT
2910 CMP #$00 AT END OF LINE? OR STMENT
2920 BNE JMPTOSEA
2930 JMP SEARCH.LINE
2940 JMPTOSEA JMP SEARCH.LINE.2+1
2950

```


100 LINE FIND LINE FIND LINE FIND LINE
 110 LINE FIND LINE FIND LINE FIND LINE
 LINE FIND LINE FIND LINE FIND LINE
 LINE FIND LINE FIND LINE FIND LINE
 LINE FIND LINE FIND LINE FIND LINE
 170 LINE FIND LINE FIND LINE FIND LINE
 180 LINE FIND LINE FIND LINE FIND LINE

Program

```

2950
2970
2980 SKIP.QUOTE
2990     INY           POINT TO END OF QUOTE
3000     LDA (TXTPTR),Y
3010     BEQ NO.MATCH
3020     CMP #""
3030     BNE SKIP.QUOTE
3040     INY           POINT TO AFTER QUOTE
3050     BNE NO.MATCH ...ALWAYS
3060
3070 TBLLEN .EQ $05
3080 FIXBUG INY
3090     BNE HERE
  
```

Checksums

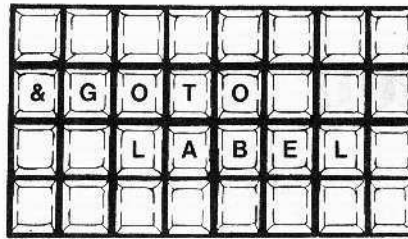
```

029D- A0 00 B9          $5BA1
02A0- AD 02 30 03 4C DE 02 20 $31AD
02A8- F0 FD C8 D0 F2 8D 8D CC $14FE
02B0- C9 CE C5 A0 C6 C9 CE C4 $F5DB
02B8- A0 C9 CE D3 D4 C1 CC CC $014E
02C0- C5 C4 AE 8D D5 D3 C5 A0 $C24A
02C8- A6 CC A0 A7 CC C9 CE C5 $C781
02D0- CE D5 CD A7 A0 BC D2 C5 $EE91
02D8- D4 D5 D2 CE BE 20 AD F6 $2BAE
02E0- 03 8D F6 02 AD F7 03 8D $C3FD

02E8- F7 02 A9 0A 8D F6 03 A9 $64E4
02F0- 03 8D F7 03 60 4C FF FF $04D1
02F8- A0 05 B9 04 03 20 F0 FD $C7D5
0300- 88 D0 F7 F0 3A 8D C5 CE $8020
0308- CF CE C9 4C D0 E7 A6 B8 $AFA2
0310- E8 86 9B A5 B9 85 9C A5 $074F
0318- 67 85 76 A5 68 85 77 A5 $5E07
0320- 76 85 B8 A5 77 85 B9 A0 $3D81
0328- 03 B1 B8 99 76 00 88 10 $1FAE
0330- F8 05 77 D0 0D A5 FF F0 $E92F

0338- BF A9 00 85 FF 68 68 4C $A54D
0340- 3C D4 A0 03 C8 B1 B8 F0 $8170
0348- 53 C9 22 D0 0C C8 B1 B8 $B444
0350- F0 4A C9 22 D0 F7 C8 B1 $586C
0358- B8 C9 AB F0 08 C9 B0 F0 $3BB3
0360- 04 C9 C4 D0 DF C8 B1 B8 $A36D
0368- C9 22 F0 54 20 98 D9 A0 $55CA
0370- FF C8 B1 B8 F0 14 C9 AB $5DA8
0378- F0 52 C9 B0 F0 4E D1 9B $7253
0380- F0 EF C9 3A F0 04 C9 2C $35B7

0388- D0 18 B1 9B D0 14 A6 78 $E7B7
0390- A5 79 20 24 ED A9 01 85 $64A2
0398- FF 20 8E FD 20 A6 D9 4C $6FA3
03A0- 1F 03 B1 B8 C9 2C F0 BD $5AFD
03A8- C9 30 90 07 C9 3A B0 03 $8AFE
03B0- C8 D0 EF 20 A3 D9 C9 00 $D5A7
03B8- D0 03 4C 1F 03 4C 45 03 $D754
03C0- C8 B1 B8 F0 DD C9 22 D0 $E087
03C8- F7 C8 D0 D6 C8 D0 9B 39 $030A
  
```



Program

Replace

```

0A4C- EC 2C 0A 5550     INC PUT1+2
0A4F- D0 D6          5560     BNE GET1     ...ALWAYS
                    5570
                    5590
                    5600 * MAKE MORE ROOM FOR A LINE,
                    5610 * MOVE BYTES FROM CURRENT.LOC
                    5620 * TO CURRENT.LOC+Y, TO MAKE
                    5630 * ROOM FOR MORE INFO.
                    5640 *
                    5670
                    5680 INSERT.Y.BYTES
0A51- A5 AF          5690     LDA AP.END
0A53- 8D 62 0A 5700     STA GET2+1
0A56- 8D 65 0A 5710     STA PUT2+1
0A59- A5 B0          5720     LDA AP.END+1
0A5B- 8D 63 0A 5730     STA GET2+2
0A5E- 8D 66 0A 5740     STA PUT2+2
                    5750
                    5760 GET2
0A61- AD FF FF          5770     LDA $FFFF     GET A BYTE
                    5780 PUT2
0A64- 99 FF FF          5790     STA $FFFF,Y   AND SAVE IT
0A67- AD 62 0A 5800     LDA GET2+1
0A6A- D0 06          5810     BNE .1
0A6C- CE 63 0A 5820     DEC GET2+2
0A6F- CE 66 0A 5830     DEC PUT2+2
                    5840 .1
0A72- CE 62 0A 5850     DEC GET2+1
0A75- CE 65 0A 5860     DEC PUT2+1
0A78- AD 62 0A 5870     LDA GET2+1     DONE?
0A7B- C5 B8          5880     CMP CURRENT.LOC
0A7D- B0 E2          5890     BCS GET2     NO, SO CONTINUE
0A7F- AD 63 0A 5900     LDA GET2+2     MAYBE!?
0A82- C5 B9          5910     CMP CURRENT.LOC+1
0A84- D0 DB          5920     BNE GET2     NO, SO CONTINUE
0A86- 98          5930     TYA           UPDATE END OF
0A87- 18          5940     CLC           PROGRAM COUNTER
0A88- 65 AF          5950     ADC AP.END
0A8A- 85 AF          5960     STA AP.END
0A8C- 90 02          5970     BCC .2
0A8E- E6 B0          5980     INC AP.END+1
0A90- 60          5990 .2     RTS           RETURN TO CALLER
                    6000
                    6040 *
                    6050 *
                    6060 * CONVERSION OF A BINARY NUMBER
                    6070 * TO DECIMAL ASCII.
                    6080 * THE ASCII VALUE IS STORED IN
                    6090 * A REVERSED ORDER
                    6100 *
                    6110 *
                    6120
                    6130
                    6140 HEX.ASCII
0A91- A9 00          6150     LDA #500
0A93- AB          6160     TAY
0A94- 8D 51 02 6170     STA BUFFER
0A97- 8D 50 02 6180     STA LENGTH
                    6190
                    6200 .1
0A9A- A9 00          6210     LDA #500
0A9C- 8D 59 0B 6220     STA MOD10
0A9F- 8D 5A 0B 6230     STA MOD10+1
0AA2- A2 10          6240     LDX #16
0AA4- 18          6250     CLC
                    6270 .2
  
```

Program Replace

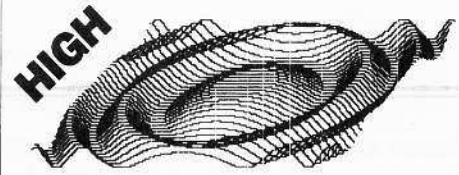
```

OAA5- 2E 53 0B 6280      ROL LINE.NUMBER SHIFT TO DIVND BIT 0
OAA8- 2E 54 0B 6290      ROL LINE.NUMBER+1 WHICH QUOTIENT
OAA8- 2E 59 0B 6300      ROL MOD10  ALSO SHIFT DIVIDEND
OAAE- 2E 5A 0B 6310      ROL MOD10+1
        6320
        6330 *-----
        6340 * A,Y= DIVIDEND-DIVISOR
        6350 *-----
        6360
OAB1- 38                SEC
OAB2- AD 59 0B 6380      LDA MOD10
OAB5- E9 0A 6390        SBC #10
OAB7- A8 6400           TAY          SAVE LOW BYTE IN Y
OAB8- AD 5A 0B 6410      LDA MOD10+1
OABB- E9 00 6420        SBC #0     BRANCH IF DIVIDEND < DIVIS
OABD- 90 06 6430        BCC .3   ELSE
OABF- 8C 59 0B 6440      STY MOD10
OAC2- 8D 5A 0B 6450      STA MOD10+1
        6470 .3
OAC5- CA 6480           DEX
OAC6- D0 DD 6490        BNE .2
OAC8- 2E 53 0B 6500      ROL LINE.NUMBER SHIFT IN CARRY FOR QUOT
OACB- 2E 54 0B 6510      ROL LINE.NUMBER+1
        6530 *
        6540 * CONCATENATE THE NEXT CHARACTER
        6550 *
OACE- AD 59 0B 6570      LDA MOD10
OAD1- 09 30 6580        ORA #S30   CONV 0..9 TO ASCII '0'...'9'
OAD3- EE 50 02 6590      INC LENGTH INCREASE THE LENGTH BY ONE
OAD6- AC 50 02 6600      LDY LENGTH
OAD9- 99 51 02 6610      STA BUFFER,Y SAVE THE ASCII LINE.NUMBER
        6630 *
        6640 * IF LINE.NUMBER <>0 THEN CONTINUE
        6670 LDA LINE.NUMBER
OADF- 0D 54 0B 6680      ORA LINE.NUMBER+1
OAE2- D0 B6 6690        BNE .1     BRANCH IF LINE.NUM
        6700
OAE4- 60 6710           RTS          RETURN TO CALLING PROGRAM
        6750
        6760 *-----
        6770 * ERROR MESSAGES
        6780 *-----
        6810
        6820 LINE.NUMBER,NOT.FOUND
OAE5- A0 1A 6830        LDY #NO.LINE-NO.LABEL-1
OAE7- B9 0F 0B 6840 .1  LDA NO.LABEL,Y
OAEA- 20 ED FD 6850      JSR COUT
OAE8- 88 6860           DEY
OAAE- 10 F7 6870        BPL .1
        6900 PRINT.LINE.NUMBER
OAF0- A0 00 6910        LDY #S00   GET LINE WITH ILL BRANCH
OAF2- B1 75 6920        LDA (CURLIN),Y LOW BYTE
OAF4- AA 6930           TAX
OAF5- C8 6940           INY
OAF6- B1 75 6950        LDA (CURLIN),Y HIGH BYTE
OAF8- 20 24 ED 6960      JSR LINPRT PRINT THE OFFENDING LINE
OAFB- 20 8E FD 6970      JSR CROUT
OAFE- 4C BF 09 6980      JMP THE.END
        7010 NO.LINE.AFTER.REM
OB01- A0 28 7020        LDY #END.MSG-NO.LINE-1
OB03- B9 2A 0B 7030 .1  LDA NO.LINE,Y
OB06- 20 ED FD 7040      JSR COUT
OB09- 88 7050           DEY
OB0A- 10 F7 7060        BPL .1
OB0C- 4C F0 0A 7070      JMP PRINT.LINE.NUMBER
        7110
        7120 NO.LABEL
OB0F- A0 C5 CE
OB12- C9 CC A0
OB15- CE C9 A0
OB18- C4 CE D5
OB1B- CF C6 A0

```

continued on page 48

RESOLUTION



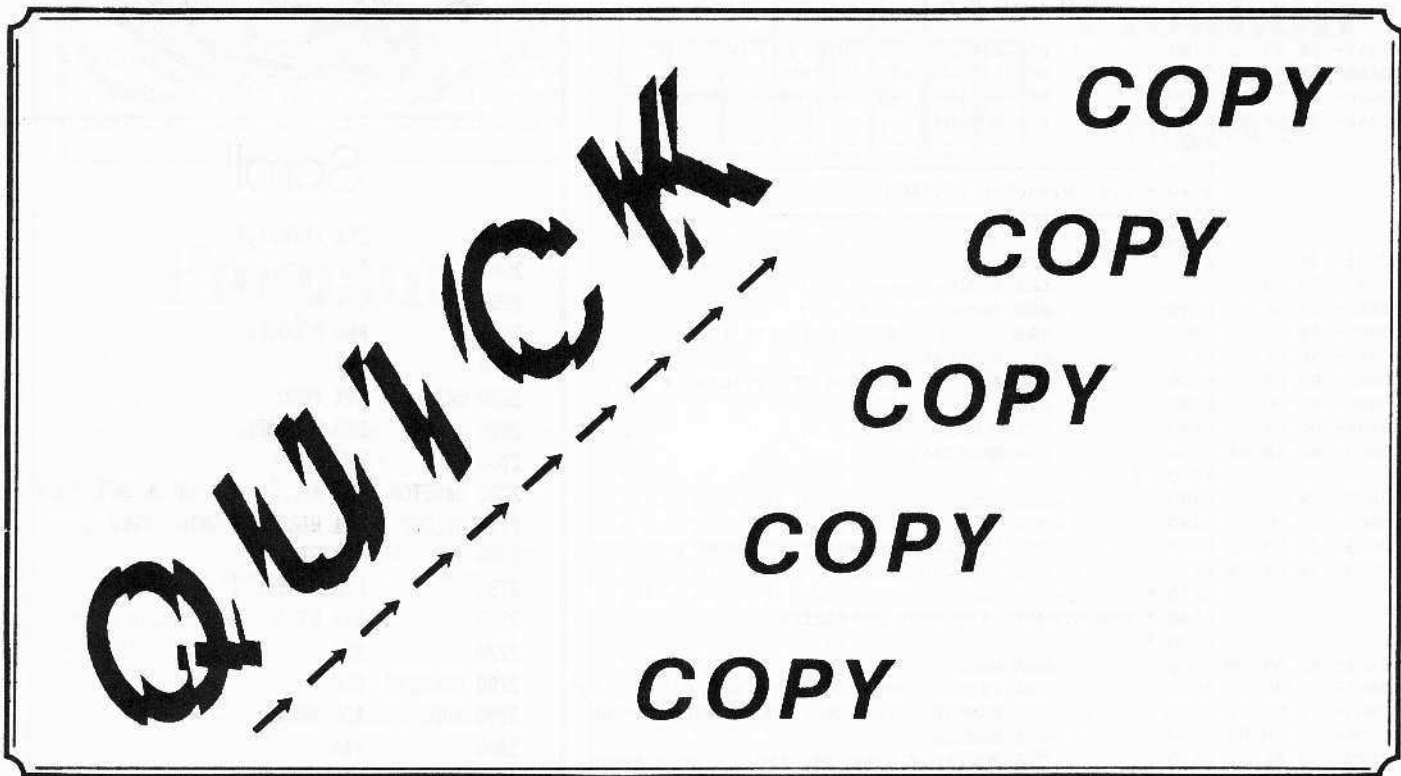
Scroll

```

2630      STA (PTR1),Y
2640      INY
2650      CPY #S28
2660      BNE RLOOP2
2670      INX
2680 OP3   CPX #S00
2690      BNE RLOOP1
2700      RTS
2720 SAVETOP LDX #0   SV TOP OR BOT ? LNS
2730 STLOOP  LDA HIRESH,X ONTO PAGE2
2740      STA PTR1+1
2750      LDA HIRESL,X
2760      STA PTR1
2770      TXA
2780 CCARRY2 CLC
2790 ADD2    ADC #S89
2800      TAX
2810      LDA HIRESH,X
2820      CLC
2830      ADC #S20
2840      STA PTR2+1
2850      LDA HIRESL,X
2860      STA PTR2
2870      TXA
2880 SCARRY2 SEC
2890 SUB2    SBC #S89
2900      TAX
2910      LDY #0
2920 SLP2    LDA (PTR1),Y
2930      STA (PTR2),Y
2940      INY
2950      CPY #S28
2960      BNE SLP2
2970      INX
2980 OP4    CPX #7
2990      BNE STLOOP
3000      RTS
3020 SCRGRHT LDA #0   SCROLL RIGHT
3030      STA OP6+1
3040      LDA #S08   OPCODE FOR 'INX'
3050      STA INY1
3060      STA INY2
3070      STA INY3
3080      LDA #S28
3090      STA OP5+1
3100      LDA #S88   OPCODE FOR 'DEY'
3110      STA DEY

```

continued on page 46



By Robb Canfield

Apple II with 48K
At least one disk drive
Blank, initialized disk

Programmers need not be paralyzed by trepidation each time they use the disk containing the program they spent many weeks to compose or many dollars to buy. If you are among the countless Apple users who worry that valuable original program disks may be hopelessly mangled in day-to-day use, there is a solution to your dilemma.

Acquisition of a simple copy program will enable you to make backups easily and quickly, thus protecting your investment of hours or dollars. Most copy programs will not copy the so-called protected software. The copy program presented is no different, and can only copy disks in standard 3.3 format. But Quick Copy program will copy a disk faster and provide more information about the copy than COPYA.

To Enter the Program

- 1) Boot the 3.3 system master.
- 2) Clear the memory of any Applesoft programs.
FP
- 3) Type and save the BASIC listing, "Copy."
SAVE COPY
- 4) Enter the monitor.
CALL -151

5) Enter and save the hex dump for "Copy.0".

BSAVE COPY.O,A\$1200,L\$150

6) Return to BASIC.

3D0G

To use the program, **RUN COPY.**

"Copy" will load the machine code. This code handles all the reading and writing to the disk; BASIC is only used to handle errors. The default values for source and target drives will appear at the top of the screen. These may need to be changed for your particular system. The program will handle single drive users.

When the choices for the source and target drives have been entered, you will be prompted to enter those disks in the appropriate drives. Single drive users will be prompted to change disks occasionally (only three times if a RAM card is in operation).

As the copy is made, the screen will continuously display the command in operation and the track/sector being affected. The track and sector appear in hex code, while the command will appear as either R(ead) or W(rite.)

Error messages, if there are any, will appear below this display, followed by three choices:

C(ontinue) with the copy. This function will try to copy that sector again.

S(kip) the current sector and continue. This function will skip the bad sector and continue to copy the disk.

E(xit) the program. This will exit you from the copy mode.

Quick Copy consists of two parts. The first part is the controller and is written in BASIC. The second part is the actual routine that reads/writes to the disk, and is written in machine language.

The BASIC program first asks for the source and target drives. Then a call is made to the machine code to initialize all the variables (CALL 4608). This routine also checks for a RAM card and modifies the program if one is in use, allowing more tracks to be read at one time. Then a call is made to the routine that handles all the reading (CALL 4611). BASIC will prompt the user to change disks if necessary, and call the write routine (CALL 4614). This process continues until a copy of the disk is finished.

Entry points to the machine code:

\$1200 (4608) : Initialize variables and check for a RAM card.

\$1203 (4611) : Read some sectors.

\$1206 (4614) : Write some sectors.

\$1209 (4617) : This routine is called when an error occurs and that sector is to be reread.

\$120C (4620) : This routine is called when an error occurs and the sector is to be skipped.

\$120F (4623) : This is the routine that handles both R/W to the disk, a sector at a time.

\$1212 (4626) : The IOB table is stored here.

A copy program is essential to the computer user. It will not eliminate the possibility of garbaging your disks. But it will prevent the destruction of hard-earned programs.

NOTE: Use of Quick Copy will leave the user without DOS, and will require cold booting the system.

Program

```

1000 *-----
1010 * THIS CODE WILL COPY A DISK
1020 * USING THE RAM CARD IF IT IS
1030 * AVAILABLE. WITH A RAM CARD
1040 * A COPY CAN BE MADE IN 3 PASSES
1050 *-----
1060
1070
1080      .OR $1200
1090      .TF COPY_0
1100
1110
1120 FLAG      .EQ $00      FLAG FOR JOB DONE
1130 TAB       .EQ 12      HTAB POS
1140 VTAB      .EQ 11      VTAB POS
1150 CH        .EQ $24
1160 SOURCE.TRACK .EQ $01
1170 SOURCE.SECTOR .EQ $02
1180 TARGET.TRACK .EQ $03
1190 TARGET.SECTOR .EQ $04
1200
1210
1220 BUFFER    .EQ $1400
1230
1240
1250 RWTS     .EQ $B7B5     MUST USE 48 DOS
1260 COUT     .EQ $FD0D     PRINT A CHARACTER
1270 CROUT    .EQ $FD0E     GENERATE A RETURN
1280 PRHEX    .EQ $FDDA     PRINT ACCUM AS A HEX DIGIT
1290 RKEY     .EQ $FD0C     GETS A CHARACTER
1300 TABU     .EQ $FB5B
1310
1320
1330

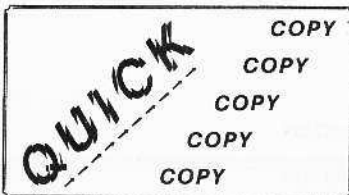
```

Copy.O

```

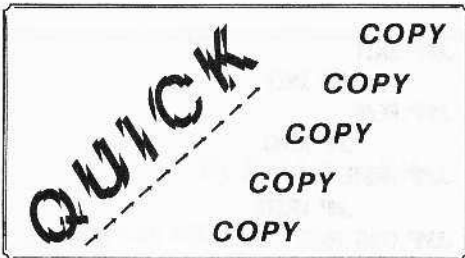
1340 *-----
1350 * JUMP TABLE
1360 *-----
1370
1380 JUMP.INIT
1390      .JP INIT
1400 JUMP.READ
1410      .JP READ
1420 JUMP.WRITE
1430      .JP WRITE
1440 JUMP.CONT.RWTS
1450      .JP CONT.RWTS
1460 JUMP.SKIP
1470      .JP SKIP
1480 JUMP.IOB
1490      .JP IOB2
1500
1510
1520
1530
1540 IOB.TABLE
1550      .HS 00
1560 SLOT      .HS 60
1570 DRIVE     .HS 01
1580 VOLUME    .HS 00
1590 TRACK     .HS 00
1600 SECTOR    .HS 00
1610 DUCT      .DA DEVICE.CHAR.TABLE
1620 DOS.BUFFER .HS 0014
1630 NOT.USED  .HS 0000
1640 COMMAND   .HS 00
1650 ERROR     .HS 00
1660 LAST.VOLUME .HS 00
1670 LAST.SLOT .HS 60

```



Checksums

10	- \$70A8	210	- \$019B	410	- \$584F	610	- \$3456	810	- \$3D7A
20	- \$7131	220	- \$5A23	420	- \$CA76	620	- \$6FB7	820	- \$3103
30	- \$FB29	230	- \$2C7F	430	- \$3FCB	630	- \$2C27	830	- \$0190
40	- \$88E2	240	- \$4D88	440	- \$6A33	640	- \$1050	840	- \$F0C8
50	- \$EDB4	250	- \$63B2	450	- \$34DB	650	- \$939D	850	- \$4FD4
60	- \$B065	260	- \$82FB	460	- \$8F84	660	- \$D1F6	860	- \$590D
70	- \$CD63	270	- \$768A	470	- \$958D	670	- \$63AD	870	- \$F395
80	- \$3005	280	- \$869A	480	- \$0421	680	- \$555B	880	- \$575E
90	- \$DAAA	290	- \$4ABF	490	- \$8709	690	- \$C343	890	- \$8350
100	- \$F8C1	300	- \$6B24	500	- \$B23F	700	- \$FFA1	900	- \$41F1
110	- \$A75F	310	- \$9D7B	510	- \$3ECD	710	- \$4C6D	910	- \$D95F
120	- \$5814	320	- \$3E95	520	- \$01B2	720	- \$0740	920	- \$3D24
130	- \$7A9E	330	- \$8D21	530	- \$B5CB	730	- \$0430	930	- \$44DC
140	- \$148B	340	- \$5D69	540	- \$6C3B	740	- \$15C1	940	- \$6B6F
150	- \$A65F	350	- \$31EA	550	- \$B060	750	- \$8344	950	- \$1889
160	- \$C8D6	360	- \$69C1	560	- \$01A1	760	- \$4684	960	- \$27BB
170	- \$12C4	370	- \$7FB0	570	- \$F1C3	770	- \$9150	970	- \$C764
180	- \$E885	380	- \$A2BA	580	- \$25CC	780	- \$D360	980	- \$0C99
190	- \$B2E3	390	- \$D24F	590	- \$2B50	790	- \$834D	990	- \$6FB8
200	- \$3A9E	400	- \$3C23	600	- \$CBD5	800	- \$E0B9	1000	- \$0100
								1010	- \$4122
								1020	- \$ABA6
								1030	- \$E500



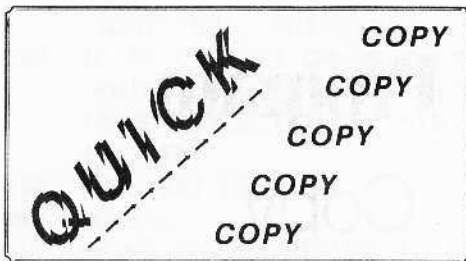
```

1680 LAST.DRIVE .HS #1
1690
1700
1710 DEVICE.CHNR.TABLE
1720 .HS 0001EF08
1730
1740
1750 *-----
1760 * THE IOB
1770 *-----
1780
1790
1800 IOB
1810 LDA #TAB PRINT OPERATION IN PROCESS
1820 STA CH
1830 LDA #UTAB
1840 JSR TABU
1850 LDX COMMAND
1860 LDA CHD,X GET CHARACTER TO PRINT
1870 JSR COUT
1880 LDA TRNCK
1890 JSR PRNEX
1900 LDA #'.
```

Program

```

1910 JSR COUT
1920 LDA SECTOR
1930 JSR PRNEX
1940 BIT %C883 TURN ON THE RAM CARD
1950 BIT %C883
1960 IOB2
1970
1980 LDA #000 CLEAR THE ERROR FLAG
1990 STA ERROR
1990 LDA /IOB.TABLE
2000 LDY #IOB.TABLE
2010 JSR RMTS
2020 PNP
2030 BIT %C881 TURN ON MOTHER ROMS
2040 LDA #000 RESET PROCESSOR STACK
2050 STA %48
2060 FLP
2070 BCS .1 NO ERROR SO LEAVE ERROR CODE INTACT
2080 STA ERROR RESET ERROR FLAG
2090 .1
2100 RTS
2110
2120
2130
2140 *-----
2150 * INITIALIZE THE IOBS
2160 *-----
2170
```



```

2100 INIT
2190 LDA #000 RESET THE BUFFERS
2200 STA DOS.BUFFER
2210 STA DOS.BUFFER+1
2220 STA FLAG RESET FLAG
2230 STA VOLUME RESET THE VOLUME NUMBER
2240 LDA #34 RESET TRACKS TO START AT 34
2250 STA SOURCE.TRACK RESET THE TRACKS
2260 STA TARGET.TRACK
2270 LDA #6F RESET THE SECTOR
2280 STA TARGET.SECTOR
2290 STA SOURCE.SECTOR
2300 LDA #087 SHOW NO RAM CARD IN USE
2310 STA RAM.CARD+1
2320
2330 STA #0803 TURN ON THE RAM CARD
2340 STA #0803
2350 LDX #08F CHECK FOR A RAM CARD
2360 .1
2370 LDA VALUES,X MOVE SOME DATA INTO RAM CARD
2380 STA #FFF0,X
2390 CMP #FFF0,X
2400 BNE .2
2410 DEX
2420 BPL .1
2430 LDA #0FF
2440 STA RAM.CARD+1
2450
2460 .2
2470 STA #0801 TURN OFF THE RAM CARD
2480 RTS RETURN TO CALLER
2490
2500
2510
2520 *-----*
2530 * READ A GROUP OF PAGES.
2540 *-----*
2550
2560 READ
2570 LDA #1 SET COMMAND TO READ
2580 STA COMMAND
2590 LDA SOURCE.TRACK
2600 STA TRACK
2610 LDA SOURCE.SECTOR
2620 STA SECTOR
2630 JSR GO.RWTS READ IN SOME SECTORS
2640 LDA SECTOR
2650 STA SOURCE.SECTOR

```

Program

```

2660 LDA TRACK
2670 STA SOURCE.TRACK
2680 RTS RETURN TO BASIC
2700 *-----*
2710 * WRITE A GROUP OF PAGES
2720 *-----*
2740 WRITE
2750 LDA #2 SET THE WRITE COMMAND
2760 STA COMMAND
2770 LDA TARGET.TRACK
2780 STA TRACK
2790 LDA TARGET.SECTOR
2800 STA SECTOR
2810 JSR GO.RWTS WRITE THE SECTORS TO THE DISK
2820 LDA TRACK
2830 STA TARGET.TRACK
2840 LDA SECTOR
2850 STA TARGET.SECTOR
2860 BCS .1
2870 RTS RETURN TO BASIC
2890
2900 .1
2910 LDA #0FF SHOW JOB DONE
2920 STA FLAG
2930 RTS RETURN TO BASIC
2940
2950
2960 *-----*
3000 * CONTROLLER FOR RWTS. EXITS WITH
3010 * CARRY SET IF ALL DONE.
3020 *-----*
3030
3060 GO.RWTS
3070 LDA /BUFFER
3080 STA DOS.BUFFER+1
3090 CONT.RWTS2
3100 JSR IOB READ/WRITE A SECTOR
3110 BCC SKIP2 NO ERROR SO CONTINUE
3120 PLA GET RETURN ADDRESS
3130 STA LAST
3140 PLA
3150 STA LAST+1
3160 RTS RETURN TO BASIC WITH AN ERROR
3170 SKIP2
3180 DEC SECTOR
3190 BPL .2
3200 LDA #6F RESET SECTOR COUNT
3210 STA SECTOR
3220 DEC TRACK
3230 BPL .2
3240 SEC SHOW THAT WE ARE ALL DONE
3250 RTS RETURN TO CALLER
3260

```


QUICK

COPY
COPY
COPY
COPY
COPY

Program

```
3270 .2
3280     INC DOS.BUFFER+1
3290     LDA DOS.BUFFER+1
3300 RAM.CARD
3310     CMP #8FF     ALL DONE?
3320     BEQ .1
3330     CMP #8B7     USE RAM CARD?
3340     BNE CONT.RMTS2 ...ALWAYS
3350     LDA #8D0
3360     STA DOS.BUFFER+1
3370     BNE CONT.RMTS2 ...ALWAYS
3380 .1
3390     CLC
3390     RTS         RETURN TO CALLER
3400
3410
3420 CONT.RMTS
3430     LDA LAST+1   RESTORE THE STACK
3440     PHA
3450     LDA LAST
3460     PHA
3470     JMP CONT.RMTS2
3480
3490 SKIP
3500     LDA LAST+1
3510     PHA
3520     LDA LAST
3530     PHA
3540     JMP SKIP2
3550
3560 -----
3570 * CONTANTS USED
3580 -----
3590
3600
3610 CHD     .AS "SRMT"
3620 VALUES .MS 837F50CC85FC1717
3630         .MS F503FB0359FFB6FA
3640 LAST    .BS 2     THE RETURN ADDRESS FOR THE STACK
```

Program Copy

```
10 HIMEM: 4607: PRINT CHR$ (4)"
   BLOAD COPY.0"
20 PR# 0: IN# 0
30 M$ = CHR$ (13):E$ = CHR$ (27
   )
40 I = 4626
50 TEXT : NORMAL : HOME
60 SS = PEEK (1528) / 16:TS = SS
   :SD = 1:TD = 2
70 VTAB 2: HTAB 11: PRINT "RAM C
   OPY"
80 VTAB 6: HTAB 9: PRINT "SLOT";
   : HTAB 15: PRINT "DRIVE"
90 PRINT "SOURCE:":; HTAB 11: PRINT
   SS:; HTAB 17: PRINT SD
100 PRINT
110 PRINT "TARGET:":; HTAB 11: PRINT
   TS:; HTAB 17: PRINT TD
120 VTAB 7: HTAB 11: GET A$
130 IF A$ = M$ THEN 160
140 IF A$ < "0" OR A$ > "7" THEN
   PRINT CHR$ (7): GOTO 230
150 SS = VAL (A$)
160 PRINT SS;
170 HTAB 17: GET A$
180 IF A$ = M$ THEN 220
190 IF A$ = E$ THEN 70
200 IF A$ < "1" OR A$ > "2" THEN
   PRINT CHR$ (7):; GOTO 170
210 SD = VAL (A$)
220 PRINT SD
230 VTAB 9: HTAB 11: GET A$
240 IF A$ = M$ THEN 280
250 IF A$ = E$ THEN 70
260 IF A$ < "0" OR A$ > "7" THEN
   PRINT CHR$ (7): GOTO 230
270 TS = VAL (A$)
280 PRINT TS;
290 HTAB 17: GET A$
300 IF A$ = M$ THEN 340
310 IF A$ = E$ THEN 70
320 IF A$ < "1" OR A$ > "2" THEN
   PRINT CHR$ (7):; GOTO 290
330 TD = VAL (A$)
340 PRINT TD
350 VTAB 12
360 INVERSE : VTAB 14
370 PRINT "INSERT SOURCE DISK IN
```

```

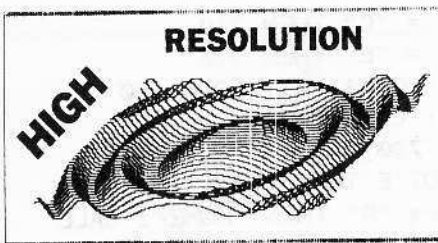
    SLOT:"SS", DRIVE:"SD
390 IF SS < > TS OR SD < > TD THEN
    PRINT : PRINT TAB( 4)"AND
    TARGET DISK IN SLOT:"TS", DR
    IVE:"TD
390 VTAB 18: HTAB 5
400 NORMAL
410 PRINT "PRESS RETURN TO COPY/
    ESC TO EXIT";: GET A$: PRINT
    A$
420 IF A$ < > E$ AND A$ < > M$
    THEN PRINT CHR$ (7): GOTO
    390
430 IF A$ = E$ THEN HOME : GOTO
    70
440 HOME
450 VTAB 2: HTAB 10: PRINT "COPI
    NG DISK"
460 CALL 4600
470 P = 0:ER = P
480 HOME
490 IF P AND SS = TS AND SD = TD
    THEN HOME : VTAB 12: PRINT
    "INSERT SOURCE DISK IN SLOT:
    "SS", DRIVE:"SD";: GET A$: PRINT
    A$
500 POKE I + 1,SS * 16: POKE I +
    2,SD
510 HOME
520 CALL 4611
530 GOSUB 790
540 IF NOT E THEN 580
550 IF A$ = "S" THEN HOME : CALL
    4620: GOTO 530
560 IF A$ = "C" THEN HOME : CALL
    4617: GOTO 530
570 IF A$ = "E" THEN 930
580 IF TS = SS AND TD = SD THEN
    HOME : HTAB 12: PRINT "INSE
    RT TARGET DISK IN SLOT:"TS",
    DRIVE:"TD";: GET A$: PRINT A
    $
590 POKE I + 3, PEEK (I + 14)
600 POKE I + 1,TS * 16: POKE I +
    2,TD
610 IF P > 0 THEN 710
620 HOME : VTAB 12: PRINT "INITI
    ALIZING TARGET DISK"
630 POKE I + 12,4: CALL 4623
640 IF NOT E THEN 710
650 VTAB 12: PRINT CHR$ (7)"INI
    TIALIZATION ERROR"
660 PRINT "TRY AGAIN, COPY DISK,
    EXIT (A,C,E)? ";: GET A$: PRINT
    A$
670 IF A$ = "A" THEN 620

```

```

680 IF A$ = "C" THEN 710
690 IF A$ = "E" THEN 930
700 PRINT CHR$ (7): GOTO 650
710 HOME : CALL 4614
720 GOSUB 790
730 IF NOT E THEN 770
740 IF A$ = "S" THEN HOME : CALL
    4620: GOTO 720
750 IF A$ = "C" THEN HOME : CALL
    4617: GOTO 720
760 IF A$ = "E" THEN 930
770 IF PEEK (0) THEN 980
780 P = P + 1: GOTO 480
790 E = PEEK (I + 13): IF NOT E
    THEN RETURN
800 A$ = "STRANGE ERROR"
810 IF E = 8 THEN A$ = "ERROR DU
    RING INITIALIZATION"
820 IF E = 16 THEN A$ = "DISKETT
    E IS WRITE PROTECTED"
830 IF E = 32 THEN A$ = "VOLUME
    MISMATCH ERROR"
840 IF E = 64 THEN A$ = "I/O ERR
    OR"
850 IF E = 128 THEN A$ = "READ E
    RROR"
860 INVERSE : VTAB 14: HTAB (40 -
    LEN (A$)) / 2: PRINT CHR$
    (7);A$: NORMAL
870 NORMAL
880 VTAB 15:ER = ER + 1
890 PRINT : PRINT TAB( 7)"SKIP/
    CONTINUE/EXIT (SCE)? ";: GET
    A$
900 IF A$ = E$ THEN A$ = "E"
910 IF A$ < > "S" AND A$ < > "
    C" AND A$ < > "E" THEN PRINT
    CHR$ (7): GOTO 880
920 RETURN
930 HOME
940 VTAB 12: PRINT "COPY ABORTED
    "
950 PRINT : PRINT "DO ANOTHER DI
    SK (Y/N)? ";: GET A$
960 IF A$ = "Y" THEN RUN
970 IF A$ < > "N" THEN PRINT CHR$
    (7): GOTO 940
980 HOME
990 PRINT "COPY COMPLETED"
1000 PRINT : PRINT "WITH "ER" ER
    RORS"
1010 VTAB 12: PRINT "ANOTHER COP
    Y (Y/N)? ";: GET A$: PRINT A
    $
1020 IF A$ = "Y" THEN RUN
1030 HOME : PRINT "BYE BYE"

```



Program

Scroll

```

3120      STA DEY2
3130      JSR HORZ
3140      RTS
3160 SCRLEFT LDA #S27      SCROLL LEFT
3170      STA OP6+1
3180      LDA #S88      OPCODE FOR 'DEY'
3190      STA INY1
3200      STA INY2
3210      STA INY3
3220      LDA #SC8      OPCODE FOR 'INY'
3230      STA DEY
3240      STA DEY2
3250      LDA #SFF
3260      STA OP5+1
3280 *
3290 * HORIZONTAL SCROLLING ROUTINE
3300 *
3320 HORZ   LDX #0      START WITH VERT LINE 0
3330 BLOOP  LDA HIRESH,X GET HIBYTE OF VERT LINE
3340      STA PTR1+1
3350      LDA HIRESL,X GET LOBYTE OF VERT LINE
3360      STA PTR1
3370 OP6    LDY #0
3380      LDA (PTR1),Y GET FIRST OR LAST BYTE ON LINE
3390      PHA      STORE IT IN STACK
3400 INY3   INY      MOVE EVERY OTHER BYTE
3410 MLOOP  LDA (PTR1),Y TO THE LEFT OR TO THE RIGHT
3420 DEY    DEY
3430      STA (PTR1),Y
3440 INY1   INY
3450 INY2   INY
3460 OP5    CPY #S28
3470      BNE MLOOP
3480 DEY2   DEY
3490      PLA      RECALL FIRST OR LAST BYTE
3500      STA (PTR1),Y
3510      INX      INC VERT LINE COUNTER
3520      CPX #SC0  FINISHED WITH 192 LINES?
3530      BNE BLOOP IF NOT, DO NEXT LINE
3540      RTS
3560 *
3570 * ROUTINE TO CALCULATE LOBYTES OF
3580 * HI-RES BASE ADDRESSES (PAGE1)
3590 *
3610 CALCLO LDA #0
3620      TAY
3630      STA PTR1
3640 CL1    LDA #4

```

```

3650      STA PTR2
3660 CL2   LDA PTR1
3670 CL3   LDX #8
3680 CL4   STA HIRESL,Y
3690      INY
3700      DEX
3710      BNE CL4
3720      CLC
3730      ADC #S80
3740      BCC CL3
3750      DEC PTR2
3760      BNE CL2
3770      LDA PTR1
3780      ADC #S27
3790      STA PTR1
3800      CMP #S78
3810      BNE CL1
3830 *
3840 * ROUTINE TO CALCULATE HIBYTES OF
3850 * HI-RES BASE ADDRESS (PAGE1)
3860 *
3880      LDA #3
3890      STA GEN1
3900      LDY #0
3910 CH1   LDA #0
3920      STA PTR1
3930 CH2   LDA #2
3940      STA PTR2
3950 CH3   LDA PTR1
3960 CH4   CLC
3970      ADC #S20
3980      STA HIRESH,Y
3990      SEC
4000      SBC #S20
4010      INY
4020      CLC
4030      ADC #4
4040      CMP #S20
4050      BMI CH4
4060      DEC PTR2
4070      BNE CH3
4080      INC PTR1
4090      LDA PTR1
4100      CMP #4
4110      BNE CH2
4120      DEC GEN1
4130      BNE CH1
4140      RTS

```


COREctions

ARCADE QUALITY GRAPHICS

Page 39, 3rd column:

BSAVE QUICKDRAW, A\$800, L\$89
should read

BSAVE QUICKDRAW.OBJ, A\$800, L\$89

Page 40:

The source code which begins on this page is "Quick Draw.Obj."

Page 42:

In the "Commands" box, the directional keys are W,A,X,D rather than W,A,Z,S.

Page 49, 2nd column:

30 PRINT CHR\$(4)"BLOAD
QUICK DRAW"

should read

30 PRINT CHR\$(4)"BLOAD
QUICK DRAW.OBJ"

SCREEN CRUNCHER

Page 22:

In illustration 1, there should be only two double zeros (00), rather than a series of three.

Page 25, 2nd column:

BSAVE UN-PACK, A\$300, L\$
should read

BSAVE UN-PACK, A\$300, L\$7A

HI-RES GRAPHICS

Page 20, 2nd column:

Page Five A\$10000
should read

Page Five A\$A000

Page 21:

The fourth decimal number in the chart on the top half of the page should read 3072 rather than 3027.

SHIMMERING SHAPES

Page 34:

In figure 3,
CMP \$7F should read CMP #\$7F

N0P should read NOP

QD.EDITOR

Page 43:

Change location 33D from 8D to 99.

Page 48

Delete line 2380

SPACE RAID

Page 60.

Add: 1545 PRINT CHR\$(4)"BLOAD
TABLES"

Component Systems Inc.

SPS -II High Performance Switching Power Supply

Introductory Offer
\$95.00



- Apple® Compatible
- + 5 volts DC @ 7 amps output
- +12 volts DC @ 4 amps output
- - 5 volts DC @ 1 amp output
- -12 volts DC @ 1 amp output
- 110 or 220 volts Input Selectable
- 5 volts DC Adjustment
- LED Power on Indicator Light
- AC Power Cord Included
- 2 Year Extended Warranty

Send check or money order for \$95.00* plus \$3.00 shipping and handling to:
Component Systems, Inc., 723 Ninth Avenue, Section A, Kirkland, WA 98033.

For Master Card and Visa orders, include complete card number and expiration date.

C.O.D. orders—add \$2.00. Sorry, no F.P.O., A.P.O. or orders outside U.S.A.

Washington residents please add 8.2% sales tax.

Buy direct—call collect—206-822-6028

*Apple is a registered trademark of Apple® Computer Corporation.

Quantity Discounts

Dealer and OEM Inquiries Invited

We Are Not PIRATES

But We Can Show You How To
UNLOCK LOCKED-UP SOFTWARE

with our complete program listings and step-by-step instructions
YOU CAN, TOO!



IN-DEPTH COVERAGE

of
Special Interest Subjects

- GRAPHICS
- DATA BASES
- UTILITIES
- GAMES

For Hobbyists and Advanced
Computists

Published Quarterly



HARD FACTS
on
APPLE SOFTWARE

An Unprecedented and Controversial disclosure of closely-held

COMPUTER SECRETS

How to: Back-Up Disks
Customize
Commercial Programs
Undo Copy-Protection & More!

Published 8 times yearly

Make checks and money orders payable to:
CORE/HARDCORE Computist
P.O. Box 44549 Tacoma, WA 98444

Annual Subscription Rate: \$25.00 for 8 issues of **HARDCORE COMPUTIST** and 4 issues of **CORE**.
Canada: \$34.00 Mexico: \$37.00
Overseas, foreign countries: \$47.00.

Name: _____ State: _____
Address: _____ Zip: _____
City: _____ Country: _____

Payment must accompany order.
U.S. funds only. No credit cards.

C-2

Program Replace

```

0B1E- D4 CF CE
0B21- A0 CC C5
0B24- C2 C1 CC 7130      .AS -" ENIL NI DNUOF TON LEBAL"
0B27- 8D 8D 87 7140      .HS 8D8D87
                                7150
                                7160 NO.LINE
0B2A- A0 C5 CE
0B2D- C9 CC A0
0B30- CD CF D2
0B33- C6 A0 C4
0B36- C5 CC CC
0B39- C1 C3 7170          .AS -" ENIL MORF DELLAC"
0B3B- 8D 7180            .HS 8D
0B3C- AE CC C5
0B3F- C2 C1 CC
0B42- A0 D2 C5
0B45- D4 C6 C1
0B48- A0 C5 CE
0B4B- C9 CC A0
0B4E- CF CE 7190          .AS -".LEBAL RETFA ENIL ON"
0B50- 8D 8D 87 7200      .HS 8D8D87
                                7210
                                7220 END.MSG
                                7250
                                7260 *-----
                                7270 * VARIABLES USED
                                7280 *-----
                                7290
0B53- 7300 LINE.NUMBER .BS 2  LINE NUMBER APT LABEL (HEX)
0B55- 7310 NUM.BYTES .BS 1    THE NUMBER OF BYTES FOR DEL
0B56- 7320 OFFSET.1 .BS 1    OFFSET FOR THE JUMPS
0B57- 7330 OFFSET.2 .BS 1    OFFSET FOR THE LABEL
0B58- 7340 TEMP.OFFSET .BS 1
0B59- 7350 MOD10 .BS 2
                                7360
                                7370
0B5B- 00 00 7380 END.PROGRAM .HS 0000
    
```

ADVERTISING INDEX

Applied Engineering	29
Collegiate Microcomputer	15
Component Systems	47
CT Manufacturing	26
Interface Age	inside back
Micro Analyst	inside front
SoftKey Publishing	48
VF Associates	11, 22

HURRY!!!

Current **HARDCORE/CORE** readers still can renew your subscriptions at the old, low rates:

- 1 year \$20
- 2 years \$36
- 3 years \$56

You have until October 31, midnight to renew at the old rates, which will then be transformed to, not a pumpkin, but:

- 1 year \$25
- 2 years \$46
- 3 years \$69

The above rates are for our readers in the U.S. Foreign subscription prices will also rise accordingly.

Top decision makers agree...

...the ONLY source they need to keep up with the latest developments in microcomputers is:

MAGAZINE TM

INTERFACE AGE

Look inside IA's pages and decide for yourself!

BOOK REVIEWS

A monthly critique of the newest computer books.

PORTABLES

Need a computer you can tote with you on business trips? IA brings you the latest offerings in this area on a monthly basis.

WORD PROCESSING

Practical suggestions on how to enhance today's word processing systems.

NEW PRODUCTS

Photos and descriptions of the newest products in every category from Business Software to Printers.

SPECIAL REPORT

Timely coverage of the microcomputer industry's most important topics.

PRODUCT HIGHLIGHT

In this monthly feature, our editors give you an in-depth evaluation of the "hottest" product of the month.

APPLICATION

Learn how to apply your new-found knowledge by reading these how-to articles.

HARDWARE & SOFTWARE COMPARISON CHARTS

IA publishes the most extensive, up-to-date, hardware and software comparison tables in the computer industry.

NO-RISK TRIAL SUBSCRIPTION ORDER

12 issues for \$21. That's a 40% savings off the single copy price. If you are not delighted, you may cancel your subscription at any time and receive a prompt refund for all unmailed issues with no questions asked.

MAIL OR CALL TODAY

- One year (12 issues) \$21.00 (U.S. only)
- Two Years (24 issues) \$36.00 (U.S. only)
- Canada One Year \$27.00
- 1 Year Air Delivered Europe only \$42.00
- 1 Year Delivered All Other Countries \$60.00
- 1 Year Surface Mail All Countries \$35.00

SUBSCRIBE TODAY!

Outside of California call toll-free 8 a.m. to 5 p.m.
Pacific Standard Time
Charge subscription with your credit card
1-800-423-6665 Ext. 56
(within California call collect 0-213-926-4862)

Payment must accompany this order



Check enclosed
 must be in U.S. funds
drawn on U.S. bank

Card No. _____ Exp. Date _____

Signature _____

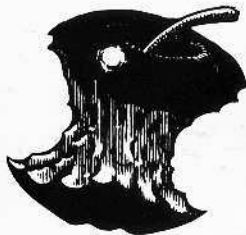
Name _____

Street _____

City _____ State _____ Zip _____

Make check or money order payable to INTERFACE AGE Magazine
P.O. Box 1234, Dept. B056 Cerritos, CA 90701
Allow 6-8 weeks for delivery after receipt of payment

***Don't miss
upcoming issues of***

CORE  **TM**

In time for Christmas,

ALL ABOUT GAMES

***THE LARGEST,
MOST COLORFUL
MAGAZINE OF THE YEAR,***

Featuring an extensive game list,
game evaluations, and
an entirely new
one-player or two-player
computer game.

AND:

DATA BASES

CORE/HARDCORE COMPUTIST
P.O. Box 44549
Tacoma, WA 98444

BULK RATE
U.S. Postage
PAID
Tacoma, WA
Permit No. 269