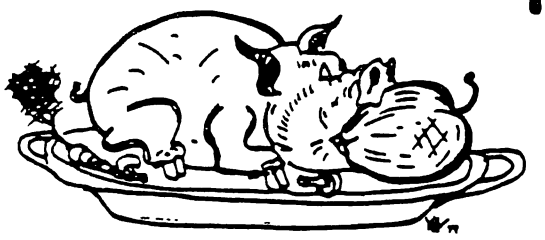


H.A.A.U.G.

HOUSTON AREA APPLE USERS GROUP

THE APPLE BARREL

Price $2.00

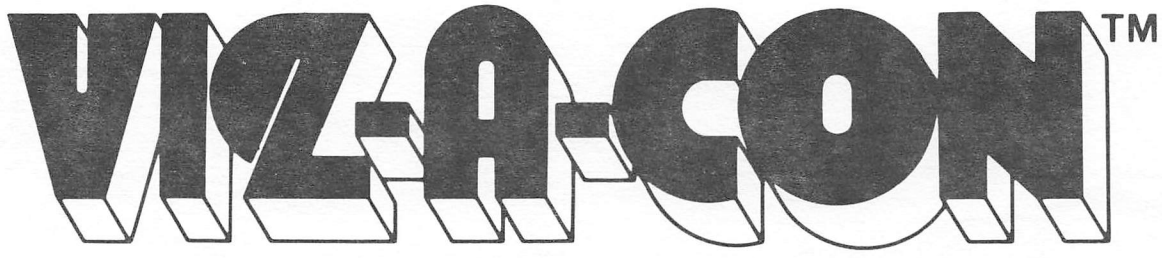
VOLUME 6, NO. 5, 1983
JUNE/JULY 1983

PRESIDENT, Steve Knouse VICE PRES., Clark Johnson
EDITOR, Mike Kramer

Circulation 1150

*** CONTENTS ***

If you're serious about VISICALC®, then you should know about



That's because VIZ-A-CON is the exciting new consolidation system for VisiCalc users. Using your VisiCalc database, and **without learning a new system**, VIZ-A-CON will:

- Perform Consolidations--Automatic roll-ups of weeks into months into years, or departments into divisions into regions.
- Allow "What If" Questions in Three Dimensions--Get answers at any level of consolidation.
- Act as a Report Writer--To your VisiCalc database, with word processor interface.

VIZ-A-CON is another imagination enhancing product brought to you by **ABACUS ASSOCIATES** and is available at better software outlets throughout the Houston area. For the name of the dealer nearest you, please call (713) 666-8146, Dept. 6.

Apple II, II+, IIE, TRS-80 I, III---$ 99.95 + 3.95 S&H
Apple III, TRS-80 II 12/16, IBM PC--$139.95 + 3.95 S&H

HAAUG APPLE BARREL

CLUB NOTES

MEETING SCHEDULE

The HOUSTON AREA APPLE USERS GROUP holds a general business meeting the **second Thursday** of each month in the rear chapel of Memorial Lutheran Church, 5800 Westheimer beginning at 6:30 P.M. A meeting featuring tutorials, access to the HAAUG software library, and special interest group sessions is held beginning at noon the **third Saturday** of each month at the UT School of Public Health in the Med Center at 6905 Bertner at Holcomb.

OFFICERS / EXECUTIVE BOARD

President	Steve Knouse
Vice Pres	Clark Johnson
Treasurer	Ruth Hughes
Secretary	Ruth Dill
Software Lib.	Phil Lauter
Hardcopy Lib.	Robin Cox
Membership	Lee Gilbreath
Local IAC Rep	Robin Cox
Region IAC Rep	Mike Kramer
Editor	Mike Kramer

APPLE HOTLINE
713-668-3102

The APPLE HOTLINE provides an easy means for the general public to learn of meeting topics, news, etc., and can also be used by members to obtain answers to Apple - related questions. Leave your name, member number (see Apple Barrel label), date, and time. You should get a return call within 48 hours.

MEMBERSHIP INFORMATION

New memberships are $30 and include the HAAUG starter kit. Renewals are $20 per year. Make checks payable to Houston Area Apple Users Group and mail to the HAAUG Post Office box, attention Membership Chairman.

CALL FOR ARTICLES

Articles and program listings should be submitted in draft hardcopy form and on disk in Applewriter][or ///, Apple DOS or SOS text, Wordstar, Palantir, or Pascal files, or via modem (358-6687). Files should not contain imbedded escape sequences or control characters and should be printed to disk fill justified if possible. Articles must be free of typing or spelling errors and should be grammatically correct as they cannot be retyped. Diskettes will be returned to the author provided his name and address are on them. Text should be printed 45 columns wide, listings 40 columns wide. Authors of published articles will receive two blank diskettes per printed page as compensation. The Apple Barrel reserves the sole right to choose which articles to use.

APPLE BARREL SCHEDULE

The following schedule will be followed for preparation and mailing of the Apple Barrel.

Ads and articles due by 1st of month
Paste ups to printer by the 5th
Mailed by 20th
Received in Houston by 25th
Received in outlying areas by 1st

APPLE BARREL REPRINT POLICY

ADVERTISING RATES

AD COST = $0.06 * MULTIPLIER * CIRCULATION

Current Circulation = 1150

MULTIPLIERS

Full Page	Half Page	Qrtr Page
1.00	0.60	0.35

Ads should be submitted in camera ready form to H.A.A.U.G. by the 1ST of the month. Charges will be billed and a copy of the Apple Barrel containing the ad will be sent.

EDITOR'S CORNER

This month's issue of the Apple Barrel features a program by Nick Fotheringham, one of our regular contributors, entitled **CIRCUIT ANALYZER**. With this program you should be able to determine whether your electrical circuits are overloaded. What Nick fails to do is tell us how to reduce those big air conditioning bills on the horizon! For the steadily growing number of Apple /// owners is **APPLE /// BASIC COMPARE**, a program which compares two versions of a program and lists the differences on the screen, on a printer, or in a disk file. Clark Johnson is back with another installment of his **Dealing With DOS Column** which covers a free "Fast DOS" patch to DOS 3.3 reprinted from Bob (S-C Assembler) Sander-Cederlof's monthly newsletter Apple Assembly Line. **Apple ///** **Peelings** discusses several items, including how to read the directory from BASIC. A short sample program is included which reads a directory, lists only the text files, permits selection of a file by number, and lists the selected file on the screen. A reprint of an article by Walt Mills of Washington Apple Pi entitled **WHAT IS A 16K RAM CARD AND WHY IS IT IN MY SLOT 0?** should answer a lot of questions about the RAM card. The information presented is equally applicable to the new 64K Apple //e's. Finally an Apple][+ motherboard map and troubleshooting guide extracted from **HOW TO FIX YOUR COMPUTER YOURSELF** by Dick Peschke of Apple-Dayton is reprinted. If you decide to do your own troubleshooting, be sure to turn the power off and discharge any static electricity from your body before touching and component or card.

SIGS

SPECIAL INTEREST GROUPS

Members who share interests are encouraged to join or form Special Interest Groups (SIGs). Although some of these groups meet separately from the regular meetings, most meet at the regular Saturday session at the times listed below. If you would like to become involved in a SIG, show up at the appropriate meeting room at the Saturday session or call the HOTLINE for meeting time and location if the SIG is not listed on the schedule.

HAAUG SATURDAY SESSION SIG ROOM ASSIGNMENTS

TIME	AUDIT	MAIN	RM204	RM208	RM228
NOON	_____	BASIC__	CP/M___	_____	EDUC___
1230	_____	BASIC__	CP/M___	_____	EDUC___
1:00	_____	BASIC__	CP/M___	STOCK__	EDUC___
1:30	NEW MEM	BASIC__	CP/M___	STOCK__	EDUC___
2:00	GEN MTG	_____	_____	_____	_____
2:30	SPECIAL	SOFTWAR	_____	PASCAL_	BUSNESS
3:00	SPECIAL	SOFTWAR	ADVNCED	PASCAL_	BUSNESS
3:30	SPECIAL	SOFTWAR	ADVNCED	PASCAL_	BUSNESS
4:00	_____	SOFTWAR	ADVNCED	GAMES__	APPL///
4:30	_____	SOFTWAR	ASSMBLR	GAMES__	APPL///
5:00	_____	SOFTWAR	ASSMBLR	GAMES__	FORTH__
5:30	_____	SOFTWAR	ASSMBLR	_____	FORTH__

SIG CHAIRMEN: CALL GUS AT 481-5329 THE WEEK BEFORE MEETING TO CONFIRM NEED FOR ROOM OR FOR TEMPORARY ROOM ASSIGNMENT.

SIG CHAIRMEN

Business	Rudge Allen
Pascal	Jon Stevens
Statistics	Lindsay Reed
Education	Brian Whaley
Assembler	Robin Cox
Adv.Topics	Tom Murdock
CP/M	Jim Huck
Sci/Engg	Mike Conway
Stocks	George Marsden
FORTH	Steve Knouse
BASIC	Glenna Payne
Games	Bill Muhlhausen
Apple ///	Mike Kramer

BUSINESS SIG TOPICS

June

SOURCE - description of the system and use of its database.

=== STOCK SPECIAL INTEREST GROUP ===

NEW SIG MEETING PLAN

STOCK/INVESTMENT SIG WILL MEET REGULARLY ON HAAUG SATURDAYS IN ROOM 208 AT ONE P.M.

SPECIAL MEETINGS WILL BE PLANNED FOR 4TH THURSDAY EVENINGS AND WILL BE ANNOUNCED AT THE HAAUG SATURDAY SESSION. WE EXPECT TO HOLD 6 TO 8 EVENING SESSIONS PER YEAR.

ARRANGEMENTS ARE IN PROCESS FOR HOLDING THE EVENING MEETINGS AT THE JUNGMAN LIBRARY JUST WEST OF THE GALLERIA ON WESTHEIMER.

SUPPORTING STORES

The following stores support H.A.A.U.G. by offering discounts to members. Be sure to show your appreciation by patronizing them.

Computer Galleries,
11538 NW Freeway, 956-0900.
2493 S. Braeswood, 661-0055

CTI, 2802 Louisiana, 526-9666.

Micro Solutions, 9949 Harwin #E, 789-5443.

Moore Business Center, 1120 Smith, 237-9063

Softec, Inc., 10875 Katy Freeway, 468-2407

Supertec, FM1960 Bypass at Eastex Freeway, Humble, 446-9770.

If you want your store included, contact the Apple Barrel or call the Hotline.

CIRCUIT ANALYZER

A HOUSEHOLD UTILITY PROGRAM

By
Nick Fotheringham

INTRODUCTION

This program was written for home owners who suffer from overloaded electrical circuits, who are considering adding or modifying circuits, or who are simply curious about the electrical structure of their homes. As written, it will track up to 200 fixtures and appliances attached to up to 24 circuits distributed among up to 4 breaker or fuse boxes. It will plot the locations of the fixtures on each circuit on a hi-resolution floorplan of your house and compare the total amperage attached to each circuit to the breaker/fuse size - displayed in inverse as a warning if the breaker size would be exceeded should all the fixtures on the circuit be turned on at the same time.

The program enables you to store and edit two data bases: one describing the circuits and the other describing the fixtures and appliances. The circuit data base includes the box number, voltage and breaker size (in amps) of each circuit. The fixture data base includes the circuit number, fixture name, name of the room in which it is located, its wattage, its amperage, and coordinates for positioning it on the hi-res floorplan. Since both the wattage and amperage are rarely known for a particular appliance, the program calculates the other if either is provided.

The information that you store in these data bases may be used by the program in the following ways: (1) a list of the fixtures/appliances and their wattages and amperages, sorted by circuit and room, may be printed, (2) the fixtures on an individual circuit may be displayed on the hi-res floorplan, (3) a list of the number of fixtures attached, their total amperage, and the breaker size of each circuit may be displayed, and (4) "what if I move this appliance to that outlet" experiments may be performed by changing the circuit number in the editor. Appliances which are often moved from one circuit to another, such as vacuum cleaners, may be stored on a dummy circuit (e.g. Circuit 24), and then tested on individual circuits to determine if a fuse may blow. You may wish to use the program to identify a relatively unused or stable circuit to which to attach your microcomputer.

PROGRAM STRUCTURE AND OPERATION

The circuit analysis program consists of three components which are located in three separate files. This structure was adopted because the program utilizes the first hi-res screen but is too large to fit in the space below this screen that is normally used to store Applesoft programs. The consequence of storing the program in this space is that when the hi-res graphics routine is called, a portion of the program in obliterated. The solutions to this problem known to me are (1) condense the program (e.g. remove REM's, etc.), (2) store once-used instructions that become non-essential prior to the first use of the hi-res screen at the end of the program (i.e. obliterate lines that have already served their purpose), (3) store the entire program above hi-res screen 1, or (4) split the program and wrap it around this screen.

I have selected the last of these solutions because it illustrates a technique which may be useful in situations where the other three approaches do not solve the problem. This technique was described by Sam Vass in Nibble (Vol. 2, No. 4). The three components of the program produced by this technique are the two 'halves' of the split program and a loading program which places them in the proper locations. The first 'half' of the program consists of a binary file which establishes a pointer linking the location of the Applesoft program (above the hi-res screen) and the location where the Apple expects to find it (below the screen). This file is constructed through the following commands:

```
FP
10 POKE 2070,1:POKE 2071,64
20 GOTO 30
RUN

BSAVE CIRCUIT ANALYZER.BEG,A$800,L$30
```

The loading program is likewise short and is constructed as follows:

```
10 POKE 103,1:POKE 104,64
     POKE 16384,0
20 PRINT CHR$(4)"BLOAD CIRCUIT
     ANALYZER.BEG.A$800"
30 PRINT CHR$(4)"LOAD CIRCUIT
     ANALYZER"
40 END
SAVE LOAD.CIRCUIT.ANALYZER
```

The main program (second 'half') is then entered as shown in the listing. Note that it is important that the first line of this program be line number 30. The program can then be run as follows:

```
RUN LOAD.CIRCUIT.ANALYZER
RUN
```

The main portion of the program uses a hi-res graphics floorplan which I was unable to write specifically for your house. Consequently, if you plan to use this feature, you will need to enter your own routine. The program currently contains a dummy routine on lines 300 to 480 to help you with the format. I recommend that you map out the floorplan on a piece of graph paper and label the coordinates of each of the corners. This will not only make it easier to develop the floorplan routine, but will also make it easier to identify the corresponding coordinates of the fixtures and appliances that you enter later.

SESSION FORMAT

When you run the program, it initially attempts to locate an existing data file named 'CIRCUITS'. If you are running the program for the first time, this file will not exist, and you will be asked to enter data from the keyboard. Descriptions of the circuits will be requested first, and then you will be asked to describe some or all of the fixtures and appliances attached to these circuits. These data will be saved when you exit normally (Option 8 from the menu). This is an important feature to remember when you are adding or editing fixtures. If you wish to save the changes you have made, you must exit normally. On the other hand, if you are experimenting with rearrangements of your appliances, you may wish to avoid saving the changes made during your session by exiting using 'RESET'. Once the existing data file has been read, or you have finished entering data from the keyboard, the menu will be displayed.

Each of the menu options may be selected using a single keystroke (1-8). Although the menu options are generally straightforward, some explanations may be helpful in understanding some of their consequences.

When you opt to edit a fixture or appliance, the names and rooms of the existing items are displayed to help you identify the item to be edited. Since all of the (up to 200) appliances cannot be displayed at the same time, a scrolling routine is used. Ten appliances are displayed simultaneously, and the left and right arrow keys are used to scroll through the longer list. When the appliance to be edited is located (and displayed in inverse), you may proceed to the editor by pressing the 'RETURN' key.

Once in the editing routine, each of the existing characteristics (name, wattage, etc.) of the appliance is displayed, and you are given an opportunity to change it. A simple 'RETURN' is interpreted as a sign that you do not wish to make a change in that characteristic. This enables you to quickly bypass correct items by pressing the 'RETURN' key. However, it also implies that if you wish to delete a characteristic, you must enter a zero or blank (' ').

When you opt to display a circuit on the hi-res floorplan, pressing any key will return you to the menu. When you opt to print or display the fixtures, the list is first sorted by room and then by circuit before being printed.

HINTS

Most fixtures and appliances contain a label which identifies the number of watts or amps used. Unfortunately, this is not universally

true. I easily found a couple of books on home repair and electrical wiring which contained a table of 'typical' values for most major appliances. These values may not be accurate for many modern appliances, but should be better than wild guesses. The program will not accept an 'I don't know' answer of zero for both wattage and amperage. If you must guess, keep in mind that a major objective of the program is to warn you of potential overloads, and thus a guess on the high side is conservative.

POTENTIAL MODIFICATIONS

I gave some thought to adding a fuel consumption rate feature that, when combined with the current cost of electricity, would calculate your estimated electric bill. This feature would then enable you to examine the effect on this bill of taking various appliances out of service. Unfortunately, the energy consumption rates supplied with most appliances, and upon which this program is based, are maximum rates, which assume continuous operation. Many major appliances, such as air conditioners and refrigerators, run intermittently. Individual variation in real consumption rates is likely to be very high. Consequently I abandoned this feature.

If you have a color monitor, you may find it useful to display the locations of fixtures and appliances in color. This can easily be accomplished by changed the 'HCOLOR =' statement in line 1810.

This program was written for residential property owners. As such it will not accept input of values for voltage, wattage and amperage which are unreasonable for residential property. If you wish to use the program for farm or other nonresidential property, you should modify these constraints in the input and editing routines.

```
30  REM  ************************
40  REM  *   CIRCUIT ANALYZER   *
50  REM  *          BY          *
60  REM  *   NICK FOTHERINGHAM  *
70  REM  *      HOUSTON, TX     *
80  REM  ************************
90  TEXT : HOME : NOTRACE : NORMAL
100  GOTO 2290
110  VTAB 23: CALL  - 958: CALL  -
     198: PRINT "PRESS ANY KEY TO
      CONTINUE: ";: GET Z$: RETURN
120  REM  SELECT A FIXTURE/APPLIA
     NCE
130  HOME : FOR M = 0 TO 39: PRINT
     "-";: NEXT : PRINT "USE '<-'
      AND '->' TO LOCATE FIXTURE
     AND <RETURN> TO MAKE A SELEC
     TION.": FOR M = 0 TO 39: PRINT
     "-";: NEXT : PRINT
140  POKE 34,5
150 S = 1:P = 1
160  HOME : FOR I = S TO S + 9
170  IF I = P THEN  HTAB 3: INVERSE
     : PRINT NA$(P);"  ";RM$(P): NORMAL
     : GOTO 190
180  HTAB 3: PRINT NA$(I);"  ";RM
     $(I)
190  NEXT
200  GET A$: IF A$ =  CHR$ (13) THEN
      POKE 34,0: RETURN
210  IF A$ =  CHR$ (8) THEN P = P
      - 1: IF P < 1 THEN  CALL  -
     198:P = P + 1
220  IF A$ =  CHR$ (21) THEN P =
     P + 1: IF P > NF THEN  CALL
      - 198:P = P - 1
230  IF P < S THEN S = S - 1
240  IF P > (S + 9) THEN S = S +
     1
250  GOTO 160
260  RETURN
270  HGR
280  POKE  - 16302,0
290  HCOLOR= 3
```

```
300  HPLOT 60,70 TO 6,70 TO 6,12 TO
     60,12 TO 60,4 TO 180,4: HPLOT
     180,4 TO 180,65 TO 90,65 TO
     90,120 TO 30,120 TO 30,70
310  HPLOT 35,12 TO 35,55 TO 38,5
     5: HPLOT 47,55 TO 50,55 TO 5
     0,12
320  HPLOT 50,65 TO 50,90: HPLOT
     50,98 TO 50,120
330  HPLOT 70,70 TO 90,70 TO 90,1
     5 TO 110,15
340  HPLOT 118,15 TO 150,15
350  HPLOT 158,15 TO 180,15
360  HPLOT 130,15 TO 130,65
490  RETURN
500  REM  MENU
510  TEXT : HOME : FOR I = 0 TO 3
     9: PRINT "=";: NEXT : PRINT
      TAB( 15)"M E N U": FOR I =
     0 TO 39: PRINT "=";: NEXT : PRINT

520  PRINT "    1. ADD A FIXTURE/A
     PPLIANCE"
530  PRINT
540  PRINT "    2. DELETE A FIXTUR
     E/APPLIANCE"
550  PRINT
560  PRINT "    3. EDIT A FIXTURE/
     APPLIANCE"
570  PRINT
580  PRINT "    4. LIST/EDIT CIRCU
     ITS"
590  PRINT
600  PRINT "    5. DISPLAY A CIRCU
     IT"
610  PRINT
620  PRINT "    6. ANALYZE CIRCUIT
     S"
630  PRINT
640  PRINT "    7. PRINT LIST OF F
     IXTURES"
650  PRINT
660  PRINT "    8. END THIS SESSIO
     N"
670  PRINT : FOR I = 0 TO 39: PRINT
     "=";: NEXT : PRINT
680  PRINT "      SELECTION (BY NU
     MBER): ";: GET A$: PRINT A$:
     A =  ASC (A$) - 48: IF A < 1
      OR A > 8 THEN 680
690  IF A <  > 8 THEN 770
700  PRINT DE$"CIRCUITS": PRINT O
     P$"CIRCUITS": PRINT WR$"CIRC
     UITS": PRINT NC
710  FOR I = 0 TO NC - 1: FOR J =
     0 TO 2: PRINT CI(I,J): NEXT
     : NEXT
720  PRINT NF
730  FOR I = 1 TO NF: PRINT NA$(I
     ): PRINT RM$(I)
740  FOR J = 0 TO 4: PRINT FX(I,J
     ): NEXT : NEXT
750  PRINT CL$
760  PRINT : PRINT "GOOD-BYE": END

770  ON A GOTO 790,1020,1120,1430
     ,1780,2130,1930
780  REM  ADD A FIXTURE OR APPLIA
     NCE
790  HOME
800 NF = NF + 1: IF NF > 200 THEN
      PRINT "NO ROOM FOR ANOTHER
     FIXTURE.  CHANGE    PROGRAM
     DIMENSIONS OR DELETE A FIXTU
     RE.": FOR I = 0 TO 2000: NEXT
     : GOTO 510
810  PRINT "YOU ARE NOW ADDING FI
     XTURE OR APPLIANCE NUMBER ";
     NF: PRINT
820  INPUT "  NAME OF FIXTURE: ";
     NA$(NF): PRINT
830  INPUT "  WHICH ROOM IS IT IN
     ? ";RM$(NF): PRINT
840  INPUT "  WHICH CIRCUIT IS IT
      ON? ";A$:A =  VAL (A$): IF
     A < 1 OR A > 24 THEN 840
850 FX(NF,0) = A
860  INPUT "  WHAT IS ITS WATTAGE
     ? ";A$:A =  VAL (A$): IF A <
     0 OR A > 20000 THEN 860
870 FX(NF,1) = A
880  INPUT "  HOW MANY AMPERES? "
     ;A$:A =  VAL (A$)
890  IF A < 0 OR A > 200 THEN 880

900 FX(NF,2) = A
910  IF FX(NF,1) = 0 AND FX(NF,2)
      > 0 THEN FX(NF,1) =  INT (F
     X(NF,2) * CI(FX(NF,0),1))
920  IF FX(NF,1) > 0 AND FX(NF,2)
      = 0 THEN FX(NF,2) =  INT ((
     FX(NF,1) / CI(FX(NF,0),1)) *
     100) / 100
930  IF FX(NF,1) = 0 AND FX(NF,2)
      = 0 THEN 860
940  PRINT : PRINT "COORDINATES O
     N DIAGRAM:"
950  INPUT "  HORIZONTAL = ";A$:A
      =  VAL (A$): IF A < 0 OR A >
     279 THEN 950
960 FX(NF,3) = A
970  INPUT "  VERTICAL = ";A$:A =
      VAL (A$): IF A < 0 OR A > 1
     91 THEN 970
```

```
980 FX(NF,4) = A
990  PRINT : PRINT : PRINT "  ANO
    THER ADDITION? ";: GET A$: PRINT
    A$: IF A$ = "Y" THEN 790
1000  GOTO 510
1010  REM  DELETE A FIXTURE/APPLI
    ANCE
1020  GOSUB 130:I = P: REM  SELEC
    T A FIXTURE
1030  HOME : VTAB 12: INVERSE : PRINT
    NA$(I);" IN ";RM$(I);" DELET
    ED!"
1040  NORMAL
1050  FOR J = I TO (NF - 1)
1060 NA$(J) = NA$(J + 1):RM$(J) =
    RM$(J + 1)
1070  FOR K = 0 TO 4:FX(J,K) = FX
    (J + 1,K): NEXT
1080  NEXT
1090 NF = NF - 1
1100  FOR J = 1 TO 1000: NEXT : GOTO
    510
1110  REM  EDIT A FIXTURE
1120  GOSUB 130:I = P: REM  SELEC
    T A FIXTURE
1130  HOME : FOR J = 0 TO 39: PRINT
    "-";: NEXT : PRINT "EDITING
    ";NA$(I);" IN ";RM$(I): FOR
    J = 0 TO 39: PRINT "-";: NEXT
    : PRINT
1140  PRINT "AS EACH VALUE IS DIS
    PLAYED, ENTER A NEW VALUE TO
     CHANGE OR <RETURN> TO RETAI
    N   THE OLD VALUE:": PRINT
1150  PRINT  TAB( 2)"NAME = ";NA$
    (I)
1160  HTAB 4: INPUT "NEW VALUE =
    ";A$
1170  IF A$ <  > "" THEN NA$(I) =
    A$
1180  PRINT : PRINT  TAB( 2)"ROOM
     = ";RM$(I)
1190  HTAB 4: INPUT "NEW ROOM = "
    ;A$
1200  IF A$ <  > "" THEN RM$(I) =
    A$
1210  PRINT : PRINT  TAB( 2)"CIRC
    UIT = NO. ";FX(I,0);"  NEW C
    IRCUIT = ";: INPUT "";A$
1220  IF  VAL (A$) < 0 OR  VAL (A
    $) > 24 THEN 1210
1230  IF A$ <  > "" THEN FX(I,0) =
     VAL (A$)
1240 FX(I,1) =  INT (FX(I,1))
1250 FX(I,2) =  INT (FX(I,2) * 10
    0) / 100
1260  PRINT : PRINT  TAB( 2)"WATT
    AGE = ";FX(I,1);"  NEW WATTA
    GE = ";: INPUT "";A$
1270  IF  VAL (A$) < 0 OR  VAL (A
    $) > 20000 THEN 1260
1280  IF A$ <  > "" THEN FX(I,1) =
     VAL (A$):FX(I,2) =  INT ((F
    X(I,1) / CI(FX(I,0),1)) * 10
    0) / 100
1290  PRINT : PRINT  TAB( 2)"AMPE
    RAGE = ";FX(I,2);"  NEW AMPE
    RAGE = ";: INPUT "";A$
1300  IF  VAL (A$) < 0 OR  VAL (A
    $) > 200 THEN 1290
1310  IF A$ <  > "" THEN FX(I,2) =
     VAL (A$):FX(I,1) =  INT (FX
    (I,2) * CI(FX(I,0),1))
1320  PRINT : PRINT "COORDINATES:
    ": PRINT
1330  PRINT  TAB( 2)"HORIZONTAL =
     ";FX(I,3);"  NEW VALUE = ";
    : INPUT "";A$
1340  IF  VAL (A$) < 0 OR  VAL (A
    $) > 279 THEN 1330
1350  IF A$ <  > "" THEN FX(I,3) =
     VAL (A$)
1360  PRINT : PRINT  TAB( 2)"VERT
    ICAL = ";FX(I,4);"  NEW VALU
    E = ";: INPUT "";A$
1370  IF  VAL (A$) < 0 OR  VAL (A
    $) > 191 THEN 1360
1380  IF A$ <  > "" THEN FX(I,4) =
     VAL (A$)
1390  PRINT : PRINT "ARE THESE VA
    LUES CORRECT (Y/N)? ";: GET
    A$: PRINT A$
1400  IF A$ <  > "Y" THEN 1130
1410  GOTO 510
1420  REM  EDIT CIRCUITS
1430 MB = 0: FOR I = 0 TO NC - 1:
     IF CI(I,0) > MB THEN MB = C
    I(I,0)
1440  NEXT
1450  FOR I = 1 TO MB
1460  HOME : FOR J = 0 TO 39: PRINT
    "-";: NEXT : PRINT  TAB( 17)
    "BOX #"I: FOR J = 0 TO 39: PRINT
    "-";: NEXT : PRINT
1470  PRINT  TAB( 2)"CIRCUIT   VO
    LTAGE   BREAKER/FUSE SIZE": PRINT
     TAB( 2)"-------   -------
     ----------------"
1480  FOR K = 0 TO NC - 1: IF CI(
    K,0) <  > I THEN 1500
1490  PRINT  TAB( 5)K + 1; TAB( 1
    5)CI(K,1); TAB( 25)CI(K,2);"
      AMPS"
```

```
1500  NEXT
1510  VTAB 22: PRINT "SELECT: (A)
      ADD (E) EDIT (N) NEXT BOX:"
      ;: GET A$: PRINT A$
1520  IF A$ = "E" THEN  INPUT "WH
      ICH CIRCUIT? ";B$: GOTO 1610

1530  IF A$ = "N" THEN 1750
1540  IF A$ <  > "A" THEN 1510
1550  HOME : IF (NC + 1) > 24 THEN
      PRINT "SORRY.  ONLY 24 CIRC
      UITS ALLOWED.": FOR L = 1 TO
      2000: NEXT : GOTO 510
1560 NC = NC + 1: VTAB 6: PRINT  TAB(
      5)"ENTERING CIRCUIT NUMBER "
      ;NC
1570  VTAB 8: HTAB 3: INPUT "BOX
      NUMBER = ";A$:CI(NC - 1,0) =
      VAL (A$)
1580  PRINT : HTAB 3: INPUT "VOLT
      AGE = ";A$:CI(NC - 1,1) =  VAL
      (A$)
1590  PRINT : HTAB 3: INPUT "BREA
      KER/FUSE AMPERAGE = ";A$:CI(
      NC - 1,2) =  VAL (A$)
1600  GOTO 510
1610 C =  VAL (B$): HOME : FOR L =
      0 TO 39: PRINT "-";: NEXT : PRINT
      TAB( 5)"EDITING CIRCUIT NUM
      BER ";C: FOR L = 0 TO 39: PRINT
      "-";: NEXT : PRINT
1620  PRINT "AS EACH VALUE IS DIS
      PLAYED, ENTER A NEW VALUE TO
      CHANGE OR <RETURN> TO RETAI
      N   THE OLD VALUE:": PRINT
1630  PRINT  TAB( 2)"BOX NUMBER =
      ";CI(C - 1,0);"  NEW BOX =
      ";: INPUT "";A$
1640  IF A$ = "" THEN 1670
1650  IF  VAL (A$) < 1 OR  VAL (A
      $) > 4 THEN  CALL  - 198: PRINT
      "ENTER 1-4 PLEASE.": GOTO 16
      30
1660 CI(C - 1,0) =  VAL (A$)
1670  PRINT : PRINT  TAB( 2)"VOLT
      AGE = ";CI(C - 1,1);"  NEW V
      OLTAGE = ";: INPUT "";A$
1680  IF A$ = "" THEN 1710
1690  IF  VAL (A$) < 105 OR  VAL
      (A$) > 240 THEN  CALL  - 198
      : PRINT "UNLIKELY. ENTER 105
      -240 PLEASE.": GOTO 1670
1700 CI(C - 1,1) =  VAL (A$)
1710  PRINT : PRINT  TAB( 2)"BREA
      KER/FUSE AMPERAGE = ";CI(C -
      1,2): PRINT  TAB( 6)"NEW AMP
      ERAGE = ";: INPUT "";A$
1720  IF A$ = "" THEN 1750
1730  IF  VAL (A$) < 10 OR  VAL (
      A$) > 100 THEN  CALL  - 198:
      PRINT "UNLIKELY. ENTER 10-1
      00 PLEASE.": GOTO 1710
1740 CI(C - 1,2) =  VAL (A$)
1750  NEXT
1760  GOTO 510
1770  REM  DISPLAY A CIRCUIT
1780  HOME : VTAB 12: INPUT "WHIC
      H CIRCUIT DO YOU WISH TO DIS
      PLAY: ";A$
1790 I =  VAL (A$): IF I < 1 OR I
      > NC THEN  PRINT : PRINT "O
      NLY ";NC;" CIRCUITS AVAILABL
      E: ";: INPUT "";A$: GOTO 179
      0
1800  GOSUB 270: REM  DISPLAY FLO
      ORPLAN
1810  HCOLOR= 3
1820  FOR J = 1 TO NF
1830  IF FX(J,0) <  > I THEN 1900

1840 K = FX(J,3):L = FX(J,4)
1850  IF K = 0 THEN K = 1
1860  IF K = 279 THEN K = 278
1870  IF L = 0 THEN L = 1
1880  IF L = 191 THEN L = 190
1890  HPLOT K + 1,L - 1 TO K + 1,
      L + 1: HPLOT K,L - 1 TO K,L +
      1: HPLOT K - 1,L - 1 TO K -
      1,L + 1
1900  NEXT
1910  GET A$: GOTO 510
1920  REM  PRINT FIXTURES/APPLIAN
      CES
1930  GOSUB 2600: REM  SORT ROUTI
      NE
1940  HOME : VTAB 10: PRINT "DISP
      LAY ON SCREEN OR PRINTER (S/
      P)? ";: GET A$: IF A$ <  > "
      S" AND A$ <  > "P" THEN 1940

1950  HTAB 1: IF A$ = "S" THEN  HOME

1960  IF A$ = "P" THEN  PR# 1
1970  PRINT "FIXTURE"; TAB( 15)"R
      OOM"; TAB( 28)"WATTS"; TAB(
      34)"AMPS": FOR I = 0 TO 39: PRINT
      "-";: NEXT
1980  IF A$ = "S" THEN  POKE 34,4

1990  FOR I = 0 TO NC - 1
2000  IF A$ = "S" THEN  HOME
2010  PRINT : PRINT "CIRCUIT NO.
      ";I + 1
```

```
2020  PRINT
2030  FOR J = 1 TO NF: IF FX(J,0)
      <  > I + 1 THEN 2070
2040 FX(J,1) =  INT (FX(J,1))
2050 FX(J,2) =  INT (FX(J,2) * 10
     0) / 100
2060  PRINT  LEFT$ (NA$(J),13); TAB(
     15) LEFT$ (RM$(J),11); TAB(
     28)FX(J,1); TAB( 34)FX(J,2)
2070  NEXT : PRINT
2080  IF A$ = "S" THEN  PRINT : PRINT
     : GOSUB 110
2090  NEXT
2100  PR# 0
2110  GOTO 510
2120  REM  ANALYZE CIRCUITS
2130  HOME : FOR I = 0 TO 39: PRINT
     "-";: NEXT : PRINT  TAB( 10)
     "CIRCUIT ANALYSIS": FOR I =
     0 TO 39: PRINT "-";: NEXT : PRINT

2140  PRINT "CIRCUIT VOLTS BREAKE
     R AMPS USED # FIXT.": PRINT
     "------- ----- ------- -----
     ---- -------"
2150 S = 0
2160  FOR I = S TO S + 11:TF = 0:
     TA = 0
2170  FOR J = 1 TO NF: IF FX(J,0)
      = I + 1 THEN TF = TF + 1:TA
      = TA + FX(J,2)
2180  NEXT
2190 TA =  INT (TA * 100) / 100
2200  IF TA > CI(I,2) THEN  INVERSE

2210  PRINT  TAB( 3)I + 1; TAB( 9
     )CI(I,1); TAB( 15)CI(I,2)" A
     MPS"; TAB( 24)TA; TAB( 35)TF

2220  NORMAL
2230  IF I =  > NC THEN 2250
2240  NEXT
2250  GOSUB 110
2260  IF NC > 12 THEN S = 12: GOTO
     2160
2270  GOTO 510
2280  REM  INITIALIZE
2290  DIM CI(23,2),FX(200,4),NA$(
     200),RM$(200)
2300 D$ =  CHR$ (4):OP$ = D$ + "O
     PEN ":RE$ = D$ + "READ ":WR$
      = D$ + "WRITE ":DE$ = D$ +
     "DELETE ":CL$ = D$ + "CLOSE"

2310  HOME
2320  ONERR  GOTO 2730
2330  PRINT OP$"CIRCUITS"
2340  PRINT RE$"CIRCUITS"
2350  INPUT NC
2360  FOR I = 0 TO NC - 1: FOR J =
     0 TO 2: INPUT CI(I,J): NEXT
     : NEXT
2370  INPUT NF
2380  FOR I = 1 TO NF: INPUT NA$(
     I): INPUT RM$(I)
2390  FOR J = 0 TO 4: INPUT FX(I,
     J): NEXT : NEXT
2400  PRINT CL$
2410  GOTO 510: REM  MAIN MENU
2420  REM  INITIALIZE A NEW FILE
2430 NC = 0:NF = 0: HOME : FOR I =
     0 TO 39: PRINT "-";: NEXT : PRINT
      TAB( 8)"INITIALIZE NEW SYST
     EM": FOR I = 0 TO 39: PRINT
     "-";: NEXT : PRINT
2440  INPUT "HOW MANY CIRCUITS? "
     ;A$: IF  VAL (A$) < 1 OR  VAL
     (A$) > 24 THEN  CALL  - 198:
      PRINT "ENTER A NUMBER FROM
     1 TO 24 PLEASE.": GOTO 2440
2450 NC =  VAL (A$)
2460  FOR I = 1 TO NC
2470  PRINT : PRINT "CIRCUIT #";I
     ;":"
2480  HTAB 3: INPUT "WHICH BOX #
     IS THIS CIRCUIT IN? ";A$
2490  IF  VAL (A$) < 1 OR  VAL (A
     $) > 4 THEN  CALL  - 198: PRINT
      TAB( 5)"ENTER A NUMBER FROM
      1 TO 4 PLEASE.": GOTO 2480
2500 CI(I - 1,0) =  VAL (A$)
2510  HTAB 3: INPUT "VOLTAGE: ";A
     $
2520  IF  VAL (A$) < 105 OR  VAL
     (A$) > 240 THEN  CALL  - 198
     : PRINT "UNLIKELY. ENTER 105
     -240 PLEASE.": GOTO 2510
2530 CI(I - 1,1) =  VAL (A$)
2540  HTAB 3: INPUT "BREAKER/FUSE
      SIZE (IN AMPS): ";A$
2550  IF  VAL (A$) < 10 OR  VAL (
     A$) > 100 THEN  CALL  - 198:
      PRINT "UNLIKELY. ENTER 10-1
     00 PLEASE.": GOTO 2540
2560 CI(I - 1,2) =  VAL (A$)
2570  NEXT
2580  GOTO 790
2590  REM  SORT ROUTINE
2600  HOME : VTAB 10: PRINT "SORT
     ING.";
2610  FOR J = 1 TO NF - 1
2620  PRINT ".";
2630  FOR I = J TO NF - 1
```

```
2640  IF RM$(I) <  = RM$(I + 1) THEN
     2690
2650 T1$ = NA$(I):T2$ = RM$(I):T1
      = FX(I,0):T2 = FX(I,1):T3 =
     FX(I,2):T4 = FX(I,3):T5 = FX
     (I,4)
2660 NA$(I) = NA$(I + 1):RM$(I) =
     RM$(I + 1)
2670  FOR K = 0 TO 4:FX(I,K) = FX
     (I + 1,K): NEXT
2680 NA$(I + 1) = T1$:RM$(I + 1) =
     T2$:FX(I + 1,0) = T1:FX(I +
     1,1) = T2:FX(I + 1,2) = T3:F
     X(I + 1,3) = T4:FX(I + 1,4) =
     T5
2690  NEXT I
2700  NEXT J
2710  PRINT : RETURN
2720  REM  ERROR HANDLING ROUTINE
     S
2730 ER =  PEEK (222):EL =  PEEK
     (218) +  PEEK (219) * 256
2740  POKE 216,0
2750  IF ER = 5 THEN 2430
2760  IF ER = 16 THEN  PRINT "SYN
     TAX ERROR ON LINE ";EL
2770  IF ER = 6 THEN  PRINT "FILE
      NOT FOUND."
2780  IF ER = 4 THEN  PRINT "WRIT
     E PROTECTED"
2790  PRINT "ERROR NUMBER ";ER;"
     DETECTED ON LINE ";EL
2800  END
```


WHAT IS A 16K RAM CARD AND WHY IS IT IN MY SLOT 0?

BY WALT MILLS

Reprinted from : THE WASHINGTON APPLE PI, MARCH 1982

The intention of this article is to describe (in very simple terms) the various uses of the 16K RAM card in the Apple II. If you are currently using a 16K Ram card (or Language card) THEN GOTO the next article; you may be able to tell me a thing or two.

WHAT IS A 16K RAM CARD?

First, a 16K RAM card by any manufacturer may use the same software as the original APPLE "Language Card"; the only difference among these various cards is price and some minor hardware configurations. The 16K RAM card is simply a printed circuit card with 16,384 bytes of additional addressable RAM. (What)? OK, first a couple of definitions: RAM-Random Access Memory- the changeable memory portion of your computer where programs are stored; Byte- A collection of points in RAM used to store a single number (from 0 to 255). Note: When you POKE or PEEK to memory, you are looking at one byte. If you have a 48K (RAM) machine, you can add a 16K RAM card and increase your memory size to 64K (now we're talking). But alas, Woz didn't design it that way, so first let's look at how our memory is laid out.

The first location in the APPLE is called Location Zero (0) - (how about that!) - and the first 256 bytes ($100 hex) are called Page Zero. "Pages" of APPLE memory are divided at every 256 bytes or $100 in Hexadecimal. You can actually poke a value into location zero with the command POKE 0,65. You can print what is there with the command PRINT PEEK (0). Most locations in Page Zero are called "reserved"- that is, Applesoft uses these bytes to store items it wants to remember. You can use any RAM location, but you will probably bomb something if you POKE around in a reserved space.

Page One is reserved for the system stack;

Page Two is the input (typing) buffer. Some of Page Three (locations 768 to 1023) is reserved for DOS. Pages Four, Five, Six and Seven are where the Apple stores the characters to be displayed on the screen. The space between 2049 to 40191 is for the Applesoft or Integer program (8192 to 24575 houses the Hires pages too). The actual Disk Operating System (DOS) is located from 40192 to 49151. BOOM! The top of a 48K machine.

The locations from 49152 to 53247 may be used (and there are above 48K), but these bytes are generally divided amoung the eight expansion slots in your APPLE and used for storage by the cards. Locations 53248 to 57343 are called the Monitor (Autostart in APPLE][Plus). Locations 57344 to 65535 are used to hold Applesoft or Interger Basic. In the Apple][Plus, the Applesoft ROM Basic (Read Only Memory) resides at this location. When you insert the 16K RAM card into slot Zero, it will fall in line at 53248 to 65535. (Excuse me, but 53248 subtracted from 65535 is 12287 or about 12K not 16K.) Right, the first 4K of the 16K RAM Card can be exchanged with a second 4K on the card as needed (more about that later). With the 16K RAM card in place you have actually duplicated the space from 53248 to 65535 and with the aid of DOS you may chose which area of memory you want to use. Normally, this is accomplished with the command INT and FP.

NOW SOMETHING USEFUL

If you have one, you will have to remove your Integer/Applesoft card from slot zero, lay it gently on the shelf and insert the 16K RAM card (following manufacturer's MFG directions). The first useful thing you can do with the 16K card is load the missing language (Integer will be assumed) to the card. Like the "soft switches" for the screen display (see APPLE][Reference Manuel, page 12) the 16K card has switches to control what may happen to it. If you PEEK/POKE to location-16255 you will write-enable the card (like removing the little tab from the disk). If you then BLOAD a program at location 53248 (HEX $D000) it will await your "Call" (pun intended). Believe it or not, Integer (and Applesoft) is nothing but a big binary program!! If you write-enable the card and type BLOAD INTBASIC, A$D000 you will have a machine with Integer Basic. (Quick, check the shelf, the old card is still there - Magic!). Actually the DOS 3.3 HELLO program will check to see if you have a 16K RAM card and do this load for you on boot-up. If you did not previously have Integer/Applesoft you can see the obvious advantages. Non-Integer owners will now have the Programmer's Aid #1 (step, trace, renumber, etc.). If you previously had an Integer Card, you will also note that the ESC I,J,K,M work as in Applesoft - this is because the Binary INTBASIC has an image of the new Autostart ROM.

OTHER LANGUAGES

If one secures a disk copy of APPLE Pascal or Fortran then it is a simple matter of loading and running these languages much like Basic.

MOVING DOS

Our user's library has a program that will allow you to actually move DOS up to the 16K RAM card and allow you to regain the use of the 10k now used by DOS in high memory. Other commercial sources have DOS movers that allow you to run both DOS 3.2 and 3.3.

NEAT STUFF

A company called Omega Software Products (the folks that brought you LOCKSMITH) market a program called "THE INSPECTOR" that may be overlaid on the INTBASIC program at location 55296 ($D800) which is unused by Integer Basic. When called by CALL -10240, a very powerful disk inspector is evoked. The beauty of this type of overlay is that it is totally transparent to other machine internals and remains ready to be called as needed.
Other short machine language programs can be hidden in this area. If you BSAVE the memory locations from 53248 ($D0000 to 65535 ($FFFF) you will have a copy INTBASIC with your new program neatly "hidden".

THE SECOND FOUR KILOBYTES

By again flipping some softswitches you

may turn off the first bank 4K of the 16K RAM card and turn on the second bank of 4K in its place. This gives us some interesting possibilities; if we write a clever binary program at location 768 (the first portion of Page Three that is unused) we can actually exchange the 4K of RAM and bring a previously stored program online at location 53248 ($D000) to 57343 ($DFFF). A company called Telephone Software Connection has developed a rather unique approach to this concept by actually changing the jump location and syntax of the DOS "CHAIN" command to "CNVRT" which will exchange the 4K and put you into a neat Hex to Dec to Hex converter without clobbering DOS, Basic or variables. This means that at any point while entering a program (FP or INT) you may type CNVRT to jump to the converter!

DON'T THROW THE INTEGER CARD AWAY

Finally, remember to keep that old Integer/Applesoft ROM card around. I understand that you can pull the existing chips and replace them with up to 8 custom PROM's - obviously stuff for another article.

THE SOURCE

The Source provides a number of interesting and useful services to those who have systems equipped with modems. Among those services is MAIL, an electronic mail capability. In this issue the Apple Barrel will begin listing the Source account numbers of those interested in using this means of sending messages (or programs for that matter). It can also be used to submit articles, wantads, etc. to the Apple Barrel by sending it to my account number. If you do submit material in this way, call the HOTLINE and leave word that I have MAIL waiting as I do not access the Source on a predictable schedule.

Steve Knouse ST8337
Mike Kramer ST3030

Apple /// BASIC Compare

By Mike Kramer

One of my all time favorite utility programs has been Applesoft Compare, written by Chuck Boody and first published in the July/August 1980 issue of Call - A.P.P.L.E. I don't know if the errors in the listing belonged to Chuck or if the type setter caused them, but it just would not work as listed in the article. With a little effort, however, I found the problems and soon had COMPARE working. I have to admit that I had never taken the time to fully understand the workings of COMPARE, but I have used it regularly to find out how the .1, .2, and .3 versions of a program under development differed after a time lapse of several weeks.

I bought an Apple /// to keep my old faithful Apple][company and to minimize contention with my wife and kids for machine time. I soon learned the joys of Apple /// Business BASIC and found myself devoting more and more time to working with the Apple ///. Before long I was up to my old tricks and found my work disks getting filled up with multiple versions of the same program. Although I had been writing a good deal of BASIC code on the Apple ///, I was reluctant to try to convert good old Applesoft COMPARE. This was mainly because I would have to determine how COMPARE worked plus get a little deeper into error handling in Business BASIC. Soon the need got so bad that I decided to go ahead and do the conversion.

Having just received a copy of APPLECON the public domain Applesoft to Business BASIC conversion program, I used it to minimize the conversion time. Unlike the old Integer to Applesoft converters, APPLECON actually converts Applesoft commands to Business BASIC commands. It changes the text window POKEs to WINDOW commands, VTAB to VPOS =, etc. It cannot, however, convert obscure PEEKs, POKEs, and CALLS, but does flag statements it can't handle by preceding them with a REM statement containing a line of dashes. But then this is not an article on APPLECON, is it? The result is presented in Listing 1 at the end of this article.

SO HOW DOES APPLE /// BASIC COMPARE WORK?

COMPARE alternately reads two Apple /// Business BASIC programs stored as TEXT files, checks for added, deleted, or changed lines, and lists the differences on the specified output device (.PRINTER, .CONSOLE, .D2/filename...). As mentioned earlier, the original Applesoft COMPARE program logic was somewhat difficult to understand. This was partly due to the use of obscure variable names and partly due to the limitations of Applesoft error handling. Since Business BASIC permits variable names with up to 64 significant characters, very descriptive variable names were used even at the loss of some execution speed. Some performance was also sacrificed by heavy use of REM statements, but the resulting program (see Listing 1) is easy to read and the logic is easy to follow. With the better error handling provided by Business BASIC, particularly the handling of end of file (EOF) conditions, it was possible to handle error conditions in a more straight forward manner.

In order to speed program execution, the subroutines which read the text files and print to the output device are placed at the beginning of the program. The first thing the program does is jump to Line 200 where a small amount of initialization is done, the title block is displayed on the screen, and the text WINDOW is set to keep the title block on the screen throughout program execution. Lines 260 - 270 determine if instructions are to be displayed. If the instructions in Lines 790 - 860 are displayed, the choice is given to continue with the comparison or end so that the necessary TEXT files may be prepared.

Line 300 displays the default destination pathname for the results of the comparison and waits for a pathname to be entered. Line 310 repositions the cursor and outputs the selected output pathname. This is done so that the default pathname will remain on the

screen if a RETURN is pressed to accept the displayed default. The logic used in Line 310 in determining the vertical position assures proper cursor placement if the dialog is on the bottom line. Line 320 OPENs the selected device for output.

The names of the files to be compared are entered in Lines 340 - 430. In Line 350 a flag is set to permit the error handling routine in Lines 910 - 920 to determine whether an error was made when entering the old file name or the new file name. A 1 or 0 could have been used as a flag, but by using a string containing the words "old" or "new" there is no question in the mind of the person reading the listing as to whether the error occurred in opening or reading the old or new file. Line 360 checks to see whether to display the catalog, open a file for input, or forget the whole thing. If a file name is entered, Line 370 prepares to handle an error condition should the file opened in Line 380 not exist. The same steps are repeated in Lines 390 for the new file. Lines 430 - 460 prepare for outputting the results, while Lines 470 - 480 set up what should be done when the end of either file is reached.

The files are read in Line 510, where GOSUB 80 results in the reading of a line of code from the "old" file and GOSUB 120 causes a line to be read from the "new" file. Line 80 initializes the input string variable for the "old" file to null. Line 90 reads the file one character at a time, building the input string, returning to the main program when a RETURN character is found or the string length reaches 255. If the end of the "old" file is reached, execution continues with Line 620. The same procedure is followed in Line 120 for the "new" file. If the end of the "new" file is reached execution continues with Line 650.

It is important to remember that the strings read in are lines of BASIC code that begin with a line number. Line 530 compares the "old" line to the "new" line. If there is a perfect match, nothing is printed and another line is read from each file. If there is not a perfect match, the line number of each program line is determined in Line 540 by taking the VAL of the two strings containing the "old" statement and the "new" statement.

Line 560 checks for changed lines by looking for matching line numbers and differences in the program line itself. If the line numbers match but the contents of the program lines differ, both the old and new versions of the line are printed along with indication that there was a change in the line.

Line 580 checks for deleted lines by seeing if the old line number is less than the new line number. If it is less, the line is printed and noted as deleted. Since the "old" line has been "used up", it is necessary to read another "old" line to get back into synchronization with the "new" file, hence the GOSUB 80.

If the checks in Lines 530 - 580 fail, the only remaining possibility is that the line was added. The line is printed and noted as having been added and a "new" line is read from disk with a GOSUB 120 to get back into synch with the "old" file. A GOTO 530 is executed to restart the comparison process.

If the "old" file becomes empty first, a branch is made to Line 620, where a new ON EOF is executed to cause the comparison to terminate when the end of the "new" file is reached. Since there is nothing remaining in the "old" file, the remaining lines are read from the "new" file in Line 630, printed, and flagged as having been added.

If the "new" file becomes empty first, a branch is made to Line 650, where a new ON EOF is executed for the "old" file. The remaining lines in the "old" file are read, printed, and flagged as having been deleted.

When both the "old" and "new" files have been depleted, a branch is made to Line 680 where the"End of Comparison" message is printed and the files and output device are closed.

Lines 710 - 770 provide the options of deleting the text files and doing another comparison.

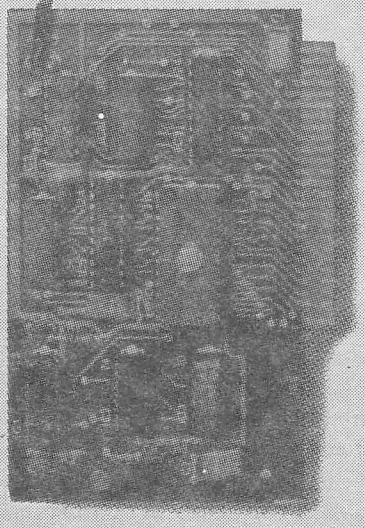
Lines 790 - 890 display the instructions and offers the options of ending or continuing program execution.

Lines 910 - 920 perform limited error handling to prevent program termination if a

WILDCARD™

MAKES BACK-UP COPIES OF PROTECTED SOFTWARE QUICKLY, EASILY, WITH JUST A PUSH OF A BUTTON.

New software locking schemes have rendered even the latest generation of copy programs virtually unusable. Locksmith™, Nibbles Away™ and other "Nibble copiers" require complicated parameter settings, much patience and great effort to use. More often than not, the results are disappointing. WILDCARD is different. Rather than copying disks track by track, WILDCARD ignores the disk and any copy protection encrypted on it. Instead, WILDCARD takes a snapshot of memory in your Apple® II. Now you can make back-up copies of protected software with the push of a button.



Features

- Hardware copying device...push button operation.
- Copies ALL* 48K memory resident software, most 64K software.
- No Parameters are necessary.
- WILDCARD lives in any slot.
- WILDCARD is undetectable by software.
- Produces autobooting disk in 2 minutes.
- Copies become accessible for alterations.
- Copies are DOS 3.3 compatible.
- Software and utilities included.

System requirements: Apple II Plus with 64K and DOS 3.3.
* Wildcard does not operate with CP/M® or other microprocessor based software.

$129.95 direct from East Side Software Co., 344 E. 63 St., Suite 14-A, New York City 10021, 212/355-2860. Please include $3.00 for handling. Mail and phone orders may be charged to MasterCard and VISA. N.Y. State residents add sales tax. Dealer inquiries welcome.

IMPORTANT NOTICE: The WILDCARD is offered for the purpose of enabling you to make archival copies only. Under the Copyright Law you, as the owner of a copy of a computer program, are entitled to make a new copy for archival purposes only and the WILDCARD will enable you to do so. The WILDCARD is offered for no other purpose and you are not permitted to utilize it for any other use, other than that specified.

bad program, volume, or pathname is entered in Lines 340 and 390. If any other error occurs, the error number and line containing the error are printed by line 920 and program execution ends.

CREATING A CAPTURE PROGRAM

The short Business BASIC program in Listing 2 creates a TEXT file called CAPTURE.EXEC which, when EXECed into a BASIC program in memory, will save the program as a TEXT file when RUN is typed. CAPTURE.EXEC should be saved on the same diskette as the COMPARE program. Note that the reserved variable OUTREC is set equal to 255 to permit saving the longest possible program line (the CAPTURE program in the Apple Business BASIC Reference Manual fails to do this) and then reset to the default value of 80. After the program in Listing 2 has been typed in, it should be executed by typing RUN 5 to create CAPTURE.EXEC. The program in Listing 2 should also be saved for later use.

SAVING BASIC PROGRAMS AS TEXT FILES

The next step is to save the two Business BASIC programs to be compared as TEXT files on the diskette containing the COMPARE program. First set the Prefix to default to the disk containing the COMPARE program by typing PREFIX$=/COMPARE (assuming the volume is named COMPARE) . Next load the older version of the program into memory with a LOAD command. Then type EXEC CAPTURE.EXEC. When the cursor returns, type RUN. The program will ask for a pathname for the output file. If you have set the prefix as suggested above, just type the file name. Otherwise type the full pathname. Do not give the name of the original program, but rather a similar name with ".OLD" appended to it. The file name must follow the file naming rules, beginning with a letter and consisting of 15 or fewer letters, numbers, or periods. When the cursor returns, catalog the disk to assure that the file was saved. Repeat the steps for the new file, substituting ".NEW" for ".OLD" in the file name.

COMPARING THE BASIC PROGRAMS

After the two programs have been saved as TEXT files, run the COMPARE program. You will be asked if you want instructions. Answer with a single keystroke, either Y or N. Next you will be asked for the destination pathname. At this point you may specify any valid output device, such as .PRINTER, .CONSOLE, or a file name. The first time through the program the default destination is .PRINTER. On subsequent passes, the previously specified destination becomes the default. Next you are asked for the name of the TEXT file containing the old version of the program to be compared. At this point you may type the file name, END, or CAT if you want to see the catalog. If an invalid or nonexistent file name is entered, you will be asked to enter it again. Next you will be asked to enter the name of the TEXT file containing the new version of the program. The dialog and options are identical to those for the old program. The results of the comparison will then be output to the specified destination device. When the comparison is finished, the program will beep and ask if the files compared should be deleted. You are then given the chance to make more comparisons.

SAMPLE RUN

Listing 3 contains two similar programs which, if compared, will illustrate the use of COMPARE. Listing 4 gives the results of the comparison, showing changed, added, and deleted lines. Two runs were made, showing OLD.BASIC to be the older version the first time and the newer version the second time.

LISTING 1

```
 10    REM            *** Apple /// BASIC Compare ***
 20    REM                Written by Mike Kramer
 30    REM                     11/10/82
 40    REM          Based on a program by Charles Boody


 50    GOTO 200
 60    REM

** Miscellaneous Subroutines **

 70    REM

 ** Get program line from old file **

 80    old.statement$=""
 90    GET#2;char$:IF char$<>retrn$ OR LEN(old.statement$)=0 THEN IF L
       EN(old.statement$)<255 THEN old.statement$=old.statement$+char$
       :GOTO 90
 100    RETURN
 110    REM

 ** Get program line from new file **

 120    new.statement$=""
 130    GET#3;char$:IF char$<>retrn$ OR LEN(new.statement$)=0 THEN IF
       LEN(new.statement$)<255 THEN new.statement$=new.statement$+cha
       r$:GOTO 130
 140    RETURN
 150    REM

** Print program line on output device **

 160    char.count=0
 170    start.char=char.count+1:char.count=char.count+line.length:PRIN
       T#1; TAB(6);MID$(line.to.print$,start.char,line.length):IF cha
       r.count<LEN(line.to.print$) THEN 170
 180    RETURN
 190    REM

 ** Main Program **

 200    blank$="
                                    ":REM 80 spaces
 210    line.length=60:bell$=CHR$(7):retrn$=CHR$(13):escape$=CHR$(27):
       clear.to.end.of.line$=CHR$(31)
 220    output.pathname$=".PRINTER":PREFIX$="/compare"
 230    TEXT:HOME:INVERSE:VPOS=1:FOR i=1 TO 4:PRINT blank$:NEXT
 240    VPOS=2:HPOS=25:PRINT"*** Apple /// BASIC Compare ***":HPOS=29:
       PRINT"Written by Mike Kramer":NORMAL
 250    WINDOW 0,5 TO 80,24
 260    VPOS=2:HPOS=33:PRINT"Instructions? ";:GET response$
 270    IF response$="y" OR response$="Y" GOTO 790
 280    WINDOW 0,5 TO 80,24
```

```
290    REM

** Get output pathname **

300    HOME:VPOS=2:PRINT:vtab= VPOS:PRINT"Destination pathname: ";out
       put.pathname$;:VPOS=vtab:HPOS=23:INPUT"";response$:IF response
       $<>"" THEN output.pathname$=response$
310    HPOS=23:VPOS=vtab-2*(vtab=24):PRINT output.pathname$
320    OPEN#1 AS OUTPUT,output.pathname$
330    REM

** Enter names of files to compare, END, or CAT **

340    vtab= VPOS:PRINT:INPUT"Old program text file name, END, CAT: "
       ;old.file.name$
350    new.or.old$="old"
360    IF old.file.name$="" THEN VPOS=vtab-2*(vtab=24):GOTO 340:ELSE
       IF LEFT$(old.file.name$,3)="cat" OR LEFT$(old.file.name$,3)="C
       AT" THEN CATALOG:GOTO 340:ELSE IF old.file.name$="end" OR old.
       file.name$="END" THEN 890:GOTO 340
370    ON ERR GOSUB 910
380    OPEN#2 AS INPUT,old.file.name$
390    vtab= VPOS:PRINT:INPUT"New program text file name, END, CAT: "
       ;new.file.name$
400    new.or.old$="new"
410    IF new.file.name$="" THEN VPOS=vtab-2*(vtab=24):GOTO 390:ELSE
       IF LEFT$(new.file.name$,3)="cat" OR LEFT$(new.file.name$,3)="C
       AT" THEN CATALOG:GOTO 390:ELSE IF new.file.name$="end" OR new.
       file.name$="END" THEN 890:GOTO 390
420    OPEN#3 AS INPUT,new.file.name$
430    HOME:IF output.pathname$=".console" OR output.pathname$=".CONS
       OLE" THEN 450
440    VPOS=11:HPOS=29:INVERSE:PRINT" Printing Comparison ":NORMAL
450    title$="Comparison of "+old.file.name$+" to "+new.file.name$
460    PRINT#1:PRINT#1 SPC(INT(40-LEN(title$)/2));title$:PRINT#1
470    ON EOF#2 GOTO 620
480    ON EOF#3 GOTO 650
490    GOTO 510
500    REM

** Read in old and new program lines **

510    GOSUB 80:GOSUB 120
520    REM

** Check for unchanged line **

530    IF old.statement$=new.statement$ THEN 510
540    old.line.num=VAL(old.statement$):new.line.num=VAL(new.statemen
       t$)
550    REM

** Check for changed line **
```

RIGEL COMPUTER SYSTEMS, INC.

12496 Bellaire Boulevard • Suite #92 • Houston, Texas 77072 • (713) 481-1063

RIGEL COMPUTER SYSTEMS, the STAR in the microcomputer field, wants to get you what you need for your APPLE computer.

		List Price	OUR PRICE
Here are some of SOFTALK's Bestsellers:			
CHOPLIFTER	Broderbund	$ 34.95	$ 24.95
WORD HANDLER	Silicon Valley	199.95	139.95
MASTERTYPE	Lightning	39.95	28.95
ZORK I	Infocom	39.95	28.95
WIZARDRY	Sir-Tech	49.95	35.95
CASTLE WOLFENSTEIN	Muse	29.95	21.95
VISICALC 3.3	Visicorp	250.00	179.95
DOSS BOSS	Beagle	24.00	17.95
HOME ACCOUNTANT	Continental	74.95	53.95
Popular Hardware:			
PROWRITER 8510	C. Itoh	$ 495.00	$ 409.95
SYSTEM SAVER	Kensington	89.95	69.95
MICROBUFFER II	Prac. Periph.	259.95	199.95
ELITE I	Rana Drives	449.00	319.95
MICROMODEM W/ T.P.	D.C. Hayes	409.00	275.95

NEW!!! NEW!!! NEW!!!
KRAFT JOYSTICS,
IDS PRISM AND
MICROPRISM PRINTERS,
AND MUCH MORE.
Our list grows everday!!!

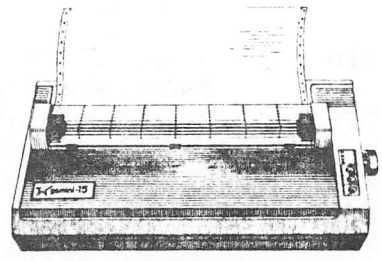
PRICE DROPS NOT REFLECTED IN OUR NEW CATALOG:		
COLOR I	Amdek	$ 319.95
GEMINI-10	Star	339.95

IF YOU DON'T CALL US BEFORE YOU BUY, THEN YOU WON'T GET THE BEST PRICE!
Prices listed are good for the month in which they are advertised; thereafter are subject to change without notice.

CALL or write for our new FREE catalog (watch for the wet ink).

WE ARE NOT A STORE!!!

star micronics·inc
THE POWER BEHIND THE PRINTED WORD.



APPLE is a registered trademark of Apple Computers, Inc.

```
560    IF old.line.num=new.line.num AND old.statement$<>new.statement
       $ THEN PRINT#1:PRINT#1;" ** Changed **":line.to.print$=old.sta
       tement$:GOSUB 160:PRINT#1;" To: ":line.to.print$=new.statement
       $:GOSUB 160:GOTO 510
570    REM

** Check for deleted lines **

580    IF old.line.num<new.line.num THEN PRINT#1:PRINT#1;" ** Deleted
        **":line.to.print$=old.statement$:GOSUB 160:GOSUB 80:GOTO 530
590    REM

** None of above so must be added **

600    PRINT#1:PRINT#1;" ** Added **":line.to.print$=new.statement$:G
       OSUB 160:GOSUB 120:GOTO 530
610    REM

** When old file is empty, show rest of new file as adde
       d **

620    ON EOF#3 GOTO 680
630    GOSUB 120:PRINT#1:PRINT#1;" ** Added **":line.to.print$=new.st
       atement$:GOSUB 160:GOTO 630
640    REM

** When new file is empty, show rest of old file as adde
       d **

650    ON EOF#2 GOTO 680
660    PRINT#1:PRINT#1;" ** Deleted **":line.to.print$=old.statement$
       :GOSUB 160:GOSUB 80:GOTO 660
670    REM

** Assume end of data error in last file and end program
        **


680    PRINT#1:PRINT#1; TAB(28);" ** End of Comparisons **":PRINT#1:P
       RINT#1:PRINT#1
690    CLOSE
700    REM

** Delete text files if desired and end or continue **


710    IF output.pathname$<>".console" AND output.pathname$<>".CONSOL
       E" THEN VPOS=13:PRINT clear.to.end.of.line$
720    PRINT:HPOS=31:PRINT bell$;"Delete text files? ";:GET response$
       :IF response$<>"y" AND response$<>"Y" GOTO 750
730    DELETE old.file.name$
740    DELETE new.file.name$
750    PRINT:PRINT:HPOS=32:PRINT"More comparisons? ";:GET response$
760    IF response$="Y" OR response$="y" THEN PRINT:GOTO 230
770    TEXT:HOME:END
```

```
780   REM

** Instructions **

790   HOME:VPOS=2:PRINT"Before running 'COMPARE' the EXEC file 'CAPT
      URE' must be created and the two"
800   PRINT"programs to be compared must be stored as text files.  T
      he steps are:"
810   WINDOW 15,7 TO 80,24
820   VPOS=3
830   PRINT" (1) Load 1st program for comparison":PRINT" (2) Type 'E
      XEC CAPTURE.EXEC'"
840   PRINT" (3) Type 'RUN'. When asked, type old file's name":PRINT
      " (4) Load 2nd program for comparison":PRINT" (5) Type 'EXEC C
      APTURE.EXEC'":PRINT" (6) Type 'RUN'. When asked, type new file
      's name"
850   PRINT" (7) Run 'COMPARE'. When asked, enter old":PRINT TAB(6)"
      file's name and new file's name":PRINT" (8) Go get a snack"
860   PRINT:PRINT TAB(10);"Press <RETURN> to run 'COMPARE'":PRINT TA
      B(10);"Press <ESCAPE> to end":GET a$
870   IF a$<>retrn$ AND a$<>escape$ GOTO 790
880   IF a$=retrn$ GOTO 280
890   TEXT:HOME:END
900   REM

** Error handling **

910   IF ERR>28 AND ERR<33 THEN PRINT:PRINT bell$;"File not found.":
      POP:IF new.or.old$="old" THEN 340:ELSE GOTO 390
920   HOME:VPOS=12:HPOS=30:PRINT"Error "; ERR;" in Line "; ERRLIN:EN
      D
```

LISTING 2

```
0   OUTREC=255:TEXT:HOME:INPUT"Pathname for captured file: ";a$:CREA
    TE a$, TEXT:OPEN#1 AS OUTPUT,a$:OUTPUT#1:LIST 1-:PRINT:CLOSE:OUT
    REC=80:END
5   OUTREC=255:HOME:INPUT"Target drive for CAPTURE EXEC (1/2) ";d$:a
    $=".d"+d$+"/CAPTURE.EXEC":CREATE a$, TEXT:OPEN#1 AS OUTPUT,a$:OU
    TPUT#1:LIST 0:PRINT:CLOSE:OUTREC=80:END
```

INTERNATIONAL APPLE CORE ™

MEMORANDUM

To: Candidates for Director
Directors
Officers
Executive Director

From: Joe Budge
Secretary

Date: April 8, 1983

Re: Election Results

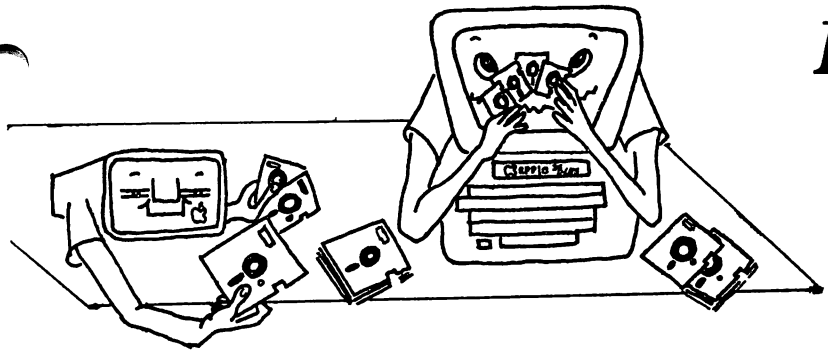
The election agency has just informed me of the outcome of the IAC's 1983 elections for Director. The following candidates received the most votes in their regions and are hence elected to two year terms on the IAC Board of Directors:

Region:	Director:
West	Jim Simpson
North	Tom Wysocki
South	Mike Kramer
East	Neal Lipson

The election agency will be sending me the certified vote counts shortly, which I will forward to all of you.

I would like to extend my congratulations and welcome to the new Directors. I would like, also, to thank all the candidates for the interest they have expressed in the IAC.

910A GEORGE STREET, SANTA CLARA, CALIFORNIA 95050 (408) 727-7652



Dealing With DOS

By

Clark Johnson

In the last two issues, we reviewed the commercial "fast - DOS" packages on the market today. These packages ranged in price from about $20 to $45. Well, as fate would have it, no sooner had I completed these articles than I ran across a free fast- DOS routine. It will be outlined in this article.

The routine was taken from Apple Assembly Line, a monthly magazine/newsletter published by S-C Software. Bob Sander-Cederlof of Dallas is the editor of the Apple Assembly Line, in addition to being the author of the S-C Macro-Assembler. I recommend both his newsletter and the assembler very highly for those interested in assembly language programming.

To make this fast DOS usable by everyone, I will publish it in its Applesoft form. For those interested in the assembly language version, please see the April, 1983 issue of his newsletter.

This fast DOS routine has the necessary basic elements of a fast DOS -- it will speed up the time required for a LOAD, BLOAD, RUN, or BRUN. It does not affect SAVES or BSAVES, and does not improve on the reading or writing of Text files. But by far the greatest disk activity in many applications is that of loading programs to be run, so this DOS patch will be very benefical. Also, a big plus -- it does not remove the INIT function. This is a major advantange, for a reason to be explained subsequently.

Below is the Applesoft listing of the patches that will speed up DOS.

```
10 READ N: IF N = 0 THEN END
20 READ A
30 FOR I = 1 TO N: READ P: POKE A,P: A = A
   + 1: NEXT   40 GOTO 10
100 DATA 44,47721,173,230,181,208,36,173,
    194,181,240,31,173,203,181,72,173,204,
    181,72,173,195,181,141,203,181,173,196,
    181,141,204,181,32,182,176,176,3,76,223,
    188,76,111,179,76,150,172
110 DATA 33,48351,238,228,181,208,3,238,229,
    181,238,196,181,238,204,181,206,194,181,
    208,11,104,141,204,181,104,141,203,181,
    76,150,172,76,135,186
120 DATA 2,44198,105,186
130 DATA 0
```

You have two basic choices on how to use this listing. You could renumber it as necessary and add it to your HELLO program. After the HELLO program is loaded (in the usual slow fashion), then all subsequent programs will be loaded faster by the patch just installed.

Another way to use the listing would be to first boot up on an unaltered DOS (the System Master would be a good choice) and then RUN the above listing. Now the modified DOS will be in the computer's memory. Since the INIT function is preserved, you will now be able to initialize a new disk that will contain the modified DOS. If you remember an earlier article on this subject, you will recall that any modification that has been made to the DOS in memory will be carried to the DOS on the new disk during INIT. Simply load in the desired HELLO program(s) and INIT as many disks as you wish.

A word of warning to Apple IIe owners - it seems that Apple Comnputer put a slightly different version of DOS on the IIe System Master (DOS 3.3e ?). This version has a patch (a bad one, at that) that occupies one of the previously unused "holes" in DOS. The fast DOS routine listed above also occupies that hole in DOS. So if you use this routine, don't use the IIe System Master to create your fast - DOS disks; use a normal DOS 3.3 instead.

There is one other "out" for the IIe owners. Since this change in DOS 3.3 is supposedly a fix for the "APPEND" DOS command, if you don't use the APPEND function, you won't have to worry about the conflict with memory space. Just go ahead and use this DOS modification to your heart's content.

I mentioned earlier that you could use this modification by either merging it into your HELLO program or by INITing new disks after running the listing. But what what if you wanted to use it on some disks that already had a set of existing programs? OK - here comes another modification to the rescue. This one will be credited to the new CALL - A.P.P.L.E. publication ALL ABOUT DOS. Again, I recommend CALL A.P.P.L.E. membership and this particular publication highly for those individuals who are really into computers (Apple computers). Michael Norton wrote in ALL ABOUT DOS a short program to make this DOS copying system more automatic. I won't include that whole program but only the necessary steps to accomplish what he intended.

The trick is to make the COPYA program copy only the first three tracks on a disk (those that contain DOS) instead of the entire disk. Using this technique, you can first create a disk with the modified fast DOS (using INIT as outlined before) and then copy the DOS from the new disk over to all your existing disks. The name of your HELLO program must be the same as on the newly INITed disk or else the boot-up will not properly run the desired program. There are commercial utility programs that will copy the DOS from one disk to another, but this small modification works great if you don't have one of these commercial programs.

First, you need to insert your System Master or other utility disk that has the program COPYA. Then LOAD this program into memory and type in these lines:

```
 75 POKE 770,3: POKE 863,3
250 FT = 1
```

Now type RUN and then insert your new fast DOS disk and the destination disk when the program is ready. The two lines above accomplish two things - 1) instruct COPYA to copy only the first three tracks on the disk and 2) prevent COPYA from formatting your destination disk. It might be wise to try this out on a back-up disk first before using it on one of your originals.

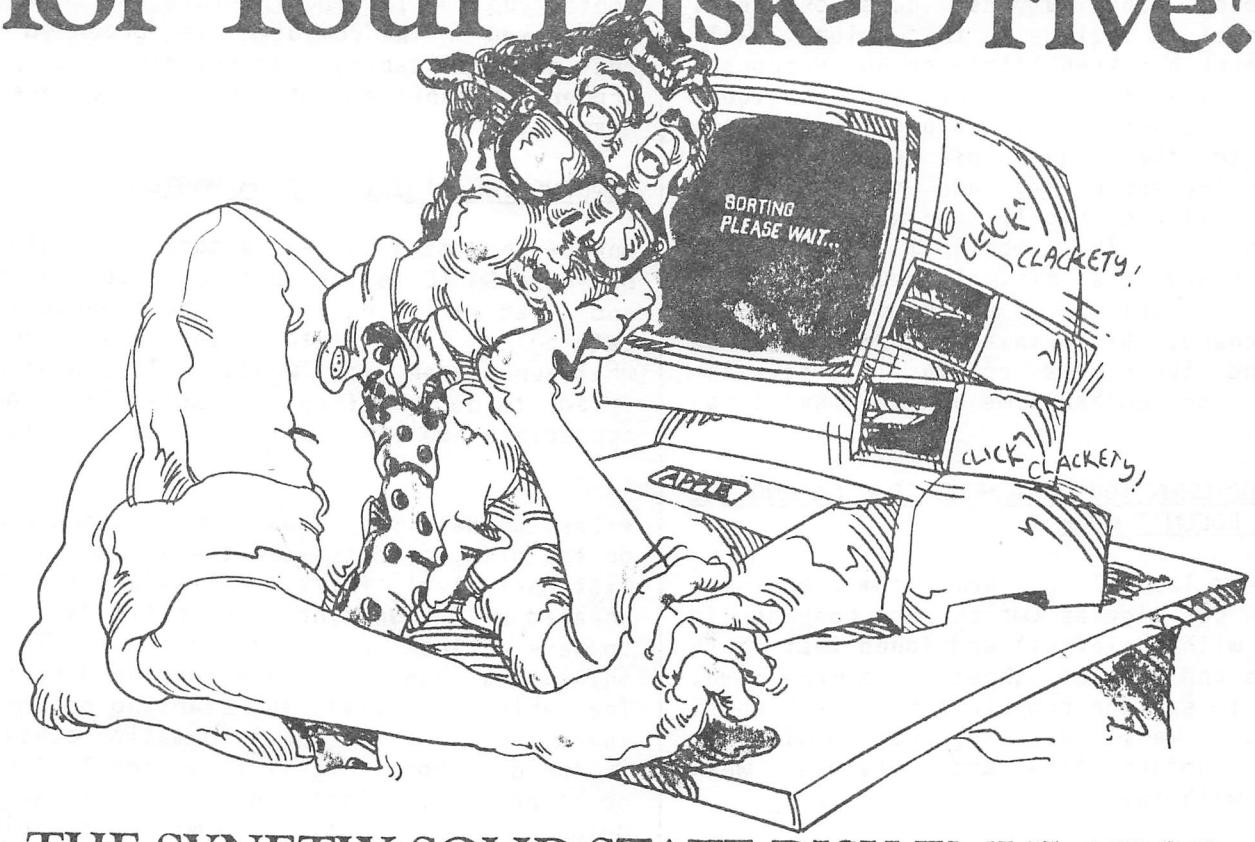
So now you have a good, free fast DOS and all the methods necessary to transfer it to any disks you wish.

IT'S FREE!!!

Did you know that, as an Apple owner, you're entitled to a year's free subscription to SOFTALK magazine? All you have to do is send the serial number on the bottom of your Apple to Softalk, 11021 Magnolia Blvd., North Hollywood, CA, 91601. If you have a friend send it in for you, he/she will get a free back issue of their choice. **You do not get a subscription automatically** when you buy your Apple, so write to Softalk yourself to be sure. It's a good magazine so don't delay.

HAAUG ADVERTISERS

Tired of Waiting for Your Disk-Drive?



THE SYNETIX SOLID STATE DISK EMULATOR FOR THE APPLE, INCREASES SPEED UP TO 1000%.

The Synetix Solid State Disk emulator operates identically to the standard disk drive, but **responds much faster** for programs using disk I/O. The SSD also saves wear on your mechanical drive and diskettes.

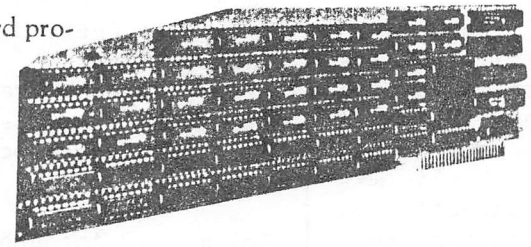
The Solid State Disk Emulator is **compatible with** DOS 3.3, Pascal, SofTech p-Systems IV.O, IV.1, and CP/M operating systems. Call us for a complete list of compatible software and benchmark results on aforementioned systems.

Database management, word processing and file manipulation programs can run up to 1000% faster.

A single board fits any slot and can respond as one or two mechanical disk drives. The SSD requires no external modification and uses only 150 ma power supply per board.

Two Models are now available at new low prices. The **Model 2201 (147K)** single disk is now only **$395.00.** The **Model 2202 (294K)** dual disk is **$695.00.**

The Model 2201 (147K) **is upgradeable** to a Model 2202 (294K) by you or by Synetix.



Contact

FIRST G&E Inc.

9721 KEMPWOOD, NO. 1541

HOUSTON, TEXAS 77080

460-4943

Apple and Apple DOS are trademarks of Apple Computer Inc. CP/M is a trademark of Digital Research

APPLE /// PEELINGS

By
Mike Kramer

A few issues back, I made an appeal for suggestions, contributions, questions, etc. for material to include in this column. Well folks, there has been little or no response. Since (1) the tricks, tips, and techniques which I discuss in this column are directly related to the amount of time I spend exploring the Apple ///, and (2) the time I spend on it is highly variable, the well often runs dry. This month I find myself faced with having a minimal amount to say or printing an article I've submitted to the Apple Orchard. Since this is supposed to be a tips and techniques column, I'll save the articles for another time and see what I can dream up.

WHAT TO DO WHEN YOU GET HOME FROM LA AND THE KEYBOARD DOESN'T WORK

Week before last, I got home from a business trip to Los Angeles (which just happened to coincide with Applefest) and found that both computers and two of three cars were down. Needless to say the computers had the highest priority. We'll save the problem with the //e for another time and discuss what happened with the ///.

The symptoms were a completely dead keyboard except for the power light and the CONTROL RESET function. The first step was to turn the /// on its back, turn the two large screws on either side of the computer a quarter turn each, turn the /// back over, and remove the cover. I then carefully removed the interface cards and set them aside. Next I turned the computer on its back and removed all the screws securing the larger of the two sheet metal plates. I lifted the base and mother board and disconnected and reconnected the ribbon cable leading to the front of the computer . The base plate and motherboard were then reinstalled. Then, with the /// still on its back, I removed the five small screws holding the keyboard cover and disconnected and reconnected the ribbon cable connector, and reinstalled the keyboard cover. It was interesting to note that the conductors on the /// keyboard are solid wire rather than foils on a printed circuit board.

To my gratification, all worked well when I booted up. If you try this, be sure that either you or the computer is grounded and that all the cables (power, disk, etc.) are properly connected before you replace the base.

READING THE DIRECTORY FROM BASIC

One of the many nice features of Apple /// Business BASIC is the ability to open a drive and read the contents of the directory (or catalog) as lines of text and use them in whatever manner you want. If you've ever tried to do this from Applesoft BASIC you'll appreciate how easy it is to list all locked files, or determine how many sectors remain on the disk, or determine the volume name, or list all text files, or list all the files created before or since a specific date (you do have a clock chip, don't you). The one thing you can't do is alter the directory. The table below lists the starting column and character count for the information contained in the directory. The information listed is no secret, but including it will save you doing a lot of counting and will also help me fatten up the column a little.

DIRECTORY INFORMATION

Item	Line	Start	Count
Volume Name	1	2	15
Locked/Unlocked	4+	2	1
File Type	4+	3	6
Storage Blocks	4+	10	5
File Name	4+	16	15
Date File Modified	4+	32	8
Time File Modified	4+	41	5
Date File Created	4+	47	8
Time File Created	4+	56	5
End Of File	4+	63	5
Storage Blocks Free	Last	15	3
Storage Blocks Used	Last	34	3
Total Storage Blocks	Last	54	3

Columns 19 through 26 of the first line of the Directory contain what appears to be a

date followed by "V0" beginning in column 29. I would expect this to be the date the disk was formatted and a volume number. Since I have a clock chip I would also expect the date to be something other than 00/00/00, but that's what it always says! If any of you find out, let me know.

So what good is all this, you say? I've just begun using the information provided by reading the directory within my programs, but feel that the possibilities are endless. I first found out about it from a program written by David Reed that downloads alternate character sets into the Apple Dot Matrix Printer. His code reads the directory and lists only the names of the files that end in the suffix ".DMP", indicating files containing the character sets.

The program in the listing below searches the directory of Drive 2 for files of type "TEXT" or "ASCI", lists their names, and asks for the number of the desired file. Since the name of the file does not have to be typed the chances of getting an error message are reduced. When the file has been selected, it is listed on the console. Possibly parts of the program could be incorporated in your programs.

```
100   DIM filename$(100)
110   TEXT:HOME
120   PRINT USING"79c";"TEXT FILES ON
      DRIVE 2"
130   PREFIX$=".d2/"
140   OPEN#1, PREFIX$
150   j=0
160   ON EOF#1 GOTO 210
170   FOR loop=1 TO 10000
180     INPUT#1;a$
190     IF MID$(a$,3,4)="TEXT" OR MID$
        (a$,3,4)="ASCI" THEN j=j+1:VPO
        S=(j+1)/3+2:HPOS=CONV((CONV&(j
        -1) MOD 3))*28:PRINT"[";j;"] "
        ;MID$(a$,16,15):filename$(j)=L
        EFT$(MID$(a$,16,15),(INSTR(MID
        $(a$,16,15)," ")-1))
200     NEXT loop
210   CLOSE#1
220   WINDOW 0,24 TO 80,24
230   PRINT:INPUT"Enter Number of file
       to list on console:";b$:choice=
      VAL(b$):IF choice<1 OR choice>j
      THEN 230
240   WINDOW 0,3 TO 80,22
250   HOME
260   OPEN#2 AS INPUT,filename$(choice
      )
270   ON EOF#2 GOTO 330
280   INPUT#2;c$
290   PRINT c$
300   GOTO 280
310   CLOSE#2
320   WINDOW 0,24 TO 80,24
330   PRINT:PRINT"Another? (Y/N):";:GE
      T d$
340   IF ASC(d$)>90 THEN d$=CHR$(ASC(d
      $)-32)
350   IF d$<>"N" AND d$<>"Y" THEN 330
360   IF d$="Y" THEN 110
370   TEXT:HOME:END
```

UPPER/LOWER CASE RESPONSE CHECKING

If you've gone through the listing above, you may have wondered what line 340 does. With the upper/lower case capability provided with the Apple //e and Apple /// comes the problem of checking for both upper and lower case responses. It is possible to check for both upper and lower case responses, but that can get messy and certainly isn't elegant. I prefer to convert the response to upper case and then check it. Line 340 checks to see if the ASCII character code for the single key response is greater than 90 (lower case). If it is 32 is subtracted to convert it to upper case before checking for validity.

For multi-character responses, a similar technique is used, but every character in the response has to be checked. Type in the following and try it. By the way, it will work on those Apple][s with lower case, too.

```
100  s$="":p$="abcdefghijklmnopqrstuvwxyz"
110  FOR m=1 TO LEN(p$)
120    IF ASC(MID$(p$,m,1))>90 THEN s$=s$+CHR
       $(ASC(MID$(p$,m,1))-32):GOTO 140
130    s$=s$+MID$(p$,m,1)
140    NEXT m
150  PRINT p$
```

DOWNLOADING CHARACTER SETS INTO THE APPLE DMP

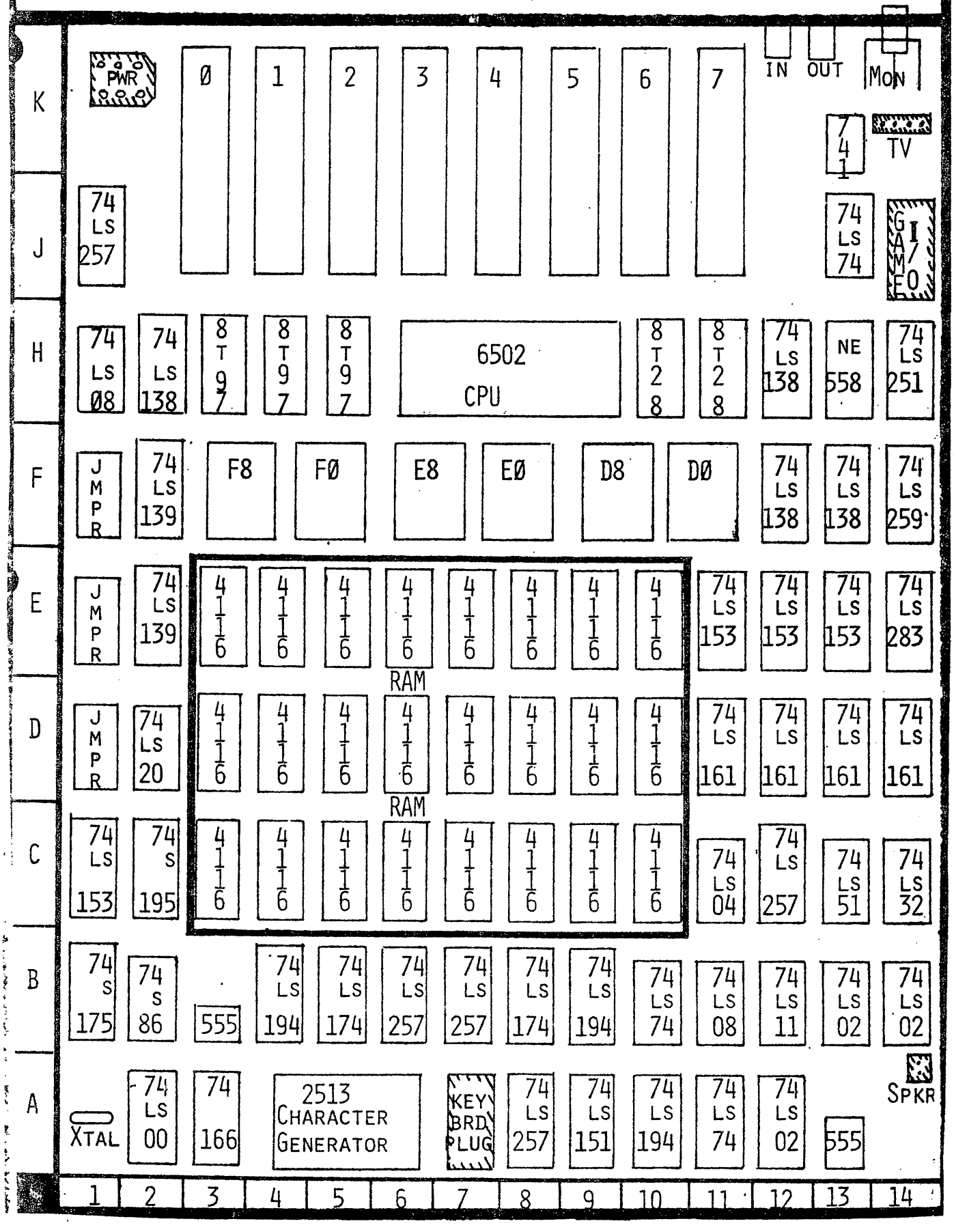
Although not well publicized, the Apple Dot Matrix Printer (DMP) can accept user defined character sets. Unfortunately, the only mention in the "manual" is that ESC ' will select a custom font and an ESC $ will select the current standard font. Apple's David Reed has written an interactive character editor that permits user definition of matching fonts for the Apple /// and the DMP. Imagine typing in script with your word processor and having matching hardcopy. The last time I talked with David, he was considering marketing his program through Apple. I hope he does.

TROUBLESHOOTING GUIDE

BY Dick Peschke
Washington Apple Pi

SYMPTOM	REPLACE THESE IC's ONE AT A TIME
1. TOTALLY DEAD SYSTEM (Power Light is ON)	A2 - 74LS00 B1 - 74S175 B2 - 74S86 B13 - 74LS02 C1 - 74LS153 C2 - 74S195
2. NO RESET or NO REPONSE (note: F14 can also be chip # 9334)	B5 - 74LS174 B6 - 74LS257 B7 - 74LS257 B8 - 74LS174 B11 - 74LS08 C14 - 74LS32 E11 - 74LS153 E12 - 74LS153 E13 - 74LS153 F12 - 74LS138 F13 - 74LS138 F14 - 74LS259 H1 - 74LS08 H3 - 8T97 H4 - 8T97 H5 - 8T97 H8 - 6502 H10 - 8T28 H11 - 8T28 H14 - 74LS251 System ROM F3 - F11 System RAM C, D, E
3. NO VIDEO (However, the Speaker does "BEEP")	A2 - 74LS00 A8 - 74LS257 A9 - 74LS151 A10 - 74LS194 B2 - 74S86 B10 - 74LS74 B13 - 74LS02 C2 - 74S195 C11 - 74LS04 D11 - 74LS161 D12 - 74LS161 D13 - 74LS161 D14 - 74LS161
4. NO TEXT MODE (F14 could be a 9334)	A3 - 74166 A5 - 2513 A8 - 74LS257 A9 - 74LS151 A10 - 74LS194 B2 - 74S86 F13 - 74LS138 F14 - 74LS259
5. HIRES or LORES GRAPHICS problems (F14 could be a 9334)	A8 - 74LS257 A9 - 74LS151 A10 - 74LS194 A11 - 74LS74 B4 - 74LS194 B9 - 74LS194 B10 - 74LS74 C11 - 74LS04 C12 - 74LS257 F14 - 74LS259 H1 - 74LS08 J1 - 74LS257
6. RAM PROBLEMS	A2 - 74LS00 B5 - 74LS174 B8 - 74LS174 C14 - 74LS32 D2 - 74LS20 E2 - 74LS139 E11 - 74LS153 E12 - 74LS153 E13 - 74LS153 F2 - 74LS139 H1 - 74LS08 RAM in Rows C, D, and/or E
7. ROM PROBLEMS	F12 - 74LS138 H1 - 74LS08 ROM - F3 - F11
8. VERTICAL or HORIZONTAL SYNC PROBLEMS	A2 - 74LS00 A12 - 74LS02 B11 - 74LS08 B13 - 74LS02 B14 - 74LS02 C11 - 74LS04 C13 - 74LS51 C14 - 74LS32 D11 - 74LS161 D12 - 74LS161 D13 - 74LS161 D14 - 74LS161
9. GAME PADDLE PROBLEMS	F13 - 74LS138 H13 - 558 H14 - 74LS251
10. CASSETTE LOADING PROBLEMS	F13 - 74LS138 H14 - 744LS251 K12 - 741
11. CASSETTE SAVING PROBLEMS	F13 - 74LS138 K13 - 74LS74
12. SPEAKER PROBLEMS	A12 - 74LS02 B6 - 74LS257 B7 - 74LS257 B10 - 74LS74 C11 - 74LS04 F13 - 74LS138
13. PERIPHERAL CARD IN SLOT WON'T WORK	H2 - 74LS138 H12 - 74LS138

MOTHERBOARD MAP



ONERR GOTO MESSAGE ROUTINE

BY LEE REYNOLDS

Reprinted from: ACES
The Apple Computer Enjoyment Society
Fort Lauderdale and Vicinity
Vol 3 No 3, May/June 1981

Some readers have undoubtedly made use of the ONERR GOTO statement in their Applesoft programming. This capability of the language is very useful for what is called "trapping" errors, instead of having to put up with the alternative, which is to have the program bomb when an error occurs. For those readers who are not acquainted with the use of this statement, here is an example: suppose one of the functions of a program was to DELETE temproary or unwanted files from a floppy disk. Presumably, the name of each file to be DELETEd would be known to the program, either because a set name was programmed in beforehand, or because the program would request the name from the operator. Now, what happens during the execution of this program if (1) you get a read error on the disk, (2) the file was not found in the catalog, or (3) the file was there, but it had been LOCKed? Without an ONERR GOTO routine,the program will "bomb" and you will be returned to Applesoft after the appropriate DOS error message has been displayed. If the delete command is preceeded by the statement:

```
100 ONERR GOTO 25000
```

then an error condition will cause the program to GOTO line 25000 instead of bombing. In this example, line 25000 is the beginning of the "error trapping" routine.

One of the first things your program will want to do at line 25000 is to determine what type of error occurred. This can be determined by examining the contents of memory location 222 with the PEEK function. The error handling routine may also be used to determine the line number at which the error occurred; the guilty line is given by the expression LINE = PEEK (218) + PEEK (219) * 256. (The same error could happen in different parts of a program, and the manner by which it is handled may be dependant on the line number where it occurred.)

The three different cases alluded to in the opening paragraph above can be identified by PEEK (222) having three different values: 8, 6 and 10. An IF statement could test each of these cases and, when true, take appropriate action. For example, an I/O error in case (1), might be handled by simple PRINTing some satement and then STOPping. In case (2) of FILE NOT FOUND, a message could read "File Not On This Disk. Please Insert Proper Disk and Hit Return", then when the operator had followed instructions, the program would go back and try to delete the program again. In case (3) of "File Locked", you might display a message like "File Locked. Do You Want It Deleted? (Y/N)", and then, depending on the Operator's answer, do whatever is necessary to unlock and delete it, or continue.

If some other error than the above thre cases had caused your program to GOTO line 25000, then the value of PEEK (222) would be something other than 6, 8 or 10. The value returned depends on the error encountered. There are two general categories to consider: (1) other DOS errors like "Write Protected", or (2) Applesoft errors, like "*** Syntax Error" or "Redimensioned Array". You can find which values of PEEK (222) correspond to which error conditions by reading page 81 or the Applesoft manual or page 114-122 of the DOS manual.

There are a total of 15 possible DOS errors and 17 Applesoft errors. It is not usually necessary for an "error trapping" routine to have to test for all 32 possibilities, but instead just to display the appropriate message and then STOP. If you had an IF statement that tested for all

32 errors and printed a message similar to the Applesoft or DOS message, that would be a lot of wasted code. It would also be unnecessarily space-consuming, for you would be duplicating the message already in the computer. Therefore, the routine listed below may be of value. It PEEKs at the locations DOS or Applesoft use to store these messages, and then PRINTs the needed message one character at a time (although it prints so fast that the entire message appears at once.)

The starting address for the first Applesoft Error Message is set in the variable ASADDR by line numbe 25010. This value only applies to Applesoft in ROM (an Apple][Plus or an Applesoft Basic card). The same line in the accompanying listing also sets the address for the first DOS error message in the variable DOSADDR. This value is for DOS 3.2 or 3.3 on a 48K machine.

If you want your error-trapping routine to process certain errors in particular ways (not by just PRINTing the message and STOPping), then the appropriate logic could be inserted between lines 25010 and 25100.

```
25000  REM ERROR TRAPPING ROUTINE
25010  ERRNO = PEEK (222) : LINE =
       PEEK (218) + PEEK (219) * 256 :
       ASADDR = 53856 : DOSADDR = 43380
25100  ADDR = ASADDR + ERRNO :  IF
       ERRNO > 0 AND ERRNO < 16 THEN
       GOTO 25130
25110  PRINT CHR$(PEEK(ADR));:  ADDR
       = ADDR + 1 : IF PEEK (ADDR) <
       192 THEN GOTO 25110
25120  PRINT CHR$(PEEK(ADDR)-128) :
       STOP
25130  ADDR = DOSADDR : IF ERRNO = 1
       THEN GOTO 25110
25140  N1=0 : N2 = ERRNO-2 : IF ERRNO
       < 4 THEN N2 = 1
25150  ADDR = ADDR + 1 : IF PEEK
       (ADDR) < 192 THEN GOTO 25150
25160  N1 = N1 + 1 : IF N1 < N2 THEN
       GOTO 25150
25170  ADDR = ADDR + 1 : GOTO 25110
```

YET ANOTHER LOOK AT THE APPLE //e

By Mike Kramer

I recently traded my old Apple][+ on an Apple //e and have had the opportunity to try many of the commercially available programs produced to date to take advantage of its new features. I have been very impressed with the //e and consider it a cost effective alternative to an Apple /// (which I also own). This is particularly so where the more advanced features of the Apple ///, such as the hard disk or printing reports to disk files, are not needed and the user is likely to have trouble with device drivers and boot disks.

The features of the Apple //e which would be most important to the typical user are:

- o Selectric style keyboard with full ASCII character set.
- o Low cost, 80 column, upper/lower case display from Apple.
- o Standard 64K byte RAM, with optional 128K bytes.
- o Compatibility with most Apple][+ software and hardware.
- o Low chip count for increased reliability.

Software products commonly available which take advantage of the 80 column display, the larger memory, and the keyboard, making the //e function much like the Apple /// are summarized below:

- o PFS:Series for //e

  PFS:File, PFS:Report, and PFS:Graph all use the TAB key to jump from item to item, use 80 column display, use cursor control keys, accept lower case. Since files are random access, memory size makes no difference in performance.

o Visicalc //e

Uses 80 column display, accepts lower case, provides 95K bytes of model space with the maximum memory configuration. Uses cursor control keys to move cursor. Does not have variable column width like Apple /// Advanced Visicalc.

o Apple Writer //e

Uses cursor control keys to move cursor, DELETE key to delete characters, TAB key to tab. The most noticeable improvement over Apple Writer][and Apple Writer /// is movement of the cursor through the document without moving the text. Cursor movement is still unpredictable in some situations.

o Quickfile ///

Functions essentially the same as Quickfile ///. Takes advantage of the extended memory to permit very large files.

o Business Graphics //e

Not tested, but should be similar to Apple /// version.

o Palantir Word Processor

Installation disk upgraded to permit installing program to run on the //e as well as the][with a Videx 80 column card or the ///. Uses cursor control keys to control cursor position, the DELETE key to delete, and the TAB key to tab. Requires a Z-80 card and CP/M

To top it off, there are rumors (see previous mention in February/March Dealing With DOS column) that Apple is nearing release of a totally new operating system for the Apple //e. Maybe it will be similar to that on the Apple ///.

On the negative side, the Apple //e 80 column displays are a bit slower than the 40 column display or the Videx 80 column display on an Apple][+ or //e. Since I am not a very fast typist, it is not much of a problem. I do, however, find myself getting impatient when scrolling through a long BASIC program. In spite of this drawback, I find myself frequently recommending the //e over the /// if the ///'s special features are not needed.

MAILING LIST SURVEY

Several of the supporting stores extending a discount to HAAUG members have requested a list of member names, member number, and expiration dates so they can verify membership before giving the discount. Also there might be occasions when it would be to the groupd's benefit to make the mailing list available to reputable groups and companies. Those who DO NOT want to be included in these lists must advise accordingly by filling out the form below and submitting it by August 1.

HAAUG MAILING LIST SURVEY

Name _____

Member Number _____ Date _____

☐ Do not include my name in supporting store list.

☐ Do not include my name and address in mailing lists.

CTI INFORMATION PROCESSING SUPPLIES, INC.

2802 LOUISIANA 526-9666

DATA/WORD PROCESSING
- SUPPLIES
- FURNITURE
- ACCESSORIES

CTI IS A HOUSTON BASED FIRM SPECIALIZING IN FULFILLING ALL YOUR WORD OR DATA PROCESSING NEEDS. **CTI** CAN SIMPLIFY YOUR ORDERING BY PROVIDING QUALITY WP/DP SUPPLIES, FURNITURE & ACCESSORY ITEMS FOR ALL TYPES OF SYSTEMS. ALL PRODUCTS ARE OF THE HIGHEST QUALITY AND FULLY GUARANTEED. FOR COMPATIBILITY QUESTIONS, OR TO ORDER CALL **526-9666.**

DISCOUNT PRICES

DATA PROCESSING SUPPLIES

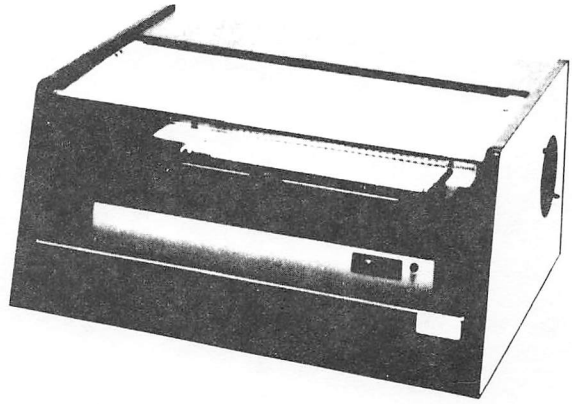
- Verbatim DISKETTES
- CONTROL DATA DISK PACKS
- DISK CARTRIDGES
- PRINTER RIBBONS
- DISKETTE MAGAZINES
- MAG TAPE
- CONTINUOUS PAPER
- PRINTOUT BINDERS

WORD PROCESSING SUPPLIES

- Verbatim DISKETTES
- PRINTER RIBBONS
- PRINTWHEELS—METAL, PLASTIC
- HEAD CLEANING KITS & DISKS
- SPECIAL FORMAT DISKETTES (CPT, LANIER, LEXITRON, MICOM, NBI)
- MAG CARDS

VIKING SOUNDSHIELDS

ELIMINATE UP TO 90% OF OFFICE PRINTER NOISE. OVER 250 MODELS AVAILABLE.

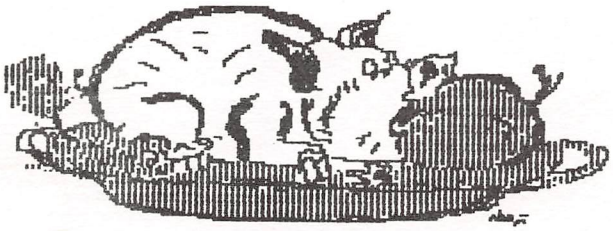


Houston Area Apple Users Group
APPLE BARREL
P.O.Box 610150
Houston, TX 77208

VOLUME 6, NO. 5, JULY 1983

RETURN POSTAGE GUARANTEED
ADDRESS CORRECTION REQUESTED

BULK RATE
U.S.POSTAGE
PAID
HOUSTON, TEXAS
PERMIT 3936



H.A.A.U.G.

697 LAST APPLE BARREL - expired Club Cop

H.A.A.U.G. Hardcopy Library
c/o Robin A. Cox
5401 Chimney Rock #607
Houston, TX 77081