



Listable  
Backupable  
COMPATIBLE WITH  
NORMAL DOS

**Beagle Bros**<sup>TM</sup>  
Micro Software Inc.

# UTILITY CITY

## 21 Useful Apple Utilities on One Disk

by Bert Kersey

### ■ APPLESOFT LIST FORMATTER

A properly-spaced lister with page breaks, each statement on new line, loops indented, if-then's called out.

### ■ COMMAND ZAP

Put INVISIBLE functioning Applesoft commands or hidden statements in your programs for protection.

### ■ Multi-Column CATALOGGER

Catalog to printer or CRT in any number of columns and any page width. File & sector codes optional.

### ■ SCREENWRITER

Format text directly on the screen.

### ■ FILENAME ZAP

Create trick or INVISIBLE file names.

### ■ DOUBLE LOADER

Run any Applesoft file while another program stays in memory.

### ■ LINE SEARCH & REPAIR

Access program lines in memory for repair or "illegal" alteration.

### ■ BIGLINER

Renummer illegally to make lines inaccessible; for program protection.

### ■ RUN COUNTER/DATER

Posts last-run date or current run-number in your programs.

### ■ HEX/DEC/BIN CONVERTER

Convert while program stays intact.

### ■ SORTFILE

Store & alphabetize data on disk. Versatile, listable and customizable.

### ■ AND MORE!

**21 Useful Utilities Total.**

### PLUS—

#### APPLE TIP BOOK #3

Including informative articles and program listings—*Copy Stoppers*, *DOS Trickery*, *Programming the Reset Key...* plus complete easy-to-read Utility City instructions & "how-it-works" documentation.

#### Free PEEKS AND POKES 11 x 17 Wall Chart enclosed

Apple's PEEKS, POKES POINTERS and CALLS on one heavy-duty poster. An indispensable programming tool!

Beagle Bros presents  
**Apple Tip Book #3**

A New Assortment of  
**APPLE II TIPS & TRICKS**



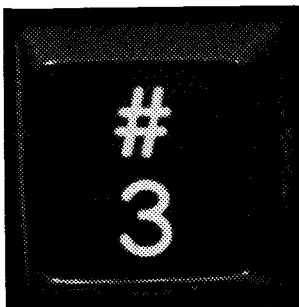
Plus Complete Instructions for Using  
**Utility City**  
Multiple-Utility Disk  
by Bert Kersey

Entire contents Copyright © 1982, Beagle Bros. Micro Software  
4315 Sierra Vista, San Diego, CA 92103

"Apple", "Apple II", and the Apple Logo are all  
registered trade marks of the Apple Computer Company.

**TABLE OF CONTENTS**

TIP BOOK #3. . . . .	2	Int Converter. . . . .	36
<hr/>		Key-Cat. . . . .	41
<b>UTILITY CITY</b>		Kill-Cat. . . . .	42
<b>INSTRUCTIONS.</b> . . . .	27	Line Search. . . . .	30
Address Checker. . . . .	40	Token Chart. . . . .	32
Bfind. . . . .	37	Token Tricks. . . . .	33
Bigliner. . . . .	45	Multi-Cat. . . . .	41
CHR# Poker. . . . .	44	Rem Zap. . . . .	39
Connect. . . . .	38	Run Counter. . . . .	44
Command Zap. . . . .	28	Run Dater. . . . .	44
Double Loader. . . . .	43	Screenwriter. . . . .	34
Filename Zap. . . . .	28	Sortfile. . . . .	37
Free Cash. . . . .	53	Text Dump. . . . .	38
Hex, Dec & Dex. . . . .	45	Xlister. . . . .	46



### PROGRAMMING THE RESET KEY

There are a couple of pokes you can do into DOS that will make RESET act something like a ctrl-C. That makes it trappable by ONERR GOTO, and thereby makes it programmable to do anything you want it to. The program below demonstrates.

WAIT A MINUTE! Be careful. If you make a ~~typo~~ typo, you could really foul things up. Also, if you have less than 48K, run down to the Computer Store with a ninety dollar bill and buy some more memory, because this program won't work with less than 48K. After you type it in, SAVE it, then RUN it. Try to stop it with a ctrl-C or a RESET. You can't. You have to let it run its course. The two pokes at the beginning of the program even program your electric cord so you can't disconnect your Apple (just kidding).

```
20 ONERR GOTO 1000
25 POKE 40286,35: POKE 40287,216
30 X = X + 1
40 PRINT X;" ";: IF X < 400 THEN 30
50 POKE 40286,60: POKE 40287,212: END
1000 PRINT : PRINT "START COUNTING...";X = 0: GOTO 30
```

Line 1000 could just as easily contain a NEW or LIST or RUN or DEL 0,999 or PRINT CHR\$(4);"PR#6" or PRINT CHR\$(4);"CATALOG"...

Here's a set of pokes that will make the reset key boot when it's pressed:

```
FOR X=1011 TO 1015: POKE X,0:NEXT
or *3F3: 00 00 00 00 00
```

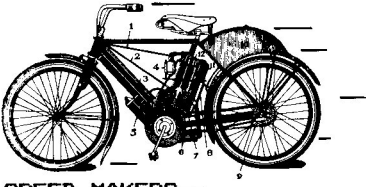
### DOS TRICKERY

Here's a DOS change you can do without Dos Boss. If you peek location 43698 (\$AAB2), you'll find a 132 (\$84). 132 is the high-byte value of CTRL-D, DOS's code character for use in print statements. Now poke in a 192 (POKE 43698,192), the hi-byte value for a @. You have now changed the code character from ctrl-D to @. Now to execute a DOS command like CATALOG from a program, you can type PRINT "@CATALOG".

Just one more way to confuse people and customize DOS to your needs, and it's kind of nice to be able to SEE the code character!

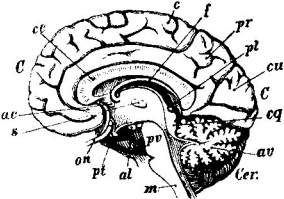
### MOST PROGRAMMERS WANT

more speed in their programs, and most programmers want their programs to occupy less memory. So, most programmers should abide by these tips:



### SPEED MAKERS--

- > Plan ahead to program efficiently. Write your program down in step-by-step plain English before you attack the keyboard.
- > Assign variables to frequently used constants. Don't use "FOR X=12 TO 72". Instead LET T=12 and LET S=72, and use "FOR X=T TO S". The variables assigned values most early in a program seem to execute fastest.
- > Use NEXT instead of NEXT X.
- > Place frequently called lines early in your program.



### MEMORY SAVERS--

- > Renumber your program by 1's starting with Line 1. Applesoft stores line numbers as strings, so the fewer digits in your GOTO's and GOSUB, the less memory used.
- > Change all variable names to one character. Same reason as above.
- > Combine lines so that fewer line numbers reside in memory.
- > DIM your arrays as small as possible, and don't waste the zero-th array.
- > Delete all REMs when you are through with them. Save a remmed version if you want, and use the remless one.
- > Delete useless words-- Change each "THEN GOTO" to a "THEN" or a "GOTO." Don't use "LET" or "END" or file names after "CLOSE".

Load a small test program. Then...

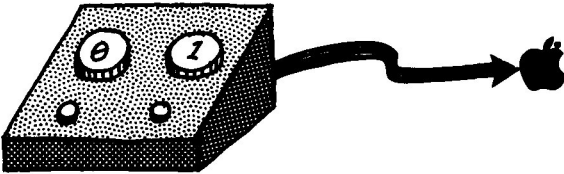
```
PRINT FRE(0)+65536
```

This will tell you how many bytes of memory are available. Now make one of the changes mentioned above and PRINT FRE(0) + 65536 again. The number will be slightly higher.



### PADDLE PANEL

Ever think of disassembling and mounting your paddles on a control panel like so?--



Kind of like electric train transformers; all of the buttons where you want them. Uncle Louie mounted his paddles on both sides of his Apple to make some nifty Raster Blaster flippers.

### SEMI-END

If you want to end a program with no distracting flashing cursor, try one of these methods:  
If you're in hi-res, make your last line--

```
1000 VTAB 1: END
```

The flashing cursor is on the text screen, but hidden by hi-res. This method won't work in lo-res or text, so try--

```
1000 GOTO 1000
```

You could program the cursor to come back when any key is pressed--

```
1000 K=PEEK(-16384): IF K<128 THEN 1000  
1001 POKE -16368,0: END
```

The poke in 1001 prevents the keypress from appearing on the screen.


### TITLETIP

You can start a file name with any character whose ASCII number is above 63. The DOS Manual says you have to start with a letter, but forgets to mention the at-sign, underscore, exponent sign, square brackets, backslash and the lower case equivalents of these characters.

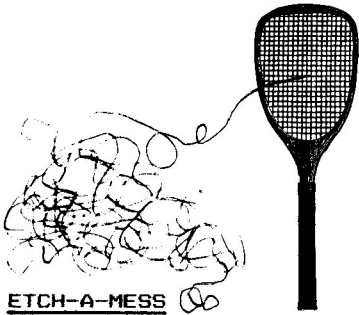
```
10 REM
```

```
=====  
RUN PUN
```

```
=====  
20 A$ = "PPS.!ZPV!TQFMFE!SVD!X"  
22 A$ = A$ + "SPOH/"  
25 L = LEN (A$)  
30 CALL - 1036  
40 CALL - 1036  
50 CALL - 998  
60 CALL - 1008  
70 PRINT LEFT$(A$,1)  
80 CALL - 211  
90 FOR I = 2 TO L  
100 X =ASC( MID$( A$,I,1))  
110 PRINT CHR$( X - 28 + L);  
120 NEXT I
```



YES, DEAR,  
AND AS SOON  
AS YOU TYPE IN  
THE PROGRAM,  
YOU WILL TAKE  
OUT THE TRASH!



Note: If you don't want to type in all of the Tip Book programs, consider getting a copy of our TipDisk (see page 26).

### ETCH-A-MESS

Hey kids! Draw some pictures all over your tv set! To save the hi-res picture, reset out of the program and type "BSAVE MONA LISA,A#2000,L#2000". To save the lo-res one, reset and type "BSAVE WHISTLER'S SISTER,A#400,L#400".

10 REM

### =====

### PADDLE PLOTTER

### (LO-RES)

### =====

```
15 DIM C$(15)
20 PX = 39 / 255:PY = PX:C = 1
25 C$(0) = "BLACK":C$(1) = "MAGENTA":C$(2) = "DARK BLUE":
   C$(3) = "VIOLET":C$(4) = "DARK GREEN":C$(5) = "GREY":
   C$(6) = "MEDIUM BLUE":C$(7) = "LIGHT BLUE"
26 C$(8) = "BROWN":C$(9) = "ORANGE":C$(10) = "GREY":C$(11
   ) = "PINK":C$(12) = "LIGHT GREEN":C$(13) = "YELLOW":
   C$(14) = "AQUA":C$(15) = "WHITE"
30 HOME : GR
40 XNU = PDL (0) * PX
50 YNU = PDL (1) * PY
60 XOLD = XNU:YOLD = YNU
70 XNU = PDL (0) * PX
80 YNU = PDL (1) * PY
100 COLOR=C: PLOT XNU,YNU
110 IF PEEK ( - 16384) > 127 THEN POKE - 16384,0:C =
   C + 1: IF C > 15 THEN C = 0
130 VTAB 21: HTAB 1: PRINT "PADDLES CHANGE POSITION-> X:
   "; INT (XNU);" Y:"; INT (YNU);: CALL - 868
145 VTAB 23: HTAB 1: PRINT " ANY KEY CHANGES COLOR-> ";
   C;":":C$(C);: CALL - 868
150 GOTO 60
```

10 REM

### =====

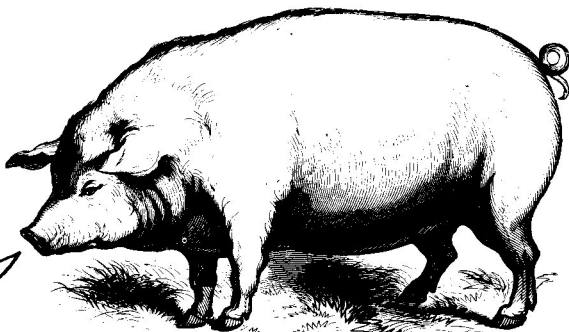
### PADDLE PLOTTER

### (HI-RES)

### =====

```
20 PX = 279 / 255:PY = 191 / 255
30 HOME : HGR
40 XNU = PDL (0) * PX
50 YNU = PDL (1) * PY
60 XOLD = XNU:YOLD = YNU
70 XNU = PDL (0) * PX
80 YNU = PDL (1) * PY
90 HCOLOR=3
100 H PLOT XOLD,YOLD TO XNU,YNU
110 REM IF XOLD < > XNU OR YOLD < > YNU THEN HCOLOR=0:
   H PLOT XOLD,YOLD TO XNU,YNU:REM (OMIT 1ST "REM" TO
   CHANGE)
120 VTAB 22
130 PRINT "X:"; INT (XNU);" "
140 PRINT "Y:"; INT (YNU);" "
150 GOTO 60
```

← LINE 110 OPTION:  
Remove the 1st word,  
"REM" to create a non-  
drawing floating def.



10 HOME:  
FOR Q=1 TO 5:  
  READ OINK:  
POKE 1023+Q,OINK:  
NEXT: DATA 205  
197,207,215,161

### FAT CAT

Here's a little experiment that might teach you something about catalog formatting-- INIT a new disk, DELETE your greeting program, and RUN this:

```
10 PRINT CHR$(4);"MONICO"  
20 FOR X = 0 TO 150  
30 PRINT CHR$(4);"SAVE FILE #";X  
40 NEXT X
```

This will take a while, so go make a sandwich or take a shower (you DO need one). Sooner or later, you will get a DISK FULL error message (around the 105th file name; less in 3.2) not because the disk is full (you've only occupied about 210 sectors), but because there is no more room for file NAMES on the disk.

CATALOG your new disk, and you will see dummy file names FILE #0, FILE #1, FILE #2, etc., up to FILE #104. Now try LOADING a program that doesn't exist. Instead of the expected FILE NOT FOUND message, you will get a DISK FULL message; a protest by DOS that he is stuffed and can't think straight.

A copy of this disk can be handy if you are particular about the order of your file names on a disk. LOAD each program you want to store, and save it in whatever position you want. Say you have a program "STARSHOOTER" that you want stored in the 7th position on the disk. Simply LOAD STARSHOOTER from another disk, and SAVE FILE #7 on your new disk. Now RENAME FILE #7, STARSHOOTER, and you've got it where you want it. Position other file names the same way, then RUN the following to DELETE the extra dummy files:

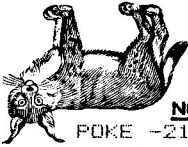
```
10 PRINT CHR$(4);"MONICO"  
20 ONERR GOTO 50  
30 FOR X = 1 TO 104  
40 PRINT CHR$(4);"DELETE FILE #";X  
50 NEXT X: END
```

Another interesting effect is to delete ALL BUT EVERY FIFTEENTH or so of your 105 files. Now you have a slower catalog! File that one under "How'd You Do That?" and show the gang.



### DEL 10

The above statement won't work in a program. DEL needs a range of lines. What WILL work is "DEL 10,10". All programs screech (silently) to a halt with a DEL statement, so make it the last one in your program.



### NO-CAT

POKE -21503,16 prevents a catalog, 17 is the normal value at this address. Subtract 16384 from -21503 for a 32K system.

### CONT-CAT

CATALOG a disk with more than 20 file names. The catalog stops when the screen is full, right? Well, SWITCH DISKS DURING THE PAUSE, and press any key to continue. What happens? Let us know; we were afraid to try it.

### SAME NAME

By using DOS's RENAME function, two or more files can have the same name. Now only the top same-name file will be recognized until you rename it! We like to use the unimaginative but easy-to-spell title, "^", several places in our catalogs as separators.

### PDL ADJUST

To adjust your paddles so they won't send your cursor crashing off the hi-res screen, set variables PX=279/255 and PY=179/255. Then let X=PDL(0)\*PX and Y=PDL(1)\*PY, and PLOT X,Y or whatever (see page 5).

### BACKSCROLL

If you have a flashing cursor near the bottom of the screen, you can scroll text up ONE line with a backspace or up TWO lines with a carriage return. To get the cursor to the bottom you can type "TEXT."

### PRNT SPCS

If you're writing programs for others to read, always use PRINT SPC(10) instead of PRINT " ". The reasons are obvious.

### SPACE SAVER #65535:

These two programs do the same thing. One uses less memory--

```
10 IF A<0 THEN B=60
```

```
20 IF A>=0 THEN B=77
```

or...

```
10 B=77: IF A<0 THEN B=60
```

```
20 (not necessary)
```



### CASSETTE TIP FOR DISKERS

Keep your cassette player within reach in case you lose DOS and are unable to save to disk. Just follow the instructions in your Apple Manual, SAVE to tape, re-boot, and LOAD from tape.



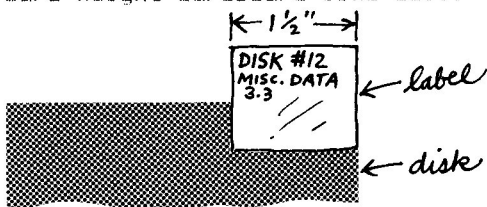
If you've been doing some weirdness on your Apple and want to start a new program, "FP" or "INT FP" (if you have Integer in ROM) is more efficient than "NEW" for clearing memory and resetting pointers. Another way to "delete" most Applesoft programs is to POKE 2049,0: POKE 2050,0. Put these pokes at the end of a program and it will erase itself.

### ELIMINATE THE NEGATIVE

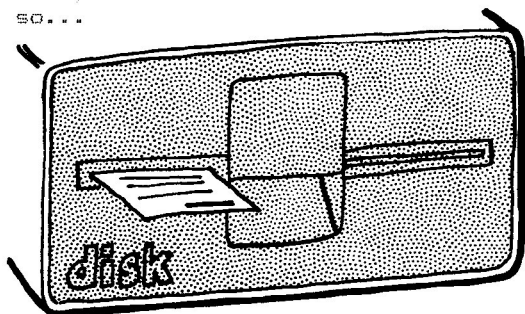
You often see memory locations expressed as negative numbers. Why? Because a bunch of guys sat around in a room one day and decided to subtract 65536 from certain numbers that are larger than 32767. Good grief! So if you see a negative number (as in CALL -958), you can add 65536 to it and learn its true identity (as in CALL 64578).

### VISI-TAGS

You can make nice 1-1/2" x 2" disk labels from business card-weight cardboard like so...



Now you can see what disk is in each of your drives like so...

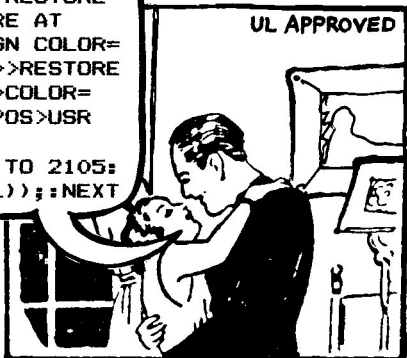




```

1 GOTO 2 AND POS COLOR=OR
  TO AND AT COLOR=-INT
  COLOR=INT USR AT RESTORE
  PLOT-COLOR= ^ -FRE AT
  COLOR=OR AT TO SGN COLOR=
  ABS+AT COLOR=SQR>>RESTORE
  PLOT PLOT SCRN(+>COLOR=
  TO SGN AT COLOR=POS>USR
  NEW PLOT
2 HOME: FOR L=2055 TO 2105:
  PRINT CHR$(PEEK(L));:NEXT

```



## PERPETUAL A<sub>1</sub> Contest!

Uncle Louie's Apple is an old one with only 0.6K, so he really has a problem with some of the software on the market today. He was pretty excited about his upgrade last Christmas-- we got him a 40-column board and a paddle. But he's still lacking in memory (in more ways than one), and needs some Two-Liners to play with.

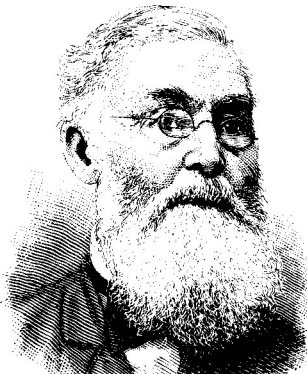
### RULES:

Your program must be written in Applesoft or Integer BASIC and be no longer than two program lines, typable without a ?TOO LONG ERROR message. Decision of Uncle Louie is final. All programs will be judged on...

(a) how impressive the program is when run.

### PRIZES:

1st Prize: Any Beagle Bros disk  
 2nd Prize: Any Beagle Bros disk  
 3rd Prize: Any Beagle Bros disk  
 4th-9th Prizes: Haven't decided yet  
 10th Prize: Uncle Louie

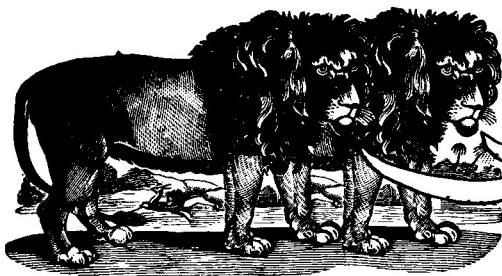


**GIVE UNCLE LOUIE  
A HAND!**

The best 2-Liners will be printed in the next Tip Book and possibly make our TipDisk. Lookit-- It's really a GOOD way to get some free software. Send your Two-Liners in today!

**CHECK OUT SOME OF OUR  
WINNERS — NEXT PAGE.**





Here are some of the winning entries in Uncle Louie's semi-perpetual Two-Liner contest. Give them a try!

=====  
 APPLESOFT  
 2-LINERS  
 =====

```
0 REM DAVE POLIDORI-- EYNON, PA
1 Y = RND (1) * 191: HGR = HCOLOR= 7: POKE - 16302,
  0: FOR I = 1 TO 750: X = X + Y / 2 - 279 * (X > 2
  79): X = X - 279 * (X > 279): Y = Y - X / 4 + 191 *
  (Y < 0): Y = Y + 191 * (Y < 0): HPLLOT X, Y: HPLLOT
  X, 191 - Y: HPLLOT 279 - X, 191 - Y: HPLLOT 279 - X,
  Y: NEXT I: GOTO 1
2 REM LET THIS ONE RUN; LOT'S OF NICE AND DIFFERENT
  DESIGNS!
```

```
0 REM REED DATA-- SAN DIEGO, CA
1 TEXT : HOME : PRINT "YOU HAVE BEEN OFFERED ONE CEN
  T ON THE": PRINT "FIRST DAY OF YOUR NEW JOB. YOU
  R SALARY": PRINT "WILL BE DOUBLED EVERY DAY FOR
  A MONTH.": PRINT : PRINT "] QUESTION: SHOULD YOU
  ACCEPT?";: FLASH : HTAB 2: PRINT " ": NORMAL
2 PRINT : PRINT "DAY", "PAY": PRINT "----", "-----"
  -: POKE 34, 8: SPD = 127 / 31: PAY = .01: FOR DAY =
  1 TO 31: SPEED= 2 * SPD * DAY: PRINT DAY, "$"; PAY
  : PAY = PAY * 2: NEXT DAY: PRINT : PRINT " ANSWE
  R: HOLD OUT FOR MORE.": : VTAB 23
```

```
0 REM DAVID GERSHON-- SEAL BEACH, CA
1 POKE 232, 255: POKE 233, 255: X = INT ( RND (1) * 12
  0) + 1: R = INT ( RND (1) * 40): HGR2 : ROT= R: DRAW
  X AT 90, 90: H = INT ( RND (1) * 8): IF H = 0 OR
  H = 4 THEN H = 1
2 HCOLOR= H: XDRAW X AT 90, 90: GOTO 1: REM TRY THIS
  ONE EVERY NOW AND THEN. IT'S OFTEN DIFFERENT.
```

```
0 REM CHRISTOPHER VOLPE-- TRUMBULL, CT
1 H$ = "303: AD 52 C0 AD 51 C0 A9 22 EA EA EA 20 A
  B FC AD 50 C0 A9 47 3E 00 00 3E 00 00 3E 00 00 3
  E 00 00 3E 00 00 3E 00 00 3E 00 00 3E 0
  0 00 3E 00 00 20 AB FC 4C 06 03 N 303B"
2 FOR I = 1 TO 40: PRINT "THIS IS TEXT PAGE ONE. ";:
  NEXT : FOR C = 1 TO LEN (H$): POKE 511 + C, ASC
  ( MID$ (H$, C, 1)) + 128: NEXT : CALL - 144
```

```

0 REM MARK BACHMAN-- AUSTIN, TX
1 HGR : FOR Z = 0 TO 10 STEP .2: FOR X = 0 TO 10 STEP
  .2: HCOLOR= 3:Y = - 10 * COS (3 * SQR ((X - 5
    ) ^ 2 + (Z - 5) ^ 2)) / 2 + 50: HPLLOT X * 20 + 2
    0 + Z * 3,Y + Z * 10
2 HCOLOR= 0: HPLLOT X * 20 + 20 + Z * 3,Y + Z * 10 +
  1 TO X * 20 + 20 + Z * 3,180: NEXT : NEXT : END
  : REM SLOW AS MOLASSES, BUT WORTH THE WAIT!

```

---

```

0 REM JERRY KRAMER-- PHILADELPHIA, PA
1 HOME : HGR2 : HCOLOR= 3: FOR A = 20 TO 170 STEP 7:
  B = 1.15 * ( SQR (6400 - (A - 95) ^ 2)): FOR C =
  0 TO 6.4 STEP .2:X = B * COS (C) + 139:Y = A -
  B * SIN (C) / 5: IF C = 0 THEN HPLLOT X,Y: NEXT
2 HPLLOT TO X,Y: NEXT : NEXT

```

```

=====
INTEGER BASIC
2-LINER
=====

```

```

0 REM DAVE MADDEN-- LISLE, IL
1 CALL -936: TAB 9: PRINT "HOVERING U.F.O. EFFECT"
  : TAB 13: PRINT "BY DAVE MADDEN"
2 POKE 766,1: POKE 765,16: FOR I=30 TO 1 STEP -1:
  POKE 767,I: CALL -10473: NEXT I: GOTO 2
3 REM REQUIRES PROGRAMMER'S AID CHIP

```

---

```

=====
APPLESOFT AND INTEGER
2-LINER
=====

```

```

0 REM MARC RINGUETTE-- TERRACE, B.C., CANADA
1 PRINT "OPENX": PRINT "WRITEX": PRINT "CALL-151": PRINT
  "2:AD 30 C0 88 D0 04 C6 01 F0 0B CA D0 F6 A6 00
  4C 02 00 60 40 00": PRINT "INT": PRINT "9POKE0,A
  :POKE1,B*42:CALL2:RETURN": PRINT "3A=66:B=4:GOSU
  B9:A=56:B=3:GOSUB9:B=1:GOSUB9:A=83:B=6:GOSUB9:A=
  74:B=2:GOSUB9:A=66:GOSUB9"
2 PRINT "5A=63:GOSUB9:A=56:GOSUB9:A=49:GOSUB9:A=74:B
  =6:GOSUB9:FORJ=1TO60:NEXTJ:A=66:B=4:GOSUB9:A=59:
  B=3:GOSUB9:B=1:GOSUB9": PRINT "7A=56:B=6:GOSUB9:
  A=49:B=2:GOSUB9:A=43:GOSUB9:GOSUB9:A=56:B=6:GOSU
  B9:END": PRINT "RUN": PRINT "CLOSEX": PRINT "EXE
  CX"
3 REM TO ENTER THIS PROGRAM IN ONLY 2 LINES, YOU MUS
  T TYPE IT WITH NO SPACES AND SUBSTITUTE A "?" FO
  R EACH "PRINT". ALSO, ADD A CTRL-D JUST AFTER TH
  E FIRST QUOTE MARK ON "OPENX", "WRITEX" & "CLOSEX".
4 REM YOU MUST HAVE BOTH APPLESOFT AND INTEGER IN YO
  UR APPLE TO RUN THIS PROGRAM. IMPORTANT: SAVE
  BEFORE YOU RUN!

```



### "OH, COME ON!" DEPARTMENT

Our favorite addresses--

Personal Computing Magazine

4 DISK DRIVE

Philadelphia, PA

Micro Magazine

PO BOX 6502

Chelmsford, MA

### QUESTION NUMBER ONE

The most-asked-question over the Beagle Bros phone is "How can I make Key-Cat run automatically when I boot my disk?". The answer is easy. Just BRUN MASTER CREATE. It will ask you what you want to call your greeting program. Enter "Key-Cat" or whatever you want to call it, and that's it! Now Key-Cat will run when you boot! Another way to make it run would be to put a PRINT CHR\$(4); "RUN KEY-CAT" as the last statement in your greeting program.

### CLOSED DEPARTMENT

You don't need to use a file name when you CLOSE a file.

SIDE A:



:SIDE B



### 2-SIDED DISKS?

We have been known to advocate your using both sides of a disk (see The Dos Boss documentation), but you won't find US doing it! Here are two reasons not to--

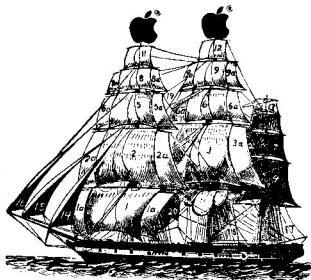
1. A stack of disks is harder to search through with stuff on both sides. Solution: Label both sides' contents on one side of the disk.
2. When you use the "wrong" side of a disk, it spins the "wrong" way, and can mess up the friction pad (or whatever it's called) inside the disk itself. This is potentially a b-i-g problem, and our good friend, Pete the Pessimist, says you can mess up not only the disk, but your DRIVE as well! Pete also refuses to go outside of his closet without a hard hat.

### 2-SIDED DISKS

If you decide to ignore the warnings above, here's a tip we've seen widely used-- Put 3.2 and 3.3 versions of your favorite utilities or whatever on opposite sides of a disk.

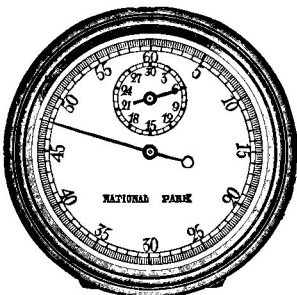
## CRUISING THROUGH APPLESOFT

Remember in the Dos Boss Book where we did a cruise through DOS to find DOS's vocabulary? Well, here's a program that does the same in good old FP!



```
20 START = 53456:FIN = 54116 : REM SUBTRACT 16384 IF 32K
30 TEXT : HOME : INVERSE : PRINT " APPLESOFT COMMANDS: "
   : NORMAL : PRINT "-----"
40 FOR X = START TO FIN:P = PEEK (X)
50 IF P = 7 THEN INVERSE : PRINT "G";: NORMAL
60 PRINT CHR# (P);: IF P < 128 THEN 110
70 N = N + 1
80 IF ERR THEN HTAB 1 + 22 * ((N - INT (N / 2) * 2) =
   0)
90 IF NOT ERR THEN HTAB 1 + 8 * (N - INT (N / 5) * 5)

100 IF PEEK (36) = 0 THEN PRINT
110 IF X = 53854 THEN ERR = 1: INVERSE : FOR I = 1 TO 50
   0: NEXT : PRINT : PRINT " APPLESOFT ERROR MESSAGE: "
   : NORMAL : PRINT "-----"
   : NEXT X
```



## NOW IS THE TIME FOR ALL GOOD

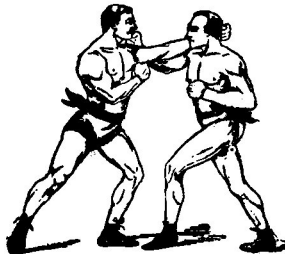
programmers to check their typing speed. Admittedly, this program has a weird clock (each Apple can be different), but you can calibrate it pretty well with a stop watch and a slight adjustment Line 30's variable ADJ. The higher it is, the slower the clock.

```
30 ADJ = 97: REM (LOWER TO SPEED UP CLOCK)
40 GOSUB 110
50 T = T + N:K = PEEK (Q): IF K < HBIT THEN 50
70 POKE QQ,0: IF K < > CR THEN PRINT CHR# (K):C#:C =
   C + (K < > BKSP):GOTO 50
80 CALL - 958:W = INT ((C + 2) / 5):S = INT (T / ADJ)


90 VTAB 22: HTAB 1: CALL - 958: PRINT W;" WORDS / ";S;"
   SECONDS = "; INT (60 * W / S);" WPM": FOR I = 1 TO
   39: PRINT "-";: NEXT
100 VTAB 24: HTAB 1: PRINT " ANY KEY = TRY AGAIN, 0 = Q
   UIT.": HTAB 1: GET A#: IF A# < > "Q" THEN PRINT A
   #:GOTO 40
105 VTAB 23: CALL - 958: END
110 NORMAL : TEXT : HOME : VTAB 22: HTAB 7: PRINT "TYPE
   WHEN YOU HEAR THE BELL.": FOR I = 1 TO 39: PRINT "-"
   : NEXT : VTAB 24: HTAB 10: PRINT "TO STOP, HIT <RET
   URN>.";

120 C# = CHR# (95) + CHR# (32) + CHR# (8) + CHR# (8):
   G# = CHR# (7):T = 0:C = 0
130 Q = - 16384:QQ = - 16368:BKSP = 136:N = 1:HBIT = 12
   8:CR = 141
140 VTAB 1: FOR I = 1 TO 11: HTAB 1:MOD = I - INT (I /
   2) * 2: PRINT CHR# (42 - 10 * (MOD = 0));:GOSUB 16
   0: NEXT : INVERSE : HTAB 1: PRINT G#:" START ";G#;:
   GOSUB 160: NORMAL : HTAB 1: CALL - 868
150 POKE QQ,0: HTAB 1: PRINT C#;: RETURN
160 FOR W = 1 TO 100: NEXT : RETURN
```





### GAMES FOR VISITORS

How many people have you scared away from your computer? All you have to do is demo your new copy of "World War Four" or your dog-eared "Star Bores" disk with its sophisticated grid-search phasors and gamma radiation shields and the non-computer types will quietly head for the exits. If you'd like to keep them around, why not demonstrate your latest version of Tic Tac Toe or one of Beagle Bros Game Pack bonus games like Name Game or Gas Crunch or Poly-Dice. Each Game Pack has at least two games not mentioned in our ads that are interesting and challenging to beginners and non-beginners as well. Not only that, every game pack game is listable AND changeable! Many of our customers have mentioned learning programming techniques from our game packs. 

### ? REMOVER

Nothing rattles my chips as much as seeing a program ask me "TYPE YOUR ANSWER HERE?". The question mark shouldn't be there, and with the Apple, it doesn't have to be. The question mark is produced in this program--

```
10 HOME: PRINT "TYPE YOUR ANSWER HERE"  
20 PRINT " (8 LETTERS OR LESS)"  
30 VTAB 1: HTAB 22  
40 INPUT ANS$
```

To eliminate the ?-mark, make Line 40--

```
40 INPUT "":ANS$
```

or, if you are completely anti-punctuation--

```
40 INPUT "";ANS$
```

### TEXT SCREEN FORMAT

In case you aren't clear about the layout of your text screen, run this program.

```
10 REM
```

### TEXT SCREEN FORMAT

```
50 HOME : NORMAL : FOR I = 1 TO 39:A$ = A$ + CHR*(95):  
NEXT : FOR I = 1 TO 24: VTAB I: HTAB 1: PRINT A$;: NEXT  
: INVERSE : FOR X = 2 TO 23: FOR Y = 5 TO 40 STEP 5:  
VTAB X: HTAB Y: PRINT LEFT*(A$,1);: NEXT : NEXT  
60 FOR Y = 1 TO 24 STEP 23: FOR I = 5 TO 35 + 5 * (Y = 1  
) STEP 5: VTAB Y: HTAB I - (I > 9): PRINT I;: NEXT :  
NEXT : NORMAL : FOR I = 1 TO 24: VTAB I: HTAB 1: PRINT  
I;: NEXT : POKE 2038,52: POKE 2039,48: INVERSE : GOTO  
60
```

### STOP

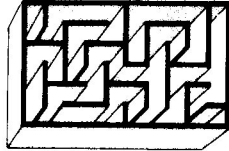
Applesoft's STOP works just like END, but it tells you the line number where your program stopped. Use STOP for de-bugging.

## Game Pack #1\*



- 1. TextTrain:** Race the on-screen clock with your text-format video "freight train." Real-time track switching & coupling simulations, hours of fun!
- 2. Sub Search:** Find & capture the invisible enemy subs on your Apple color graphics scope! Sound-enhanced scanner, tracer & instrument panel!
- 3. Pick-a-Pair:** A colorful Apple party game for all ages and skill levels! Uncover and remember the hidden graphics symbols to score big & win!

## Game Pack #2\*



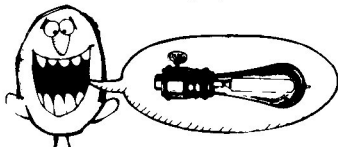
- 1. Wowzo:** Our challenging changeable maze game! Capture targets in a flexible maze, and outmaneuver your opponent before time runs out!
- 2. Elevators:** Keyboard control 4 elevators at one time in your CRT skyscraper. You'll need a computer to solve this one!
- 3. Quick-Draw!** You command two colorful gunmen who shoot it out on your Apple screen!

## Game Pack #3\*



- 1. Magic Pack:** Four mind-bending tricks in one fantastic Magic Show! Only you and your Apple know how to perform these amazing feats!
- 2. Slippery Digits:** A challenging & colorful number-action game for all ages. A great demonstration of your Apple's capabilities!
- 3. Onk!** A nerve-racking sound-enhanced video dice game with unpredictable results and lots of laughs!

## Game Pack #4\*



- 1. Buzzword:** A comical story-creator with endless possibilities. 5 changeable stories in memory plus a fascinating "Create Your Own Story" program!
- 2. Triple Digits:** A thinker's game with numbers. Score in four ways and outfox your opponent!
- 3. Corn Game:** A kids' guessing game involving 3 farm animals and endless supply of corn!

# dos boss™

## DISK COMMAND EDITOR

by Bert Kersey and Jack Cassidy

**Dos Boss** is an extremely versatile, easy-to-use Apple utility package that will customize your disk system and personalize your personal computer! Here are just SOME of Dos Boss's useful features—

**Rename DOS Commands** by simply entering the command you want changed (say "CATALOG") and your new command (say "CAT"). Now "CAT" will catalog your disks. Other changes are just as easy . . .

**Change the "Disk Volume"** heading to anything you want; your name, disk title or code, with or without the Volume Number. Inverse, Flash or Normal!

**"Save-Protect" your programs!** An unauthorized copy attempt will produce a "NOT COPYABLE!" message.

**One-key program selection!** Run programs by pressing only the key indicated on the screen. Instant free-space on disk with one key too!

**Customized Catalogs!** Create multi-column catalogs that fit more file names on the screen. Catalog only the file-types you want (A, I, B and/or T). Omit or alter sector numbers and language codes too!

**Rewrite Error Messages!** "SYNTAX ERROR" can be "TRY AGAIN!" or "NO COMPRENDE!"; "DISK FULL" can be "BURP!" . . . anything you want!

**All of DOS BOSS's change features may be appended to any of your programs, so that anyone using your disks on any Apple (booted or not) will be formatting DOS the way you designed it!**

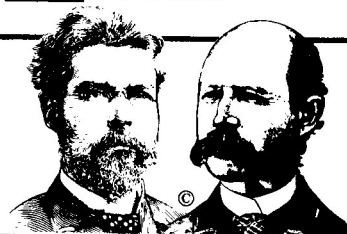
**Plus the DOS BOSS BOOK!** 36 pages of valuable Apple info! An excellent learning tool covering all DOS BOSS features PLUS a new collection of Beagle Bros. Apple tips & tricks; a great companion to our original Beagle Bros. Apple Tip Book (also included free!).

Inside The DOS BOSS BOOK:

- Discover some strange Apple bugs!
- Put Inverse REM Statements in your listings!
- Two-sided Apple disk tips!
- Make your programs un-listable!
- Custom-format your catalogs!
- Change DOS with creative POKING!



**DOS BOSS ©**  
**and The DOS BOSS BOOK**  
3.2 or 3.3 Applesoft



**Beagle Bros™**  
**MICRO SOFTWARE**



Alpha Plot Hi-Res image dumped to printer

## alpha plot

Hi-Res Apple Graphics/Text Utility

by Bert Kenney & Jack Cassidy

48K REQUIRED

**HI-RES DRAWING:** Create hi-res pictures & charts, **appendable to your programs.** Keyboard or Paddle control; Optional Xdraw Cursor (see lines before you draw!); Any color mix or REVERSE (opposite of background); Circles, Boxes & Ellipses, filled or not. Bonus Programs too—**SCRUNCHER** stores hi-res in as little as 1/3 normal disk space. **SHIFTER** transfers any portion of the hi-res screen. Also superimpose hi-res images and convert Hi-Res to Lo-Res & back for fascinating abstracts!

**HI-RES TEXT:** Beautiful upper & lower case with Descenders; color or reverse; Positionable anywhere (NOT restricted by Htabs & Vtabs). Professional looking **PROPORTIONAL SPACING!** Adjustable Type Size, Leading (line spacing) & Kerning (letter spacing). Multi-directional typing; up, down, even backwards!

### PLUS . . . APPLE TIP BOOK NUMBER FOUR!

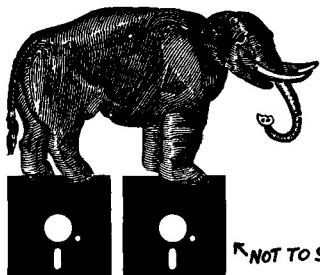
PAGES of tips for making the most of your Apple's advanced graphics capabilities. Alpha instructions plus shape table tips, new tiling & color tricks and fascinating animation experiments!

- ✓ Alpha Plot on Applesoft Disk
- ✓ Beagle Bros Apple Tip Book # 4
- ✓ Apple PEEKS, POKES & POINTERS Chart.

ALPHA PLOT was actually written as a keyboard plotter to draw the above portrait for our animated "Talking Heads" demo, which you've probably seen on our games disks or Dos Boss. Since then, Alpha Plot has expanded feature-wise to become a really complete graphics package with sophisticated TEXT features as well. Our favorite is PROPORTIONALLY-SPACED TYPE, quite a bit more attractive, we think, than the normal Apple text. Also, you can SHIFT ALL OR PART of a picture to a new

screen location. It's kind of like being able to append a subroutine to your programs. For example, you can imprint your logo on all of your hi-res charts or graphs. You can also temporarily or permanently SUPERIMPOSE hi-res pages, handy for comparing images or setting up animation layouts. There's also a program called SCRUNCH that compresses Hi-Res. The picture above occupies only 17 sectors instead of the normal 34. One-third compression is not uncommon; it depends on the amount of detail in the image. HI-LO PLOT is still another program on the Alpha Plot disk. With it, you can make a Lo-Res "impression" of your Hi-Res picture. The list goes on... I'm sure you'll like Alpha Plot.

*Bert Kenney*



### DISK ZAP?

Just how fragile are diskettes? Why not find out? Take a disk that is expendible, and place it an inch or two from a magnet. Will it still boot? Lean it against your color tv screen. Now call the dog over to the Apple...

Disks can be zapped in many ways, but are surprisingly tough when you consider what they consist of. We couldn't resist cutting one open. Can you?

### **S6,D1**

If you have more than two disk drives, label them with a black Dymo-type label ("S6,D1", "S6,D2", etc.) so you can tell at a glance which drive goes to which slot.

While you've got your labeler out, try labeling a few disks. Not bad, right!?

### **PDL - 0**

Also, identify your PADDLES with "-0-" and "-1-" Dymo labels. You can tell which paddle is which with the program below. RUN it and mess with your paddles...

```
10 HOME: BUTTUN = -16287
20 FOR X = 0 TO 1: NORMAL
30 IF PEEK(X+BUTTUN) > 127 THEN INVERSE
40 VTAB 10: HTAB 1 + 20 * X
50 PRINT "PADDLE "; X; " ";; NORMAL
60 PRINT " ";; PDL (X); SPC(2)
70 NEXT X: GOTO 20
```

The word "Paddle" should light up when the appropriate button is pressed, and you should get values from 0 to 255 when you turn the knobs. If you don't, you can clean the contacts inside (carefully) with some electrical contact cleaner. The paddles are undoubtedly the weakest Apple hardware link. Good old Rob & Brent at the Apple Store replaced my original thumb-killer paddle buttons with nice fat red ones they bought from Radio Shack (what?). I highly recommend it for all serious paddlers.

### SPACE ERASERS

Notice the SPC(2) after the PDL value in Line 60 above. This erases trailing digits when the values drop from three-to-two or from two-to-one digit. Try the program with and without the SPC(2) and you'll see.





### GOSUB POKER

In Apple's other language, the late Integer BASIC, you could set line numbers equal to variables (like LET X=SHUFFLE). Then you could say GOSUB SHUFFLE or IF EMPTY THEN SHUFFLE. When you looked at your program, you could tell what was happening without rems. Well, you can't do that in Applesoft, not even with this program; it's only about 30% as good. Type it in carefully; the first two lines have to be EXACTLY as printed here, or the whole thing gets scrambled.

```
10 GOTO 30
20 POKE 2213,48 + G - INT (G / 10) * 10: POKE 2212,48 +
  INT ((G - INT (G / 100) * 100) / 10): POKE 2211,48
  + INT ((G - INT (G / 1000) * 1000) / 100): POKE 2
  210,48 + INT ((G - INT (G / 10000) * 10000) / 1000
  ): POKE 2209,48 + INT (G / 10000): GOSUB 00000:
  RETURN
```

```
30 SHUFFLE = 12345:COUNT = 678
40 G = SHUFFLE: GOSUB 20
45 G = COUNT: GOSUB 20
50 END
678 PRINT "THIS IS COUNT.": RETURN
12345 PRINT "THIS IS SHUFFLE.": RETURN
```

Notice that instead of "GOSUB SHUFFLE", you have to say "G=SHUFFLE: GOSUB 20". When you do, the GOSUB in line 20 is actually CHANGED in the listing.

### PAGE BUG

Turn your Apple OFF & ON and run this program.

```
20 HOME: GR: REM (That's GR, not HGR)
30 FOR X = 0 TO 255
40 HPLOT X,Y
50 PRINT X
60 NEXT
```



You have just drawn all over Page Zero! WEIRD THINGS happen if you don't set the page pointer at 230. It should be set to 32 or 64 for hi-res page 1 or 2. HGR and HGR2 do this for you, but when you turn on your Apple it's set to 0. See your Peeks & Pokes Chart.

↖ RE-BOOT AFTER RUNNING!

### IF INVERSE THEN GOSUB 99

The above statement won't work. What will work is "IF PEEK(50)=63 THEN GOSUB 99". To find out the format of the characters about to be output to the screen, PEEK at location 50 by typing PRINT PEEK(50). A 63 answer means inverse. 255 means normal. 127 is flash. You can also create the format you want by poking in the above numbers (although simple commands do the same thing)--POKE 50,63 will create inverse type and POKE 50,255 will create normalize. POKE 50,127 will flash alphabetical characters. You'll have to add a POKE 243,64 to flash numbers and other characters. By the way, there's NO WAY to inverse or flash lower case.

## BETTER GETTERS

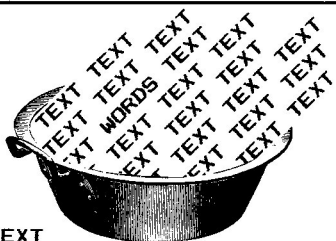
Applesoft's blasted INPUT function won't accept (1) commas, (2) colons, or (3) leading spaces. Try answering an INPUT with a comma or colon in the answer and you get the famous "?EXTRA IGNORED" statement. It doesn't make a lot of sense the first time you see it. Well, let's do something about the problem. Here's are two little inputters. Both feature different types of inputs that accept (1), (2) AND (3)!!

---

```
20 HOME :BKSP$ = CHR$(8):CR$ = CHR$(13):Q$ = CHR$(
34)
30 PRINT "WHAT'S YOUR NAME? ";
40 NAME$ = "": GOTO 50
50 GET LTR$: IF LTR$ = BKSP$ THEN NAME$ = LEFT$(NAME$,
LEN(NAME$) - 1): NORMAL : PRINT LTR$;" ";
60 IF LTR$ = CR$ THEN 100
70 IF LTR$ > = " " THEN NAME$ = NAME$ + LTR$
80 INVERSE : PRINT LTR$;
90 GOTO 50
100 EMAN$ = "": FOR X = LEN(NAME$) TO 1 STEP - 1:EMAN$
= EMAN$ + MID$(NAME$,X,1): NEXT X
110 NORMAL : PRINT : PRINT : PRINT Q$;NAME$;Q$" SPELLED
BACKWARDS": PRINT "IS ";: PRINT Q$;EMAN$;".":Q$: PRINT
120 PRINT "TYPE ANOTHER NAME: ";
130 GOTO 40
```

---

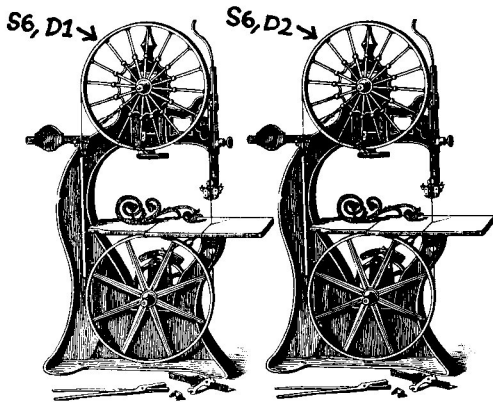
```
20 HOME
30 PRINT "TYPE YOUR NAME ": PRINT "(LAST,FIRST): ";
40 CALL -657
50 NAME$ = "": FOR X = 512 TO 767:C$ = CHR$( PEEK (X)):
IF C$ < > CHR$(141) THEN NAME$ = NAME$ + C$: NEXT
60 L = LEN(NAME$): FOR X = 1 TO L: IF MID$(NAME$,X,1)
< > CHR$(172) THEN NEXT : PRINT "PLEASE SEPARAT
E YOUR NAMES WITH A COMMA.": GOTO 30
70 LAST$ = LEFT$(NAME$,X - 1)
80 FIRST$ = RIGHT$(NAME$,L - X)
90 L = LEN(FIRST$): IF LEFT$(FIRST$,1) = CHR$(160) THEN
FIRST$ = RIGHT$(FIRST$,L - 1): GOTO 90
100 PRINT : PRINT "HELLO, ";FIRST$;" ";LAST$;".": PRINT
"MY NAME IS COMPUTER, APPLE!"
```



## BSAVE TEXT

Sure, you can BSAVE the text screen! Just type BSAVE TEXTTEST,A#400,L#3FF. Any time you want it back, type "BLOAD TEXTTEST". Of course, any command you type to save the screen will appear when you BLOAD it. To prevent this, put the BSAVE statement as the last line in your text-printing program:

```
PRINT CHR$(4);"BSAVE TEXTTEST,A#400,L#3FF"
```



**WHY BUY MORE THAN ONE DISK DRIVE?**

1. You can make faster disk copies.
2. You can use certain software that requires two drives.
3. You will have a backup drive when one goes bad (it happens).
4. You can have immediate access to twice the data.
5. You can get rid of that extra \$500 bill in the dresser drawer.

**MOD**

I learned how to program in Integer BASIC, and have since converted my efforts to Applesoft. The INT feature I miss most is the MOD or remainder function. 3 MOD 2 is 1, 99 MOD 10 IS 9, etc. This is a very useful function that can be simulated in Applesoft with many more keystrokes. Here is a MOD program in Integer--

>LIST

```

10 REM INTEGER
20 FOR X = 1 TO 100
30 IF X MOD 3 = 0 THEN PRINT X
40 NEXT X

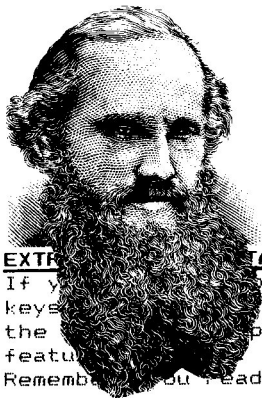
```

To convert to Applesoft, just change Line 30 to--

```

30 IF X - INT(X/3) * 3 = 0 THEN PRINT X

```



**EXTRA IMPORTANT TIP**

If you own one of the newer Apples with the grey keyboard, you will find that underneath the keyboard panel is one of the most important features you will ever use. Try it. You'll be amazed! Remember, you read this here!

```

10 REM
=====
MARQUEE
=====
20 HOME :SP = 50
30 VTAB 1: PRINT "TYPE YOUR COPY BELOW.": PRINT
40 INPUT "":A$: IF LEN (A$) > 40 THEN VTAB 3: CALL -
    958: VTAB 20: PRINT "40 CHARACTERS OR SHORTER, PLEASE.":
    CHR$ (7): GOTO 30
50 HOME : VTAB 3: PRINT A$:L = LEN (A$)
60 FOR I = 3 TO 12: VTAB I - 1: CALL - 868: VTAB I: PRINT
    A$: FOR J = 1 TO 99: NEXT : NEXT
70 VTAB 1: PRINT "(KEYS 0-9 CONTROL SPEED.)":; FOR I = 1
    TO 999: NEXT : HTAB 1: CALL - 868
80 IF L < 40 THEN FOR I = L + 1 TO 40:A$ = A$ + " ": NEXT

90 P = P + 1: IF P > 40 THEN P = 1
100 VTAB 12: HTAB 1: PRINT RIGHT$ (A$,41 - P);: IF P >
    1 THEN PRINT LEFT$ (A$,P - 1)
110 K = PEEK ( - 16384): IF K < 128 THEN 130
120 POKE - 16368,0: IF K > 175 AND K < 186 THEN SP = 30
    *(K - 176)
130 FOR I = 1 TO SP: NEXT : GOTO 90

```

### IN USE?



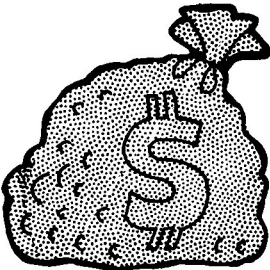
We just decided last Tuesday that IT IS O.K. to type keyboard commands while your drive's "In Use" light is lit as long as you have a prompt and flashing cursor. The Apple drives run for about one second after they are finished doing whatever it is they do in there. No one anywhere knows why.

### A\$=??

Sometimes, strings can contain hidden characters like leading or trailing spaces or control-characters. Setting screen output to inverse, will reveal hidden spaces. Type A\$="FISH ": PRINT A\$. The Apple will hand you a four-letter word apparently without the trailing space. Now try INVERSE: PRINT A\$. You can now see all five characters, including the space. Another way to search for hidden characters is to PRINT A\$; LEN(A\$). If the number is higher than the number of visible characters in the word, you can suspect some hidden control characters or spaces.

### SQUISHY INVERSE?

If your inverse characters are bleeding together and difficult to read, try turning the little black knurled screw near the paddle socket until you see what you want.



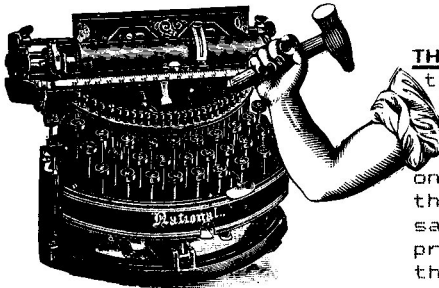
### HOW TO GET RICH:

Market replacement Apple "K" & "D" key caps with little lumps on them. Your fingers can then feel their way to the correct keys. The Apple /// uses this method, and it works.

## LINE TRACER

Any time you want to know what line is operating in a program, put a PRINT PEEK(117) + PEEK(118)\*256. To find out what line an ONERR error was encountered, PRINT PEEK(218) + PEEK(219)\*256. This program demonstrates--

```
13 ONERR GOTO 5000
14 PRINT "THIS IS LINE "; PEEK (117) + 256 * PEEK (118)
   ;".": FOR I = 1 TO 500: NEXT
202 PRINT "THIS IS LINE "; PEEK (117) + 256 * PEEK (118)
   );".": FOR I = 1 TO 500: NEXT
768 PRINT "THIS IS LINE "; PEEK (117) + 256 * PEEK (118)
   );".": FOR I = 1 TO 500: NEXT : GOTO 14
5000 PRINT "PROGRAM STOPPED AT LINE "; PEEK (218) + 256 *
      PEEK (219);".": END
```



## THERE'S MORE THAN ONE WAY TO

type Apple's three keyless characters; the backslash, the left square bracket and the underscore; and here's one of them: Try pushing any three or four keys at the same time. It's difficult to predict what will appear on the screen. If you have a

lower case adaptor, you can often produce lower case letters by pressing two keys. With this in mind, we set out to find our elusive keyless characters, figuring that some combination of other keys would produce what we wanted. Sure enough... Simultaneously press the shift, U & I keys and HOLD THEM DOWN while you type a Y, then an H and then a J. There they are; the three illusive characters, each with a normal character thrown in. Kind of strange, huh? You can make them useable by typing these characters in a listing as described here, and then delete the characters you don't want by normal editing methods.

## 40-COLUMN FIXER

Extensive research by our Extensive Research Department shows that more people need to know this text tip than any other, so pass it around. To align text characters when they are printed on the screen, simply align them IN THE LISTING WHEN YOU TYPE. For example, this--

```
10 PRINT "PROGRESS REPORT FOR H.H.LUMPY
--          MONEY IN      MONEY OUT      BILL
S DUE      $   4.00      $44444.00      $   4
44.00"
```

*←note left margin alignment.*

will appear like this when run--

```
PROGRESS REPORT FOR H.H.LUMPY--
MONEY IN      MONEY OUT      BILLS DUE
$   4.00      $44444.00      $   444.00
```

This alignment will take place only when you ENTER a print statement, not when you list it.



## BETTER INVERSE

Inverse titles look better if you frame them with spaces. Try it--

```
10 HOME: INVERSE: HTAB 17: PRINT "HEADLINE"  
20 VTAB 12:HTAB 16: PRINT " HEADLINE ": NORMAL
```

If you've got room, you can really frame a title--

```
10 HOME: INVERSE: HTAB 18: PRINT SPC(7)  
20 VTAB 2: HTAB 18: PRINT " TITLE "  
30 HTAB 18: PRINT SPC(7): NORMAL
```

# “ ”

## YOU CAN Q\*OTE THIS:

Here's one way to get quote marks in a PRINT statement--

```
10 Q*=CHR$(34)  
20 PRINT "THIS IS ";Q*;"ILLEGAL.";Q*
```

Did you know that semi-colons are often unnecessary in spite of what your Mother told you? Line 20 could be--

```
20 PRINT "THIS IS "Q*"ILLEGAL."Q*
```

## REMLESS REMS

You can leave the word "REM" out of unencountered REM statements, but your Apple will pars the heck out of your remark. Type this in; it runs fine, but lists funny--

```
0 GOTO 2  
1 THIS IS LINE ONE, A SIMPLE STATEMENT, NOTHING MORE.  
2 END
```

## RAM, ROM, PEEK, POKE

You can PEEK at RAM and ROM, but you can only POKE into RAM. That's why you can change DOS, but not Applesoft.

## RND CHART

If you are a random-number nut like me, you'll like watching your Apple draw numbers out of his hat and graph the results.

```
10 HOME : FOR N = 0 TO 9: VTAB 21: HTAB 2 + 4 * N: PRINT  
N: HTAB 1 + 4 * N: PRINT "----": NEXT  
12 X = RND (- 65536 + PEEK (78) + 256 * PEEK (79))  
15 HGR : HCOLOR= 3  
20 C(0) = 3:C(1) = 1:C(2) = 2:C(3) = 5:C(4) = 6: FOR I =  
5 TO 9:C(I) = C(I - 5): NEXT  
30 HCOLOR= 6: FOR Y = 9 TO 159 STEP 10: HPLLOT 0,Y TO 279  
,Y: NEXT : HCOLOR= 3: FOR Y = 9 TO 159 STEP 50: HPLLOT  
0,Y TO 279,Y: NEXT  
50 FOR I = 1 TO 1000000  
60 A = INT ( RND (1) * 10)  
70 X(A) = X(A) + 1  
80 IF X(A) - INT (X(A) / 50) * 50 = 0 THEN INVERSE: FOR  
BUZZ = 1 TO 9:S = PEEK (- 16336): NEXT  
90 IF X(A) > 160 THEN AVG = INT (I / 10): VTAB 24: HTAB  
14: INVERSE: PRINT CHR$( 7);" AVERAGE: ";AVG;" ";  
VTAB 1: HCOLOR= 3: HPLLOT 0,161 - AVG TO 279,161 - A  
VG: HPLLOT 0,159 - AVG TO 279,159 - AVG: NORMAL : END  
100 VTAB 23: HTAB 1 + A * 4 + (X(A) < 100): PRINT X(A):  
NORMAL  
110 HCOLOR= C(A): HPLLOT 3 + A * 28,160 - X(A) TO 15 + A *  
28,160 - X(A)  
120 NEXT
```

10 REM

=====

THE GREAT  
TRACE MYSTERY!

=====

20 H# = "HHHHHHHHH"; I# = "IIIIIIIIIII"

30 HOME : VTAB 10: PRINT I#: FOR I = 10 TO  
19: VTAB I: HTAB 10: PRINT "J": NEXT : VTAB  
20: PRINT "G"; H#

40 COLOR= 8: HLIN 0,9 AT 19: HLIN 0,9 AT 39  
: FOR I = 19 TO 39 STEP 2: PLOT 9,I: NEXT

50 VTAB 5: PRINT "TRACE WITH THE RIGHT ARROW  
& <REPT> KEY.": VTAB 9: END

← (too hard to explain —  
just RUN it.)

### COPY STOPPERS

Everyone wants to know how to protect their disks from copying, so they can quit their jobs and sell their computer programs. Well, quit your job anyway, but sooner or later EVERYTHING will be copyable. But don't fret; the recording industry survives, doesn't it? As our friend R. W. T. Sector once said, "You can't stop all of the people from copying a disk all of the time, but you can SLOW some of the people DOWN some of the time, depending on WHICH COPY PROGRAM they are using". With these wise words in mind, try this trick:

- A. Type "POKE 44033,16".
- B. Put a new disk in your drive and INIT it.

What you have done is change the track of your catalog from 17 (normal) to 16. Other numbers, 3-15, will work too when poked into 44033.

To transfer programs from a normal disk to your new disk:

1. Boot a normal disk.
2. LOAD a program from the disk.
3. Type "POKE 44033,16" (assuming you used 16).
4. Insert your protected disk, the one you INITed in step B above.
5. SAVE the program.
6. Type "POKE 44033,17" (that's normal).
7. Insert your normal disk and continue with step 2.

Remember, the little trick above won't work with all copy programs, and it won't stop anyone who knows much about copy-protection. Copy-protect schemes that work today are EXTREMELY complicated (and they won't work next August).

Another way to slow down copyers is with our program, DOS BOSS. It will change DOS commands and error messages for you, and slow potential copyers down quite a bit.

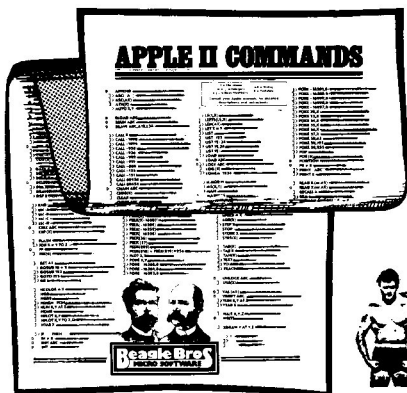
The great race is indeed on between the Copyers and the Copy-Protectors. Progress is being made every day in both camps, and, if you ree-ealy think about it, there can only be one winner-> THE COPYERS.



## Beagle Bros TipDisk

Hey out there! Don't type in all of those programs from the Beagle Bros Tip Books! Here they are--typed, tested and ready to run--ALL OF THE PROGRAM LISTINGS FROM TIP BOOKS 1, 2, 3 AND 4! Many are useful; some are useless; ALL are interesting, listable and copyable. And each program teaches another elusive fact about making your Apple do its thing. No one around here has had time to count them all, but you can figure around four times as many programs as you have here in this Tip Book. The Two-Liners too, from all over the world (and elsewhere). Order now, and you'll have your disk in a few days!

BEAGLE BROS TIPDISK #1: \$20.00



SINCE I GOT MY BEAGLE BROS COMMAND CHART, I'VE ACQUIRED NEW VIM AND VIGOR!

\*An unsolicited endorsement

### A \$2.50 FREEBEE

We've retired the old Apple Command Chart as the head Beagle Bros freebee, and replaced it with our Peeks & Pokes Chart. We've still got plenty of Command Charts in the attic though, really nice too-- a handy alphabetical list of 190 Applesoft, Integer and DOS commands. If you want one, send us \$2.50 + 75 cents postage & handling (don't you hate it when mail order places do that?), and we'll send you a Command Chart right away.

# Utility City Instructions

## AN OVERVIEW

Welcome to Utility City! What you have here is a large collection of utilities that I'm sure you will find useful. I hope you can and will use them for their intended purposes, AND to learn something about what makes your Apple tick.

## THE CATALOG

As you will see in the catalog, many of the U-City programs (the ones that do something TO one of YOUR programs) consist of two or more files, usually a Text file and an Applesoft file. See the instructions for DOUBLE LOADER for an explanation of what's happening here. The file names that end in ".A" are the Applesoft programs that run AFTER you load one of your own programs and EXEC the text file (without the ".A"). If this sounds confusing, forget it. Just read the instructions for each program you use, and...



DO NOT RUN THE PROGRAMS THAT END IN ".A"

You may LOAD them, LIST (most of) them, and change (most of) them, but RUNning them with no other program in memory will possibly foul up a pointer or two and crash around inside of your Apple (nothing serious).

## TRANSFERRING PROGRAMS TO OTHER DISKS

Use the FID program on your System Master to transfer programs. Remember to transfer both the Text file AND the Applesoft file (with the ".A") when both exist. For example, to transfer the BFIND program, BRUN FID, and enter "BFIND=" as the file name. Now, both BFIND and BFIND.A will be copied for you. See your DOS Manual for more FID details.

## "WRITE" PROGRAMS

The Applesoft files that end in ".WRITE" can be LOADED and then RUN to write a Text file to your disk.

## APPLESOFT ONLY, PLEASE

Integer programs will not function with most of the files on this disk (except INT CONVERT, which is an Integer program itself).

## LIMITATIONS

I have tried to think of everything each program CANNOT do, as well as what it can do.

## BUGLIST

Run the Buglist program on the U-City disk to see a list of the any changes, additions, or revelations we have encountered since this book was printed.

Have some fun!

## Filename Zap

WHAT IT DOES: Allows you to rename any file with an invisible file name OR with a trick file name made of any combination of inverse, flash & normal characters.

HOW TO USE IT: <1> SAVE your program on disk with an ordinary file name (no control characters please). <2> UNLOCK the file (or leave it unlocked). <3> RUN FILENAME ZAP. <4> Select from the options shown, and follow the instructions on the screen.

LIMITATIONS: <1> You can't have inverse or flashing characters in a file name that backspaces over itself (erases characters). <2> The only way you can load or run a file name with inverse or flash characters is from the immediate mode-- Catalog and type LOAD or RUN, then trace over the file name in the catalog. <3> The only way you can load or run an invisible file name is under program control. Your invisible file name's visible name (with CHR\$(8)'s) will be displayed on the screen after the conversion is made. WRITE THIS NAME DOWN or remember it, or you might have end up with an inaccessible file. You can temporarily expose the ctrl-H's (backspaces) by running the CTRL-FIND program on the U-City disk.

SUGGESTED USES: <1> Use inverse or flashing file names as decorative touches or accents to your catalogs. Don't overdo the flashing, o.k.? <2> Use invisible file names to disguise what's really on your disk. Some files, like binary files, need not be seen by the user, so why show them in the catalog?

HOW IT WORKS: <1> Inverse and flashing file name characters are recognized by DOS when you trace over them with the cursor. The program simply puts the names you want on the screen so you can trace over them to rename a file. <2> File names are made to erase themselves and/or file codes by inserting backspaces (ctrl-H's or CHR\$(8)'s) into the name. Since DOS adds spaces after every file name to make it 30 characters long, these spaces follow your backspaces and ERASE whatever was on the screen. If you've got good opticals and watch carefully, you can see it happen!

## Command Zap

WHAT IT DOES: Allows you to put invisible commands or rem statements in your programs.

HOW TO USE IT: The best way is to type NEW, and try this test-- Type in a line 100:  
100 PRINT "GOTCHA!": REM^

Type the "^" immediately following the word REM, as shown above. When you LIST 100, a space will be added before the ^. This is important. When you LIST, you have--

```
100 PRINT "GOTCHA!": REM ^
```

Now, from the left of the ^, count each space and text character to the left. You should come up with 21. Now add 21 ^'s plus the statement "IF X = 256 THEN GOSUB 12345" immediately following the word REM (no space after the REM). Now LIST 100, and you should have--

```
100 PRINT "GOTCHA!": REM ^^^^^^
^^^^^^^^^^^^^^^^^IF X = 256 THE
```

```
N GOSUB 12345
```

Now EXEC COMMAND ZAP. When asked "Which Line?", enter 100. When asked "Another Line?", answer N. Now LIST 100, and you should see--

```
100 IF X = 256 THEN GOSUB 12345
```

RUN the program, and it will print--

```
GOTCHA!
```

Your command executes, even though it has been hidden in the listing by a fake command. The fake command you see in the listing could be a rem statement or anything long enough to hide your command (the fake statement can't function since it follows a rem). Make the fake command spaced the same way Applesoft would do it (notice the two spaces between "THEN" and "GOSUB") so everything looks authentic. Now type--

```
SPEED=99: LIST
```

The listing is slowed down so you can see your trickery in action! "SPEED=255" sets the speed to normal after the LIST. You can hide the LINE NUMBER too if you want. Just add five more ^'s to make 26. Maybe try typing--

```
100 PRINT "GOTCHA!": REM ^^^^^^
^^^^^^^^^^^^^^^^^THIS IS
A NICE TITLE, ISN'T IT?
```

EXEC COMMAND ZAP again. Then LIST. The command as well as the line number is now hidden! Longer line numbers will require more ^'s, so count carefully.

SUGGESTED USES: <1> Hide your name in your programs to identify illegal copies. <2> Hide commands that will bomb a program if any tampering is done. <3> Make rem statements that are more predominant and easier to read because they are flush left hiding the line number and the word "rem".

LIMITATIONS: You cannot hide a command longer than two or three words (it must be shorter than one line). CALLS, GOSUBS and GOTOs work just fine.

HOW IT WORKS: The program looks for ^'s (token #94) in the selected line number, and pokes backspaces (ctrl-H's or token #8) in their place. Applesoft's LIST feature takes the backspaces literally and backs up. Your rem statement then overprints the real statement.

## Line Search

**WHAT IT DOES:** Allows you to locate Applesoft program lines in memory.

**HOW TO USE IT:** <1> LOAD your program and write down the line numbers you wish to find. <2> EXEC LINE SEARCH. <3> Follow the instructions on the screen.

**SUGGESTED USES:** Repair garbaged program lines, or make "illegal" changes to Applesoft programs (see "Token Tricks" in this book).

**LIMITATIONS:** Line Search will not work on Integer BASIC programs, since the language structure is entirely different from Applesoft.

**HOW IT WORKS:** Let's take a look at a very small program. Type this in and follow along--

```
1 REM COUNT
2 FOR I = 1 TO 10
100 PRINT I: NEXT I
1000 PRINT "END": END
```

Line Search first finds the start of your program by PEEKing at locations 103 & 104. Try it yourself--

```
PRINT PEEK(103) + PEEK(104)*256
```

This will usually give you a 2049 or hex #801. Now find the END OF PROGRAM with--

```
PRINT PEEK(175) + PEEK(176)*256
```

The longer your program, the higher this number will be. In this case, we get 2097 or hex #831. Now, let's look at the entire program in the monitor, by listing just before and just after the start and end points--

```
CALL -151
*800.832
```

You will now see something like--

```
800- 00 0C 0B 01 00 B2 43 4F
80B- 55 4E 54 00 1B 0B 02 00
810- 81 49 D0 31 C1 31 30 00
818- 21 0B 64 00 BA 49 3A 82
820- 00 2E 0B E8 03 BA 22 45
828- 4E 44 22 3A 80 00 00 00
830- 0A 4E 44
```

Each two-digit hex number above represents part of your program; a line number, character or command token. Let's take a look at each one--

### Location Value

800	00	There is always a zero before the start of a program.
801-802	0C 0B	These two numbers tell us that the NEXT program line starts at location #0B0C (Apple's two-byte hex numbers are always backwards in this strange form).
803-804	01 00	This is the first Line Number, 1 (0001).

805	B2	This is the symbol or TOKEN for the word "REM". Each word Applesoft knows has a token number.
806-80A	43 4F 55 4E 54	These five numbers are the ASCII values for the characters C-O-U-N-T.
80B	00	A zero token means end of line, so Line 1 is finished.
80C-80D	18 08	This is the start of the next program line, the location that was referred to up in 801. The values 18 & 08 tell us that the NEXT program line starts at \$818.
80E-80F	02 00	Line number 2 (\$0002).
810	81	Token for FOR.
811	49	Character "I".
812	D0	Token for "=".
813	31	Character "i".
814	C1	Token for TO.
815-816	31 30	Characters "i" & "O". Applesoft handles numbers in a program as words, spelling them out digit by digit.
817	00	End of line.
818-819	21 08	Next line starts at \$821.
81A-81B	64 00	Line number 100 (\$0064).
81C	BA	Token for PRINT.
81D	49	Character "I".
81E	30	Character ":".
81F	82	Token for NEXT.
820	00	End of line.
821-822	2E 08	Next line starts at \$82E.
823-824	E8 03	Line number 1000 (\$3E8).
825	BA	Token for PRINT.
826	22	Quote mark.
827-829	45 4E 44	Characters "E", "N" & "D".
82A	22	Quote mark.
82B	3A	Character ":".
82C	80	Token for END.
82D	00	End of line.
82E-82F	00 00	Two zeros as at this point mean end of program.
830-832	?? ?? ??	Could be part of an old program.

### TAKING OUT THE GARBAGE

Occasionally, a program will be zapped by mysterious forces and become inoperable. In the listing, you will find illegal line numbers or nonsense statements, like "COLOR= GOSUB HPLLOT". What has probably happened is that ONE BYTE has been changed in a line number or LINE LOCATION NUMBER. If the latter is true, your program will jump to where it's told which is now some random place in memory. The numbers it finds there are interpreted as tokens, and thus the garbage. Use Line Search to find the memory location of the garbaged line and/or the last good line in the program. Check the pair of numbers just before each line number. Each one should point to a location JUST AFTER A 00 (the end of the



current line). When you find an incorrect number, poke in a correct one. For example, to poke a 100 (\$64) into location 2049 (\$801), type POKE 2049,100 OR enter the monitor with a CALL-151, and type 801: 64.

## Applesoft Tokens

Hex	Dec	Chr	Hex	Dec	Chr	Hex	Dec	Token	Hex	Dec	Token
\$00	0	@	\$40	64	@	\$80	128	END	\$C0	192	TAB(
\$01	1	aA	\$41	65	A	\$81	129	FOR	\$C1	193	TO
\$02	2	bB	\$42	66	B	\$82	130	NEXT	\$C2	194	FN
\$03	3	cC	\$43	67	C	\$83	131	DATA	\$C3	195	SFC(
\$04	4	dD	\$44	68	D	\$84	132	INPUT	\$C4	196	THEN
\$05	5	eE	\$45	69	E	\$85	133	DEL	\$C5	197	AT
\$06	6	fF	\$46	70	F	\$86	134	DIM	\$C6	198	NOT
\$07	7	gG	\$47	71	G	\$87	135	READ	\$C7	199	STEP
\$08	8	hH	\$48	72	H	\$88	136	GR	\$C8	200	+
\$09	9	iI	\$49	73	I	\$89	137	TEXT	\$C9	201	-
\$0A	10	jJ	\$4A	74	J	\$8A	138	PR#	\$CA	202	*
\$0B	11	kK	\$4B	75	K	\$8B	139	IN#	\$CB	203	/
\$0C	12	lL	\$4C	76	L	\$8C	140	CALL	\$CC	204	^
\$0D	13	mM	\$4D	77	M	\$8D	141	PLOT	\$CD	205	AND
\$0E	14	nN	\$4E	78	N	\$8E	142	HLIN	\$CE	206	OR
\$0F	15	oO	\$4F	79	O	\$8F	143	VLIN	\$CF	207	>
\$10	16	pP	\$50	80	P	\$90	144	HGR2	\$D0	208	=
\$11	17	qQ	\$51	81	Q	\$91	145	HGR	\$D1	209	<
\$12	18	rR	\$52	82	R	\$92	146	HCOLOR=	\$D2	210	SGN
\$13	19	sS	\$53	83	S	\$93	147	HPL0T	\$D3	211	INT
\$14	20	tT	\$54	84	T	\$94	148	DRAW	\$D4	212	ABS
\$15	21	uU	\$55	85	U	\$95	149	XDRAW	\$D5	213	USR
\$16	22	vV	\$56	86	V	\$96	150	HTAB	\$D6	214	FRE
\$17	23	wW	\$57	87	W	\$97	151	HOME	\$D7	215	SCRN(
\$18	24	xX	\$58	88	X	\$98	152	ROT=	\$D8	216	PDL
\$19	25	yY	\$59	89	Y	\$99	153	SCALE=	\$D9	217	POS
\$1A	26	zZ	\$5A	90	Z	\$9A	154	SHLOAD	\$DA	218	SQR
\$1B	27	[	\$5B	91	[	\$9B	155	TRACE	\$DB	219	RND
\$1C	28	\	\$5C	92	\	\$9C	156	NOTRACE	\$DC	220	LOG
\$1D	29	]	\$5D	93	]	\$9D	157	NORMAL	\$DD	221	EXP
\$1E	30	^	\$5E	94	^	\$9E	158	INVERSE	\$DE	222	COS
\$1F	31	_	\$5F	95	_	\$9F	159	FLASH	\$DF	223	SIN
\$20	32	sp	\$60	96	`	\$A0	160	COLOR=	\$E0	224	TAN
\$21	33	!"	\$61	97	a	\$A1	161	POP	\$E1	225	ATN
\$22	34	#\$	\$62	98	b	\$A2	162	VTAB	\$E2	226	PEEK
\$23	35	%	\$63	99	c	\$A3	163	HIMEM:	\$E3	227	LEN
\$24	36	&	\$64	100	d	\$A4	164	LOMEM:	\$E4	228	STR\$
\$25	37	'	\$65	101	e	\$A5	165	ONERR	\$E5	229	VAL
\$26	38	,"	\$66	102	f	\$A6	166	RESUME	\$E6	230	ASC
\$27	39	%'	\$67	103	g	\$A7	167	RECALL	\$E7	231	CHR\$
\$28	40	(	\$68	104	h	\$A8	168	STORE	\$E8	232	LEFT\$
\$29	41	)	\$69	105	i	\$A9	169	SPEED=	\$E9	233	RIGHT\$
\$2A	42	*	\$6A	106	j	\$AA	170	LET	\$EA	234	MID\$
\$2B	43	+	\$6B	107	k	\$AB	171	GOTO			
\$2C	44	,	\$6C	108	l	\$AC	172	RUN			
\$2D	45	-	\$6D	109	m	\$AD	173	IF			
\$2E	46	.	\$6E	110	n	\$AE	174	RESTORE			
\$2F	47	/	\$6F	111	o	\$AF	175	&			
\$30	48	0	\$70	112	p	\$B0	176	GOSUB			
\$31	49	1	\$71	113	q	\$B1	177	RETURN			
\$32	50	2	\$72	114	r	\$B2	178	REM			
\$33	51	3	\$73	115	s	\$B3	179	STOP			
\$34	52	4	\$74	116	t	\$B4	180	ON			
\$35	53	5	\$75	117	u	\$B5	181	WAIT			
\$36	54	6	\$76	118	v	\$B6	182	LOAD			
\$37	55	7	\$77	119	w	\$B7	183	SAVE			
\$38	56	8	\$78	120	x	\$B8	184	DEF			
\$39	57	9	\$79	121	y	\$B9	185	POKE			
\$3A	58	:	\$7A	122	z	\$BA	186	PRINT			
\$3B	59	;	\$7B	123	(	\$BB	187	CONT			
\$3C	60	<	\$7C	124	)	\$BC	188	LIST			
\$3D	61	=	\$7D	125	~	\$BD	189	CLEAR			
\$3E	62	>	\$7E	126		\$BE	190	GET			
\$3F	63	?	\$7F	127		\$BF	191	NEW			

## TOKEN TRICKS!

### TOKEN LISTERS

These two programs zap their first lines and then list them. Both are nice demos of Applesoft's token system.

```
10 GOTO 256
256 TEXT : POKE 2054,0: FOR I = 0 TO 255: POKE 2051,I: POKE
    2053,I: VTAB PEEK (37) - 1: LIST - 255: NEXT : POKE
    2051,10: POKE 2053,171: POKE 2054,50
```

```
10 REM <-LOOK AT THIS.....
20 HOME :Q = - 16384
30 START = PEEK (103) + PEEK (104) * 256
40 VTAB 10: PRINT "THIS IS LINE 10--"
50 VTAB 15: PRINT "HIT ANY KEY TO END."
60 FOR X = 234 TO 32 STEP - 1
70 POKE START + 4,X
80 VTAB 11
90 CALL - 868: LIST 10
100 FOR I = 1 TO 40
110 IF PEEK (Q) > 127 THEN 130
120 NEXT I,X
130 POKE - 16368,0: LIST
```

### CONTROLAT

Remember how a zero ENDS a program line? You can end one anywhere you want by poking a zero (ctrl-0) into the middle of it. Experiment with listing and running after you have done your poking.

### LIST PREVENTION

POKE 2049,1 will make LIST list your first line repeatedly, because you have told the Apple that the NEXT LINE starts at \$801 instead of \$8-something else.

### DENUMBER

This program will ruin itself, so SAVE before you RUN!

```
10 HOME
20 PRINT "THE PROBLEM WITH THIS PROGRAM IS THAT"
30 PRINT "AFTER YOU RUN IT, IT WILL NEVER BE THE"
40 PRINT "SAME AGAIN."
50 POKE 2051,50: POKE 2057,40: POKE 2148,20: POKE 2167,
    10: LIST
```

### REM EXPANDER

Watch this! A RUN changes the A's<sup>(234 of them please)</sup> to all kinds of rude error messages. A GOTO 100 repairs everything.

```
10 REM AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
    AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
    AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
    AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
    AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
    AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
    AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
20 N = 234
30 FOR I = 2054 TO 2287: N = N + 1: IF N > 251 THEN N = 2
    35
40 POKE I,N: NEXT : LIST : END
100 REM GOTO 100 TO FIX.
110 FOR I = 2054 TO 2287: POKE I, ASC ("A"): NEXT : LIST
    : END
200 REM
=====
REM EXPANDER
=====
```

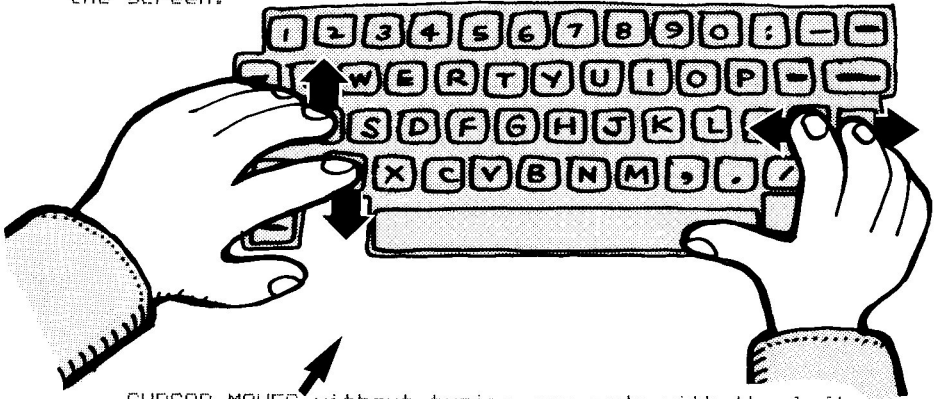
## Screenwriter

WHAT IT DOES: Lets you compose text screen layouts; including inverse, flash & normal type; and centered, flush left & right copy; directly on the screen. These screen layouts may be saved to disk, and loaded into programs whenever you wish, thus saving program memory and programming time.

HOW TO USE IT: Run the program. It will ask if you want to load a file. A file, in this case, is an actual binary image of a text screen layout. If you are just learning, answer "N" and practice typing as you read along. You will now be asked to "stand by", so do as you are told. The Apple is reading the screen at this point to set up a new file.

TYPING MODE is indicated by a flashing "^" or "\_" cursor on the screen. You may type copy as you normally would in any of the text screen's 960 locations (HTAB 1-40 & VTAB 1-24). The carriage return will put the cursor at the left margin on the next line down.

INVERSE, FLASH and NORMAL type is accomplished by pressing CTRL-I, CTRL-F or CTRL-N for the appropriate format. A brief note will appear at the lower left of the screen.



CURSOR MOVES without typing are made with the left & right arrows and the ctrl-A & ctrl-Z keys. To jump rapidly about the screen, press ESC and then the appropriate cursor key. You will need to press ESC and the key for each jump you want to make.

SPECIAL CHARACTERS without keys may be typed too. The left square bracket is typed by pressing CTRL-J (ctrl-shift-M). The underscore is typed with a CTRL-L. The backslash is typed with a CTRL-X.

LOWER CASE is turned off and on with CTRL-S. The cursor will be a flashing "\_" for lower case and a flashing "^" for upper case. If you don't have a lower case chip in your Apple, the lower case mode will only produce miscellaneous characters. Press CTRL-S and you will be typing in upper case again.

CTRL-C will give you access to Screenwriter's many

commands. A letter corresponding to each command will appear at the bottom of the screen in cryptic form (C/G/H/L/M/P/Q/S). To find out what these command letters stand for, type H (for Help). Now you will see an explanation of each letter. Any key will return you to typing mode.

CTRL-C+C will let you clear the screen or a line or column of type. Follow the instructions on the screen. If you answer "N" to the question "O.K.?", your cleared line or column will be restored.

CTRL-C+M will let you move text on the screen. The computer will ask you if you want to move a BLOCK of text or if you want to FORMAT text.

Block movement is achieved by answering "B", and then selecting opposite (diagonal) corners of the block you want moved. Then select the upper-left corner of the new block location, and let the program do its thing.

Formatting means making your text flush left, right or centered on the screen. You may format one line at a time or the entire screen. Follow the screen instructions.

CTRL-C+G will flash a text screen grid on top of your screen copy so you can see where the edges of the screen are. The four lines of text available below hi- or lo-res graphics are indicated by a thicker horizontal line between vtab 20 and 21. Any key will turn the grid off.

CTRL-C+L lets you load a file from disk. If you want to see the catalog at this time, type "CATALOG". Warning: Whatever type was on the screen will be ERASED by a catalog.

CTRL-C+S lets you save text to disk. You cannot access the catalog after a ctrl-S. If you need to see the catalog at this time, SAVE the screen under the name "TEST" or something, select ctrl-L, and type "CATALOG". Later you may rename TEST anything you want.

CTRL-C+P lets you dump the type on the screen to your printer. You will need to turn your printer on and enter its slot number when asked. Inverse and flashing characters will, of course, appear as normal in the printout.

SUGGESTED USES: Screenwriter lets you construct attractive screen layouts without having to use PRINT statements, VTAB, HTAB, INVERSE and the like. If you want, you can dump your composition to your printer and construct your print statements from there. The most efficient thing to do, however, is store your text layouts on disk and then call for them when appropriate from your BASIC program. A simple PRINT CHR\$(4);"BLOAD (name of file)" will do it. You may rename your files, of course. You might want to name them all starting with the character "Q" or "Q" to indicated their type. Or you could end all text screen files with ".T", as in "MENU.T".

## Int Converter

Note: Your Apple must have Integer Basic language capabilities to use this program.

WHAT IT DOES: Converts Integer BASIC programs to Applesoft BASIC without correcting syntax.

HOW TO USE IT: Load Int Converter and insert a disk with your Integer program on it. Then type RUN (return). Using the right arrow key, trace over the load command on the screen, and hit return. Then trace each of the two program lines, 32766 and 32767, hitting return after each. Then trace the "GOTO 32766". A text file will now be created by your Apple; give it some time. Now trace over both the FP and EXEC commands on the screen. An Applesoft file will then be created from the text file. You may now delete the text file and save the Applesoft file that in memory, even though it probably won't work.

You must now correct the Integer syntax to conform to Applesoft. Every "TAB", for instance, must be changed to "HTAB". Every not-equals "#" must be changed to "<>", and so on. The easiest way to do this is with Southwestern Data's "Apple Doc" disk; ask your dealer. It's a great program with many other uses. Another program that we highly recommend is Synergistics' Global Program Line Editor.

Here is a list of the commands you will have to change when converting Integer to Applesoft--

<u>Integer</u>	<u>Applesoft Equivalent</u>
TAB	HTAB
# (not equal)	<>
X MOD Y	X-INT(X/Y)*Y
X/Y	INT(X/Y)
RND(X)	INT(RND(1)*X)
A\$(LEN(A\$)+1)=	A\$=A\$+
A\$(X,X)	MID\$(A\$,X,1)
A\$(4,6)	MID\$(A\$,4,3)
A\$(1,3)	LEFT\$(A\$,3)
GOTO A*100	ON A GOTO 100,200...
GOSUB A*100	ON A GOSUB 100,200...
INPUT"HUH?",W	INPUT "HUH";W
PRINT ASC(A\$)	PRINT ASC(A\$)+128

AND THE BIG DIFFERENCE: In Integer, if an IF statement is NOT true, the rest of the program line IS executed.

SUGGESTED USES: Convert Integer programs to Applesoft.

LIMITATIONS: You can't easily convert Applesoft to Integer, unless you repair all syntax first, a rather buggy job.

HOW IT WORKS: The program writes a text file of the Integer listing. When exec'd from Applesoft, each line enters itself as if typed directly from the keyboard.

## **Bfind**

WHAT IT DOES: Finds the starting location and length of the most recently BLOADED file, and converts each to hex and decimal.

HOW TO USE IT: <1> BLOAD the file in question. <2> EXEC BFIND.

SUGGESTED USES: You need the start address and length of a binary file in order to BSAVE it to another disk. Save a binary file by typing "BSAVE FILE,A\$(s),L\$(l)" where (s) is the start address and (l) is the length. Omit the \$'s if you are using decimal numbers.

LIMITATIONS: You must BLOAD the program in question immediately before you EXEC BFIND.

HOW IT WORKS: BFIND peeks at DOS locations 43616-17 and 43634-35 and converts the values there to hex.

## **Sortfile**

WHAT IT DOES: Sorts alphabetically and stores lists of items on disk.

HOW TO USE IT: Run Sortfile. Hit K any time you want to see all of the options. <N>EW FILE will erase any existing file from memory letting you start a new file. <G>ET lets you load a file from disk into memory. <S>AVE lets you save the file in memory to disk. <A>DD ITEM lets you add one or more items that will be inserted alphabetically into your file. <D>ELETE ITEM lets you remove an item from your file. <R>ENAME ITEM lets you change the name of an item or file. <L>IST prints a full or partial list of your file to printer or CRT. <C>ATALOG lets you view your disk catalog. <Q>UIT lets you quit. Be sure to save your file before you quit or it could be lost. GOTO 9999 will re-enter the program retaining your file. RUN will erase your file.

PRINTER NOTES: If your printer is connected to other than slot #1, change the slot value at the start of the program.

SUGGESTED USES: Use Sortfile for small lists, class rosters, etc., that you want to store and update from time to time. Sortfile may be changed to fit your needs.

LIMITATIONS: You must know the number of an item in order to rename or delete an item. To find the number, simply list all or part of your file.

HOW IT WORKS: Normal DOS text files and string arrays are used. List it and see.

## Connect

WHAT IT DOES: Joins two Applesoft programs together to make one larger program.

HOW TO USE IT: <1> Save your HIGHER-numbered program on the same disk with the "Connect" program. <2> Load your LOWER-numbered program into memory. <3> Type--

A\$="HIGH# FILENAME"

(use the actual file name). <4> EXEC CONNECT, and let the program do its thing. You will get an "Append Complete" message when it is through. If you forgot Step 3, you will still get the message, but it will be a lie!

SUGGESTED USES: Append often used routines (music pokes, sorters, copyright messages, etc.) to your programs.

LIMITATIONS: <1> It's up to you to renumber your programs so that their line numbers don't overlap. Use the program called "RENUMBER" on your Apple System Master Disk. If your numbers do overlap, the appended program will probably be worthless. It is a good idea to keep your two original programs until you are sure all is o.k. <2> You could get a Memory Full error if your programs are too large. Before you load your small-numbered program, it's a good idea to set the Start of Program pointer as low as possible by typing FP.

HOW IT WORKS: <1> The START of Program pointer (for the small-number program) is changed to just beyond the END of the program. <2> The large-numbered program is loaded at the newly specified location. <3> The Start of Program pointer is set back where it was originally.

CONNECT WRITE: To put "Connect" on your own disks, load the program called "Connect Write". Then put each disk in your drive, and RUN.

## Text Dump

WHAT IT DOES: Allows you to transfer your text screen to your printer.

HOW TO USE IT: <1> Append Text Dump to your program with the "Connect" program on the U-City disk. <2> At the point in your program where you would like the text dump to begin (when the screen is full or as you like it), insert a GOTO 63990. <3> Turn on your printer, and run your program.

SUGGESTED USES: Use Text Dump to make printouts of screen formats where a long series of PRINT statements

would take too long to produce.

LIMITATIONS: <1> You cannot have any line larger than 63990 in your program. <2> If your printer is connected to other than slot 1, change Line 63992. <3> The top line on the screen, not the printout, will be messed up. <4> Inverse and Flash will print as Normal type on your printer.

HOW IT WORKS: <1> Each screen location is peeked. <2> Flash, Inverse and Normal values are shuffled to make them all Normal. <3> Each location is printed onto to text line one, and therefore to your printer.

## **Rem Zap, REM FIND and CTRL-FIND**

WHAT THEY DO: Rem Zap allow you to convert (almost) normal rem statements to sometimes visible, sometimes invisible, inverse rem statements in your listings. Rem Find and Ctrl-Find are identical and make the rems invisible.

HOW TO USE THEM: <1> Load an Applesoft program and enter a rem statement. <2> Write down or remember the line number, and EXEC REM ZAP. <3> Enter the line number(s) that have rems you want converted to inverse, and Quit. Now RUN REM FIND (or Ctrl-Find). Your rems will be converted to inverse in your listing. The big drawback is that a PR#0 or RESET will make the rems invisible!

SUGGESTED USES: <1> Fool your friends. <2> Fool your enemies. <3> Hide secret messages in your programs. <4> Make some rems that really stand out.

LIMITATIONS: We've saved the worst for last-- Rem Zap won't zap numbers, spaces and other nearby characters or M's. All M's will appear normal in your listings. G's will zap, but beep when they are invisible. J's will do a line feed.

HOW THEY WORK: In case you haven't guessed, Rem Zap converts your rem statement's characters (ASCII 64-95) to CONTROL-CHARACTERS. Rem Find (and Ctrl-Find) are simply programs that expose ctrl-characters as inverse. After you run Rem Find or Ctrl-Find, you will be able to see ctrl-characters in your catalog listings. Try typing an illegal statement, like "AFLJK" (return). Instead of an audible ?Syntax Error beep you will see an inverse G on the screen. That explains why the G's in your listings beep if you haven't exposed them with Rem Find. M's can't be zapped because ctrl-M is a carriage return. Listings get really weird looking without carriage returns!



## Address Checker

WHAT IT DOES: Address Checker is a learning tool that scans a selected range of addresses and compares and flashes decimal values that have changed since the last scan. If you are a beginner, you can learn quite a bit about how the Apple stores values in memory. Use Address Checker while referring to your Peeks & Pokes Chart.

HOW TO USE IT: Here's a sample run on Page Zero (addresses 0-255, where most of the action is): <1> RUN ADDRESS CHECKER. <2> Select R. You will see about 110 addresses and their peeked values printed on the screen. <3> Select C. Enter the command "COLOR=1" (return). <4> Trace over the "COLOR=1:GOTO 40" statements with the cursor. <5> Watch the screen and notice the numbers. The ones that flash are the values that are DIFFERENT from the previous scan. The value at location 48 is probably flashing if you changed the COLOR value. Location 48 always stores a value equal to 17 times the current COLOR value. Locations 78 and 79 usually always flash because they produce a random number for Applesoft. Select R again and 48 will not flash because its value did not change. You may scan any memory range you wish with an A keypress. Addresses and values will print until the screen is full. No numbers will flash until the second consecutive scan.

It's best not to POKE around on Page Zero unless you know what you're doing, so change the start address to 768 (\$300). Now you can see what a POKE does. Select C and enter "POKE 780,123" (you CAN enter commas!). Do the trace and watch. Only location 780 will be flashing, hopefully with the 123 you poked in.

LIMITATIONS: <1> Certain values (like those at locations 50 and 51) that you change with pokes will be RE-CHANGED when you run the program, so they won't flash. <2> Certain commands like CALL-151 won't work either for obvious reasons. <3> DON'T LOOK AT, AROUND OR NEAR LOCATION 49237! Everything will explode. If you intend to do this anyway, hide ALL of your disks in the refrigerator, and watch the monitor from behind your chair. You have been warned.

SUGGESTED USES: Do some memory snooping. Notice what happens where. Many of the values and locations on the Peeks & Pokes Chart were determined using this program.

HOW IT WORKS: Each location is peeked and put in an array (B(A)). It is then compared with the previous value (A(A)). List it and see.

## Multi-Cat

WHAT IT DOES: Allows you to send catalogs to your printer or CRT in any page width and any number of catalog columns.

HOW TO USE IT: Run Multi-Cat. You will be asked questions regarding printer or screen catalog, page width and number of catalog columns. "Print File Codes?" asks if you want to display the normal lock/unlock code, A/I/B/T file code and sector numbers. Vertical spacing refers to single, double or whatever spacing. The numbers you select (except vertical spacing) will determine the number of characters that will be displayed in your file names. If you have all short file names, this will probably not concern you. The length of file name number will be displayed on the screen. It will flash as a warning if it gets below 3. The only way to change this number is to change one or more of the other options. You can lengthen your file names by selecting fewer catalog columns or a wider page width or by eliminating file codes. The options mentioned here will be automatically requested only on your first catalog. Thereafter, you may still change parameters by selecting option C.

SUGGESTED USES: Make hard copy printouts of your long catalogs. The multiple-column feature allows you to print more file names per page.

LIMITATIONS: <1> A FILE NOT FOUND error will occur if you attempt to access a shortened file name. <2> DO NOT HIT RESET during the catalog. You will mess up DOS and possibly need to re-boot.

HOW IT WORKS: DOS is altered according to your specifications (as in Dos Boss). The number of files is counted, and the catalog is printed. Then DOS is reset to normal. In case you didn't know, a NORMALLY-FORMATTED catalog may be sent to your printer by typing "PR#(slot)" and "CATALOG".

## Key-Cat

WHAT IT DOES: Lets you select and run programs with one keystroke from your catalog, and quickly find the amount of space free on a disk.

HOW TO USE IT: Run Key-Cat. You will see a normal catalog. If you have more than 17 file names on your disk, the catalog will pause as usual when the screen is full. Press any key to continue, OR, if you see the file name you want on the screen, PRESS <RETURN>, and selection letters will appear to the left of each file name. Now press the key you want, and the appropriate file will RUN, EXEC or BRUN, depending on whether it is

a BASIC, TEXT or BINARY file (see Limitations below).

To quit Key-Cat, select a Z. To find sectors used (SEC USE) and sectors free (SEC FRE), hit a # (shift-3) when the key letters are on the screen.

SUGGESTED USES: Use Key-Cat as your Hello program on your disks. Save it under the file name "N". Then you can simply type RUNN (2 N's) to run it. To make Key-Cat (or N) run when you boot your disk, you must change the name of your greeting program to Key-Cat (or N). To do this, BRUN MASTER CREATE from the System Master disk (or BRUN UPDATE 3.2.1 if you have DOS 3.2). Instructions are in your DOS Manual.

LIMITATIONS: <1> If you select a binary file such as a hi-res picture, that is meant to be BLOADED, not BRUN, it will crash when you select it with Key-Cat. <2> Likewise, all text files aren't meant to be EXEC'd, and those that aren't will crash. <3> If you select a program with flashing, inverse or control characters in its file name, Key-Cat won't recognize it, and the error will cause Key-Cat to re-catalog.

HOW IT WORKS: Key-Cat actually reads the screen, letter-for-letter (not the disk), to get the selected file name.

ADDITIONAL FEATURE: If you would like normal file codes to appear along with the selection letters, change Line 20 in Key-Cat to--

20 COL=1

Actually, COL, the column in which the letter will appear, can be 1, 3, 4 or 7. COL=0 will cause no sector codes to show and change B, A, I & T to \*, J, > & ". If these special codes are inverse, the file is locked.

## Kill-Cat

WHAT IT DOES: Makes ctrl-C (or any key you choose) make a clean break in a partial catalog. Any other key will continue the catalog as usual.

HOW TO USE IT: Run Kill-Cat. You may change Line 20 if you want a key other than ctrl-C to stop the catalog. For example, enter KEY#=CHR\$(13) for <return> or KEY#=" " for space or KEY#="S" if you want S to stop your catalog. For some reason, ctrl-C and no other key will let you kill a catalog even BEFORE the screen is full.

SUGGESTED USES: Append Kill-Cat to your greeting programs so you can utilize its feature each time you boot your system.

LIMITATIONS: None that we know of. The small "wait for a

keypress" routine is placed at location \$BA69 (for 48K) in DOS. As far as anyone around here knows, it is unused.

HOW IT WORKS: Kill-Cat tells DOS at location \$AE63 to jump to location \$BA69 and check for a ctrl-C press instead of jumping to location \$FDOC to see if ANY KEY is pressed as it normally does.

## Double Loader

Read this carefully and take notes. Then rewrite it so it makes sense to you.

WHAT IT DOES: Creates a TEXT FILE X which, when EXEC'd, will RUN Applesoft Program P, while any other Applesoft Program stays in memory.

HOW TO USE IT: <1> Run Double Loader. <2> Enter the names of the two programs, A and P, described above. Stand by while text file X is written. <3> Add the following line of commands to the end of Program P (with a line number; be sure the program GETS TO these commands):

```
LOC=768: POKE 103, PEEK(LOC): POKE 104, PEEK(LOC+1):  
POKE 175, PEEK(LOC+2): POKE 176, PEEK(LOC+3): END.
```

<4> With any Applesoft program in memory, EXEC (file X Name), and Program P will run. When it ends, the pokes in step 3 will be executed and your original program will be restored.

SUGGESTED USES: Use Double Loader for utilities or subroutines you often use while programming, but don't want appended to your programs. Without Double Loader, you would have to SAVE your program, RUN your utility, then RELOAD your program. Many of the programs on Utility City use this "double load" method.

LIMITATIONS: If you reset, or ctrl-C out of a program, the end pokes will not be executed, and your original program will not be accessible, although it is still there in your Apple's memory. If this happens, do a GOTO to your end line (see #3 above).

HOW IT WORKS: <1> The original Start of Program pointers are stored in locations 768-769 (\$300-301). <2> The original End of Program pointers are stored in locations 770-771. <3> The END of Program pointers are poked into the START of Program locations 103-104. <4> A zero is poked into the new Start of Program location minus 1, and Program P is loaded. In other words, Program P is sort of APPENDED to your original program. When you are finished with Program P, the pointers are restored to their original values.

## Run Counter & Run Dater

WHAT THEY DO: Run Counter updates a number in your program listing, reports on the screen how many times it has been run, and re-saves your program with the updated run-number prior to program execution. Run Dater works similarly; it tells you the last date on which your program was run, asks for today's date, and re-saves itself with the new date, before execution.

HOW TO USE THEM: <1> Using the Connect program on the U-City disk, append your Applesoft program to either Run Counter or Run Dater. (Load Run Counter. Type A\$="PROGRAM NAME". EXEC CONNECT.) Your program must have NO LINE numbered smaller than 17, so you may have to do a little re-numbering BEFORE you append. DO NOT RENUMBER RUN COUNTER OR RUN DATER, or they may bomb. <2> Add your program's name between the quote marks in Line 11. <3> You may want to change Line 10 in either program. It contains the counter that is updated each time you run either program. Make these changes carefully in the format shown. You may remove the REM statement. DO NOT add any program line numbered smaller than 10. <4> Save your program. It will now run normally after it re-saves itself with its updated information.

SUGGESTED USES: Use both programs to keep track of what use your disk library is getting.

## CHR\$ Poker

WHAT IT DOES: Finds and returns a number for each text screen location in memory and each printable character, ASCII 0-255.

HOW TO USE IT: Enter the vtab and htab values for the screen location and the character you want.

SUGGESTED USES: Use it to determine the values for poking any character (including the three "illegal" characters) directly to the screen. The one screen location you CANNOT PRINT TO is VTAB 24, HTAB40. Doing so will scroll the screen up one line. Instead, you can POKE 2039, (value of the character you want printed). No scroll!

LIMITATIONS: It is usually more convenient to type "FLASH: VTAB 1: HTAB 1: PRINT "A": NORMAL" than to "POKE 1024,65" even though the second command takes less space.

HOW IT WORKS: The program actually prints the character you select in its three modes on the left of the screen, and then PEEKs those three locations to determine the character values.

## Bigliner

**WHAT IT DOES:** Renumbers Applesoft program lines to 65535 to make them inaccessible to most people.

**HOW TO USE IT:** <1> Load your program. <2> EXEC BIGLINER. The instructions are on the screen. Bigliner will either up-number your highest-numbered program lines, one at a time to 65535 (all the same number), or it will down-number all lines above 63999 so you can access them.

**SUGGESTED USES:** <1> Protect your copyright notice or special routines from unauthorized tampering by putting them at the end of your program. <2> Perform unauthorized tampering. (Note: Don't tamper with a Beagle Bros program, or your Apple will ~~explode~~ *explode itself*, turn into a TRS-80.

**LIMITATIONS:** Bigliner will not renumber GOSUBS, GOTOS and THEN line numbers within your statements. To play it safe, use Bigliner for REMs only, or number your lines accordingly before you use Bigliner. All "big lines" will have the same number, 65535, the biggest legitimate line number possible on the Apple.

**HOW IT WORKS:** Bigliner attaches itself to the end of your program, finds each line number in memory (as in Line Search on the U-City disk), looks at each line number, and pokes in new values as requested.

## Hex, Dec & Dex

**WHAT THEY DO:** The HEX program converts decimal numbers 0-65535 to hexadecimal numbers \$0000-\$FFFF. DEC converts numbers \$0000-\$FFFF to positive and negative decimal numbers. DEX does both of the above plus conversions to binary. All three programs may be used with your current program staying intact.

**HOW TO USE THEM:** <1> To use HEX, type A=1234 (the decimal number you want converted). Then EXEC HEX. <2> To use DEC, type A#="12AB" (the hex number you want converted). Then EXEC DEC. <3> To use DEX, EXEC DEX.

**SUGGESTED USES:** Use DEC or HEX for quick conversions of one or two numbers. Use DEX when you have a lot of converting to do, or if you want binary conversions, or if you want to convert from negative decimal.

**LIMITATIONS:** You cannot convert from binary to hex or decimal with DEX (it won't do Roman numerals either).

**HOW THEY WORK:** DEC pokes the value A into locations 768 & 769, enters the monitor, and lists the two locations. HEX simply multiplies each "digit" of A# by  $16^0$ ,  $16^1$ ,  $16^2$  &  $16^3$ . DEX does all of the above.

## **Xlister**

**WHAT IT DOES:** Produces an easy-to-read re-formatted listing of any Applesoft program. Each program statement will be listed on a new line. Each statement following a FOR will be indented until a NEXT is encountered. All statements within a line following an IF-THEN statement will be marked with an asterisk (\*). Only one space instead of two will appear between tokens (such as THEN and GOTO). Printer page breaks will occur for printed listings.

**HOW TO USE IT:** Load your program and then EXEC XLISTER. Select options from the questions on the screen. Pressing <return>-only will select the displayed default value. If you are using a printer, be sure it is TURNED ON, or the program will hang up. Xlister may be stopped at any time with a CTRL-C. Do not use <reset> or your program will be lost.

**PRINTER NOTES:** If your printer is connected to other than SLOT #1, load XLISTER.A, and change Line 50. Also, Xlister assumes a page length of 66 lines with page-break margins of 3 lines top and bottom. Line 60 contains these values; change them if you want. SAVE XLISTER.A after you have made the changes.

**SUGGESTED USES:** Use Xlister on programs you want to de-debug or list in a nicely-formatted printout. Having each statement on a new line makes a tremendous difference in following the logical sequence of a program. The "\*" callout after IF-THEN statements shows you at a glance the statements which will be encountered only if the IF statement is true.

**LIMITATIONS:** <1> Xlister is much slower than the normal list routine, so it is far more practical with a printer than without. <2> Loops are indented two spaces to the right or left for every FOR or NEXT encountered. If your program does not follow a normal sequence (if you have a NEXT before a FOR or a FOR at the top of a program and its NEXT at the end), you will see some illogical looking indentation, not too objectionable. Perhaps you have found a bug?

**HOW IT WORKS:** Xlister is appended to your program and then looks at every byte between your program's start and end points, analyzes it, and prints it in its interpreted form. See Line Search, How it Works, for a discussion of Applesoft's token system.



**Beagle Bros**<sup>TM</sup>  
**INDOOR SPORTS**