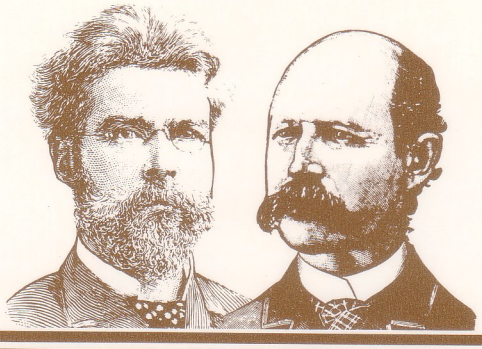


UNLOCKED and UNPROTECTED.
Compatible with
- **Apple IIe®**
- **Apple II+***
- **Apple II***

*APPLE II and II+ require RAM Card. (Apple IIe does not.)

Beagle Bros™
Micro Software Inc.

BEAGLE BASIC

APPLESOFT® ENHANCEMENT UTILITY
BY MARK SIMONSEN

REQUIRES APPLE® IIe / *OR* II or II+ WITH RAM CARD (Language Card)

BEAGLE BASIC puts Applesoft into RAM, letting you customize and enhance it—

RENAME ANY COMMAND or Error Message. For program protection, encryption, or even foreign translation. Even the *new* commands that follow are re-nameable—

ELSE: Common in many programming languages, but missing from Applesoft until now. ELSE follows If-Then's, like this—

IF X=2 THEN PRINT "Yes": ELSE PRINT "No"

HSCRN: Read any Hi-Res point to determine its current color (off/on status). Fast and simple— Useful in hi-res animation.

SWAP: Normally, to swap two variables' values, you need a third variable and an extra split-second. **SWAP X,Y** exchanges variables, arrays or strings in *one* quick step.

PAGE, MIX, RESL and **MODE:** No more awkward graphics screen-switch Pokes. For example, just type "**PAGE1**" or "**PAGE2**" to switch pages, instead of POKE -16300,0 or POKE -16299,0 (*never* look up those hard-to-remember Pokes again!).

TONE: Beagle Basic's "**TONE P, L**" command plays a note of Pitch P, Length L. It's simple— No Pokes or Calls are necessary.

TXT2: Allows Text Page 2 to act exactly like Page 1, for printing, listing, catalogging, etc. Switching text pages opens up all kinds of new programming possibilities.

ZERO MEMORY COST: Beagle Basic's new commands cost you NO memory since they use only the space formerly occupied by obsolete Applesoft Cassette commands (like SHLOAD, STORE, etc.).

GOTO & GOSUB may now be followed by variables. Use understandable commands such as "GOTO COUNTER" or "GOSUB SONG" (where COUNTER and SONG have been assigned line-number values). GOTO expressions too (as in GOTO X+100).

ESCAPE-CURSOR: Normally, you can't tell if you are in Escape Mode (moving the cursor). With Beagle Basic, ESC temporarily changes the normal cursor to a flashing "+".

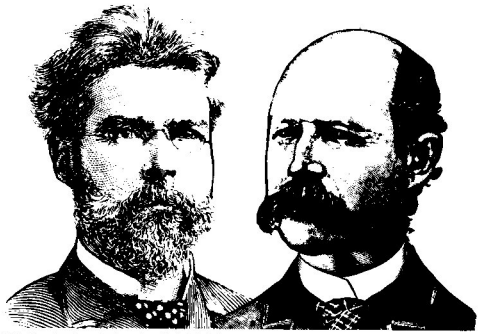
BETTER BEEP: Apple's ctrl-G Bell can be turned into one of 256 more pleasing sounds. Even Apple error messages *sound* better!

PLUS: New one-word commands to clear text, replacing Call-958 and Call-868. New command to **SCROLL TEXT DOWN,** not normally possible before Beagle Basic. And...

INVERSE REM STATEMENTS Makes remarks appear as bold stand-out headlines in your Applesoft listings.

PLUS Apple Tip Book #6 Included in this manual—A new collection of entertaining and useful Apple programming tips. Plus an 11x17 **PEEKS & POKES** Chart.

INCLUDES FREE PEEKS, POKES & POINTERS WALL CHART AND APPLE TIP BOOK #6



BEAGLE BASIC was written by Beagle Bros programmer **MARK SIMONSEN.** Beagle Basic is compatible with any APPLE IIe computer (*or* any APPLE II or APPLE II+ with a RAM card or "language card").

Beagle Basic and **APPLE TIP BOOK #6** (back half of this book) are published by BEAGLE BROS Micro Software Inc., 4315 Sierra Vista, San Diego, Ca 92103.

The Beagle Basic instructions and Apple Tip Book #6 were written by **MARK SIMONSEN** and **BERT KERSEY.**

Table of Contents

BEAGLE BASIC 2-20
How to Use Beagle Basic 2-3
Enhancing Applesoft 4
  Command Editor 12
  Error Message Editor 12
  List Formatter 5
  Adding Basic Commands 6
    ELSE, SWAP, TONE 6-7
    HSCRN, SCRLDN, TXT2, G2, CLRKEY 8-9
    Text & Bell, Hi-Res and Cursor Commands 10
  Other Features 11
    ESCape Cursor, Modify Bell 11
    Modify GOTO & GOSUB 11
  Save and Quit 11
Beagle Basic and the Token System 18
Program Notes 19-20
Index 40
APPLE TIP BOOK #6 21-39
Free Cash 49

Beagle Basic Instructions

What is BASIC?

Technically, "BASIC" should always be capitalized. We'd rather not conform.

BASIC is your Apple's native language. Apple IIe's and II+'s are equipped with *Applesoft Basic*, while ancient (pre-1980) Apple II's came with *Integer Basic* hardware. Applesoft, by far the most common "resident" language, determines exactly which words (or "commands") your Apple understands. It also determines what actions your Apple will take when each command or combination of commands is typed. Since Applesoft is stored in ROM (**R**ead-**O**nly **M**emory), it is normally unchangeable. The abilities and vocabulary given to your Apple by Applesoft are set in concrete. Permanent. Period.

What is BEAGLE BASIC?

Beagle Basic is a set of programs that let you enhance Applesoft and use new commands and features in your programs. It also lets you rename standard commands and error messages, for whatever reason you might have. Beagle Basic gets around Applesoft's "permanency" by moving it from ROM into RAM (**R**andom-**A**ccess **M**emory) or *changeable* memory.

A NEW BASIC IN AUXILIARY MEMORY

Don't do this now. Just read about it.

When you boot a System Master disk on an Apple IIe (or an Apple II or II+ with a RAM card), the Hello program loads the "other" BASIC (Applesoft or Integer, whichever one is *not* built into your Apple) into memory "above" Apple's normal 48K. On an Apple IIe or II+, you would then have *Applesoft* in ROM and *Integer* in RAM. You can switch between these two languages with DOS's FP and INT commands.

Well, don't tell anyone where you read this, but Integer Basic is a worthless dinosaur! *Nobody* writes programs in Integer anymore. Yeah, sure, it's faster than Applesoft and it has a MOD function, but look at all it's missing. (See Appendix M of the Applesoft II Reference Manual or Appendix L of the equivalent IIe Manual.)

There are two versions of NEWBASIC on the Beagle Basic disk— "NEWBASIC IIE" and "NEWBASIC II/II+". Don't worry; the appropriate version will be accessed according to the type of Apple you are using.

So, let's not load *Integer* as our second language in RAM. Instead, let's put an enhanced version of *Applesoft,* called "NEWBASIC", there. Newbasic is loaded into memory by the "NEWBASIC LOADER" program. Your first job is to run the "CREATE NEWBASIC" program, select the Applesoft enhancements that you want and save them on disk as "Newbasic". Just like you can use cousin *DOS Boss* to change DOS, you can alter Apple's *Applesoft* vocabulary, and modify the way the Apple *behaves* when commands are typed. Beagle Basic gives you options of twenty brand new commands and two modified commands.

If you program in Applesoft, you're going to have some fun with Beagle Basic.

How to Use Beagle Basic

To make a back-up, use the COPYA program on your System Master disk.

Your original Beagle Basic disk is a normal DOS 3.3 disk. You can boot it, catalog it, save to it... and *ruin* it— Make a back-up copy please.

Beagle Basic requires 64K or more.

Beagle Basic requires your Apple to have at least 64K of memory. If you have an Apple IIe, you are set. If you have an Apple II or II+ (normally 48K), it must be equipped with extra memory in the form of a RAM card (or "language card").

USING NEWBASIC

To use your version of Newbasic (see top of this page), you must run the program called "NEWBASIC LOADER", which will load NEWBASIC into your Apple's auxiliary memory (our way of saying "RAM card" or "language card" or "bank-switched memory").

ENHANCING APPLESOFT

Applesoft (or Newbasic) may be changed to perform many different functions. You must first *enhance* Applesoft, then *save* the enhanced version on disk under the name "NEWBASIC". Here's what you do:

1. Boot the Beagle Basic disk.
2. Run the program called "CREATE NEWBASIC".
3. Use CREATE NEWBASIC to customize Applesoft by changing its vocabulary and adding new commands.
4. Save your enhanced version of Applesoft on disk under the name "NEWBASIC".

ENHANCING APPLESOFT WITH BEAGLE BASIC

CREATE NEWBASIC

CREATE NEWBASIC is the program on the Beagle Basic disk that lets you customize Applesoft and/or save it onto disk under the name "NEWBASIC". Boot the Beagle Basic disk and select (**C**), the CREATE NEWBASIC option, from the menu, or (**Q**), Quit, and type:

RUN CREATE NEWBASIC (return)

Create Newbasic's first job is to load some form of Applesoft (modified or unmodified) into your Apple's auxiliary memory. Here's what happens when Create Newbasic runs:

A. Create Newbasic will look for some form of Applesoft (normal or modified) in your Apple's auxiliary memory. If found, step D is next.

B. Create Newbasic will look on the disk for a modified Applesoft file called "NEWBASIC" to be loaded into auxiliary memory.
If found, step D is next.

C. Create Newbasic will give up and transfer Applesoft from its normal ROM location into auxiliary memory.

D. Create Newbasic will now display its Main Menu.

The Main Menu

Create Newbasic's Main Menu offers you five Applesoft change options on the screen, plus Save and Quit. Details for each option are printed on the page shown:

(**C**) COMMAND EDITOR ..... page 13
(**E**) ERROR MESSAGE EDITOR ..... page 15

(**L**) LIST FORMATTER ..... page 5
(**B**) ADD NEW BASIC COMMANDS ..... page 6
(**O**) OTHER FEATURES ..... page 11

(**S**) SAVE NEWBASIC ..... page 11
(**Q**) QUIT ..... page 11

You can usually return to this Menu from Create Newbasic's various modes by pressing the **ESC** key or **1** (see keychart notes on page 12).

MAIN MENU OPTION **C**

Command Editor

Every Applesoft command may be renamed to almost any word you want. See "Changing Commands and Error Messages" on page 12 for instructions.

MAIN MENU OPTION **E**

Error Message Editor

Applesoft error messages may be reworded too, for clarity or just for fun. See "Changing Commands and Error Messages" on page 12.

MAIN MENU OPTION **L**

List Formatter

Selecting option **L** from the Main Menu presents you with three choices for changing Applesoft's rigid list format. Typing the letter (**A**, **B** or **C**) turns the corresponding feature On or Off ("YES" or "NO" printed in inverse) in your Apple's memory.

NOTE: If you want to change something in an inverse remark, don't cursor-trace over it. You must instead *type over* it, or re-type it completely.

Inverse Rems can be a problem in printer listings. See "Printer De-Bugger" on page 33 for a solution.

(A) INVERT REM STATEMENTS

This is the most useful list-format option. Turning this feature On ("YES") causes your program remark statements to be displayed in inverse type when listed. The word "Rem" will be inverse too (try renaming it to "]" to tone it down a bit).

(B) CHANGE LIST WIDTH

The width of your listings can be altered to be from 1 to 40 characters wide. Normal is 33. Select option **B** and enter the width you want to try. If you change the list-width to 40, extra spaces won't be added in quote statements, and you won't have to POKE 33,33 before ESCape editing.

(C) LIST INDENTATION

The list indentation under each program line number may be changed from the standard 5 to any value from 0 to 39. Select **C** and enter the number you want to try. The less the indentation, the more code you can squeeze on the screen.

TO USE NEWBASIC
(after you have CREATED it and SAVED it),
RUN NEWBASIC LOADER.

MAIN MENU
OPTION **B**

Add New Basic Commands

Newbasic gives your Apple more power by offering twenty all-new commands and two slighty-new ones, each one completely optional. Since the machine-language instructions for every command must occupy a certain amount of memory space (the amount depends on the complexity of the command), memory is "stolen" by *writing over* the space used by the least-used Applesoft commands. I'll bet my $64.95 joystick that you never use the cassette SHLOAD command. For that matter, who uses cassettes at all? Applesoft has several cassette commands that waste space. Let's put that space to better use.

Here is a list of optional Newbasic commands and the old Applesoft ones that they replace. Type the letter to the left of the command to turn it On or Off ("YES" or "NO" in inverse). And remember, these commands may be *renamed* any time you want. See page 12.

(A) ELSE (replaces cassette SHLOAD command)
We've saved the best for first. ELSE is a common command in many programming languages, but missing from Applesoft until now. It completes the IF-THEN statement but remains optional. This is how you use ELSE:

```
10  PRINT "TYPE Y OR N ";: GET A$
20  IF A$ = "Y" THEN  PRINT "YES": ELSE  PRINT "NO"
```

ELSE makes Applesoft more "English-like" and friendly. It omits the need for another program line after an IF statement.

Notice how ELSE's may be nested just like for-next loops:

ELSE will not be recognized by most renumber programs. You could use GPLE to change all ELSE's to LIST's; renumber as usual; then change all LIST's back into ELSE's!

```
10  PRINT "TYPE Y OR N: ";: GET A$: PRINT
20  PRINT "TYPE 1 OR 2: ";: GET B$: PRINT
30  IF A$ =  CHR$ (3) OR B$ =  CHR$ (3) THEN  END
40  IF A$ = "Y" THEN PRINT "YES":
      IF B$ = "1" THEN PRINT "ONE":
      ELSE PRINT "TWO":
    ELSE PRINT "NO":
      IF B$ = "1" THEN PRINT "ONE":
      ELSE PRINT "TWO"
50    GOTO 10
```

The next two new commands replace the cassette LOAD and SAVE commands, not DOS's LOAD and SAVE. The commands "LOAD filename" and "SAVE filename" will still work fine.

(B) SWAP (replaces cassette LOAD command)
Here's the normal way to swap variable values X & Y:

```
TEMP=X: X=Y: Y=TEMP
```

Notice how a third variable was necessary (and an extra split-second of execution time). SWAP makes swapping (common in many sorting routines) much simpler:

```
10 A = 1.11:B = 2.22: PRINT A,B
20  SWAP A,B: PRINT A,B
30  SWAP A,B: PRINT A,B
```

Notice that you can SWAP not only variables, but integers, strings, array values, and combinations of the above:

```
10 A% = 1:B% = 2: PRINT A%,B%
20  SWAP A%,B%: PRINT A%,B%
30  SWAP A%,B%: PRINT A%,B%

10 A$ = "A":B$ = "B": PRINT A$,B$
20  SWAP A$,B$: PRINT A$,B$
30  SWAP A$,B$: PRINT A$,B$

10 A(1) = 1:B(1) = 2: PRINT A(1),B(1)
20  SWAP A(1),B(1): PRINT A(1),B(1)
30  SWAP A(1),B(1): PRINT A(1),B(1)

10 A(1) = 1:B = 2: PRINT A(1),B
20  SWAP A(1),B: PRINT A(1),B
30  SWAP A(1),B: PRINT A(1),B
```

(C) TONE (replaces cassette SAVE command)
Use TONE to play music without having to use any clunky Pokes or Calls. To play a note, use this command:

```
TONE P, L
```

The note's pitch P may be any value from 0 to 255. The length L may range from 0 to 65535. Some complete song recipes are printed in the tips section of this book.

Turn to "Tone Tunes" on page 34.

(D) HSCRN (replaces cassette RECALL command)
HSCRN tells you the "color" of any hi-res coordinate. It performs similarly to lo-res's SCRN(function. Just type "HSCRN" followed by the hi-res coordinates you want to check. Then "PRINT PEEK(234)". **0** (zero) means the point is black (off). **1** means the point is white (on). For example, to check location 275 (X), 10 (Y), do this:

Notice that HSCRN requires two commands.

```
HSCRN 275,10: PRINT PEEK(234)
```

Here is a program that shoots a white "bullet" at a white line. Give it a try:

```
10  HOME : TEXT : HGR : PRINT "POW!!!": PRINT
15 X = 17:Y =  INT ( RND (1) * 99) + 5
20  HCOLOR= 3: HPLOT 0,Y TO 279,Y
30  FOR Y = 159 TO 1 STEP  - 1
40  HSCRN X,Y:HIT =  PEEK (234)
50  IF HIT THEN  PRINT "A HIT!": HPLOT 12,Y - 3 TO
     18,Y + 3: HPLOT 12,Y + 3 TO 18,Y - 3: END
60  HCOLOR= 0: HPLOT X,Y + 1
70  HCOLOR= 3: HPLOT X,Y: NEXT
```

Since Apple's hi-res colors are made up of black and white dots (the program below proves it by reading each dot in a green line), you won't be able to determine any colors with HSCRN.

```
10  HOME : TEXT : HGR
20  HCOLOR= 1: HPLOT 0,99 TO 279,99
30  FOR X = 0 TO 279: HSCRN X,99
40  PRINT  PEEK (234);: NEXT
```

(E) SCRLDN (replaces cassette STORE command)
Each SCRLDN command scrolls the text screen down one line, not a common occurence these days.

Scrolling up can be done by PRINTing at Vtab 24.

```
10  TEXT : HOME : NORMAL
20  PRINT "STAND BY. I'M POLISHING THE TV SCREEN..."
30  FOR I = 1 TO 23: SCRLDN : NEXT : VTAB 24
40  FOR I = 1 TO 23: PRINT : NEXT : GOTO 30
```

NOTE: Selecting "YES" to add a new function automatically replaces some old command in the Command Editor. After that, you may rename the new command if you want. View changed commands with options **C** (from the Main Menu) and **6** or **7** (see keychart).

In immediate mode, PR# and IN# are always DOS commands.

The following two commands replace BASIC's IN# and PR# commands, not to be confused with DOS's IN# and PR#. It is always best to use the DOS versions of these commands from within programs. For example, PRINT CHR$(4)"PR#1"

(F) TXT2 (replaces Basic IN# command)

TXT2 reveals page 2 of text and lets you do everything there that you can do on page 1.

Since Applesoft programs normally use the same memory space that text page 2 uses, you must make your programs re-locate (actually re-RUN) themselves. Do that by adding the first line shown in this example. Warning: SAVE before you RUN!

```
10  IF  PEEK (104) < 12 THEN  POKE 104,12: POKE 3
    072,0: PRINT  CHR$ (4)"RUN TEXT TEST": REM
    THIS PROGRAM
20 A$ = "PAGE ONE.": TEXT : HOME : GOSUB 90
30 A$ = "PAGE TWO.": TXT2 : HOME : GOSUB 90
40  VTAB 23: HTAB 22: GET A$: PRINT A$
45  IF A$ = "1" THEN  TEXT : GOTO 40
50  IF A$ = "2" THEN  TXT2 : GOTO 40
60  PRINT : PRINT "HUH?": END
90  FOR I = 1 TO 93: PRINT A$;: NEXT : VTAB 22: PRINT
    : PRINT "SELECT PAGE (1 OR 2):": RETURN
```

(G) G2 (replaces Basic PR# command)

Naming this command "GR2" would be a problem; see page 14.

G2 works like GR but lets you access page 2 of lo-res. The same rules apply for lo-res page 2 as for text page 2 (see above).

Same re-location commands as in program above.

Before you can lo-res PLOT on page 2, you need to execute a TXT2 command.

```
10  IF  PEEK (104) < 12 THEN  POKE 104,12: POKE 3
    072,0: PRINT  CHR$ (4)"RUN LO-RES TEST": REM
     THIS PROGRAM
20  TXT2 : HOME : G2 :X = 8:Y = 9
30  FOR H = X TO 30 STEP 4: IF H <  > 24 THEN  FOR
    V = Y TO 13: COLOR= 5: HLIN H,H + 2 AT V: NEXT
    : COLOR= 0: PLOT H + 1,10: PLOT H + 1,12
40  NEXT H: FOR I = 1 TO 10: READ H,V: PLOT X + H
    ,Y + V: NEXT : DATA 1,4,2,3,2,4,5,4,9,2,10,1
    ,14,1,14,3,20,1,22,3
```

(H) CLRKEY (replaces Applesoft WAIT command)

Use CLRKEY instead of POKE -16368,0 to clear the keyboard buffer. Like this:

Similar example on page 27.

```
10  TEXT : HOME : PRINT ">";
20 KEY =  PEEK ( - 16384): IF KEY < 128 THEN 20
30  CLRKEY : REM TEST WITHOUT THIS COMMAND.
40  PRINT  CHR$ (KEY - 128);: GOTO 20
```

Since options I, J, and K (below) replace the same set of lo-res commands, only ONE of these options may be used at a time. (For example, if option K displays "---/NO" instead of "YES/NO", you need to turn OFF option I or J before you can use K.)

(I) TEXT & BELL COMMANDS

(replace PLOT, HLIN, VLIN and COLOR=)

CLLN (for "**CL**ear **LiN**e") replaces CALL -868 or ESC-E. Clears a text line from the cursor position to the right edge of the text window.

CLDN (for "**CL**ear **DowN**") replaces CALL -958 or ESC-F. Clears text from the cursor position to the bottom of the text window.

DON'T USE ANY OF THE COMMANDS ON THIS PAGE IF YOU WANT TO PROGRAM IN LO-RES.

SCRLUP replaces CALL -912. Scrolls text up a line.

BELL replaces CALL -198 or PRINT CHR$(7). Rings Apple's control-G bell. You can even customize the bell itself; see next page.

(J) HI-RES COMMANDS

(replace PLOT, HLIN, VLIN and COLOR=)

MODE1 replaces POKE -16304,0 (graphics switch).
MODE2 replaces POKE -16303,0 (text switch).

MIX0 replaces POKE -16302,0 (full-screen graphics)
MIX1 replaces POKE -16301,0 (split graphics & text)

PAGE1 replaces POKE -16300,0 (page 1 switch).
PAGE2 replaces POKE -16299,0 (page 2 switch).

RESL1 replaces POKE -16298,0 (lo-res switch).
RESL2 replaces POKE -16297,0 (hi-res switch).

(K) CURSOR COMMANDS

(replace PLOT, HLIN, VLIN and COLOR=)

CRSU moves the cursor up one line.
CRSD moves the cursor down one line.
CRSL moves the cursor left one character.
CRSR moves the cursor right one character.

```
10  TEXT : HOME : HTAB 6: SPEED= 255
20  PRINT "<USELESS DATA. PLEASE FILE>"
30 VTAB 11: FOR I = 1 TO 40: PRINT "-";: NEXT :SPEED= 150
40 N =  INT ( RND (1) * 3): IF N THEN S =  PEEK (49200)
50  IF N = 0 THEN  CRSU : REM CURSOR UP
60  IF N = 1 THEN  CRSD : REM CURSOR DOWN
70  PRINT "+";: IF  NOT  PEEK (36) THEN 10: ELSE 40
```

MAIN MENU
OPTION **O**

Other Features

Two of the features below aren't really Applesoft changes; they are *monitor* changes. But Newbasic doesn't care; add them to your repertoire.

(A) ADD ESCAPE-CURSOR

ATTENTION GPLE'ers— DON'T USE THIS FEATURE.

(Won't work properly with GPLE in memory)
The escape cursor indicates escape mode (moving the cursor for editing with ESC, then I, J, K, and M) by changing the cursor to a flashing plus sign. Any non-cursor key returns the normal cursor unharmed.

(B) MODIFY BELL

Newbasic lets you change Apple's bell value to make control-G sound lower with a longer duration, or higher with a shorter duration. Any number, 1-99, may be entered, but the most interesting values are between 10 and 30.

(C) MODIFY GOTO/GOSUB SYNTAX

Normally the GOTO and GOSUB commands must be followed by a line NUMBER (as in GOTO 12345). Selecting YES to this option lets GOTO and GOSUB precede a variable (as in GOTO JAIL) or an expression (as in GOSUB X+10).

```
10 LOOK = 100:BOOGIE = 200:MARINE = 300:HALT = 66
20  GOSUB LOOK: GOSUB BOOGIE: GOSUB MARINE
30  FOR I = 1 TO 3: GOSUB I * 100: NEXT : GOTO HALT
40  PRINT "DON'T PRINT THIS."
66  STOP : REM PRINT "BREAK IN 66"
100  PRINT "THIS IS LINE 100.": RETURN
200  PRINT "THIS IS LINE 200.": RETURN
300  PRINT "THIS IS LINE 300.": RETURN
```

MAIN MENU
OPTION **S**

Save Newbasic

Once you have configured Applesoft the way you want it, you can save the set-up on disk under the name "NEWBASIC". Select **S** from the Main Menu. (If you already have a file named "Newbasic" that you want to keep, exit the program now, RENAME it, type "RUN" and select SAVE from the Main Menu.)

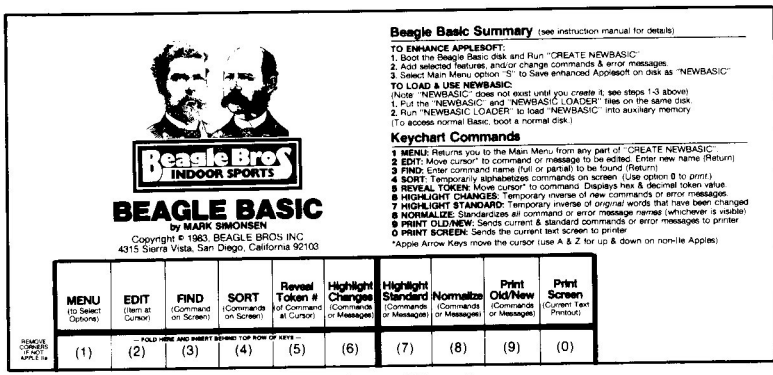
MAIN MENU
OPTION **Q**

Quit

You may exit Create Newbasic by typing **Q** from the Main Menu. If you haven't saved the current Basic, you will be notified after quitting. To save after quitting, type "RUN" and then select option S.

Changing Commands and Error Messages

One of Beagle Basic's most interesting features is the ability to change Applesoft's usually rigid commands and error messages. Run the CREATE NEWBASIC program and select **C** or **E** from the Main Menu (see page 4).



The Beagle Basic Key Chart

The keychart is needed only for making changes to commands and error messages, not for writing programs with Beagle Basic.

Both the Command Editor and Error Message Editor utilize the keychart that came with your Beagle Basic disk. Fold and insert the chart behind the top row of keys on your Apple. Following are descriptions of each of the ten keychart commands and instructions for changing Applesoft commands and error messages.

1: MAIN MENU

Return to the Main Menu (see page 4) from Create Newbasic's various modes by typing **1** as noted on the keychart or by pressing **ESC**.

2: EDIT (Commands or Error Messages)

While in the Command or Error Message Editor, move the cursor to the command or message you want to edit (change), select keychart option **2** and type a new word or words and press Return. The new command or error message will immediately be in effect. To confirm this, you may return to the Main Menu (**1**), quit (**Q**) and give it a try. Usually you can return to Create Newbasic by typing "RUN". If you somehow zapped it with your test, you may have to type "RUN CREATE NEWBASIC."

Changing Commands

To change any Applesoft command, or to see the commands currently in effect, type **C** from the Main Menu. You will enter the COMMAND EDITOR and the screen will be filled with the current Applesoft commands or "keywords"— that is, every word that you can type that your Apple will understand (except disk or *DOS* commands like CATALOG, UNLOCK, INIT, and so on).

MOVE MODE: In the upper-left corner of the screen you will see the word "**MOVE**". This means you are in the "Cursor-Move Mode". The cursor is the inverse bar over one of the commands. To move this cursor, use the Left & Right Arrow Keys and the Up & Down Arrow keys (or use **A** & **Z** for up & down if you don't have an Apple IIe). Move the cursor around now. I'll wait here...

FREE CHARACTERS: The number after the word "**FREE**" in the upper-right corner of the screen indicates the number of spare characters remaining in the command table. For example, if you shorten the INVERSE command to "INV", you *increase* the number of free bytes by 4. If you change LET to "ASSIGN", you *reduce* the number of free bytes by 3. You cannot lengthen a command if you have no spare characters; you must shorten another command first. No command longer than the cursor, seven characters, is allowed.

EXPERIMENT: Change the HOME command to "CLS" (many languages use CLS to mean "Clear Screen").—

1. After running Create Newbasic, select the Command Change Mode (**C**) from the Main Menu. You will see the current 107 commands on the screen.
2. Use the Arrow Keys or the FIND option (page 16) to move the cursor to HOME (the 5th command down in the 4th column).
3. Type **2** to edit the word. The word "**EDITING**" will appear in the upper-left corner of the screen, and the bar-cursor will change to a blinking square.
4. Type "CLS" as the new command, and press Return.

That's it! You have changed the HOME command to "CLS". You are now back in Move Mode and the free characters (upper-right of the screen) have been increased by one because "CLS" is one character shorter than "HOME".

Later on, typing "CLS" will clear the screen. If you type "HOME", your Apple will grumble "?SYNTAX ERROR" (Computerese for *"Never heard of it!"*).

To shorten INVERSE to "INV", move the cursor to INVERSE (7th word in the 1st column), and press **2** to edit. Use the Right Arrow key to trace over the I, N and V, and hit Return. Easy, right? And you have gained four more spare characters. Later on, typing "INV" will create inverse type as expected. Typing "INVERSE", however, will print "?SYNTAX ERROR" in inverse, because your Apple *tokenized* "INVERSE" into two words; "INV" (which it recognized *and executed*) and "ERSE". "Erse" means nothing to most Apples, so ?SYNTAX ERROR.

COMMAND RULES: New commands may contain no spaces, lower case or control characters. Other than that, all characters are legal. The position of the Caps Lock key on the IIe has no bearing on input; all keystrokes are interpreted as upper case. Pressing Return will accept whatever has been entered up to the cursor. If the cursor is on the *leftmost* character in a command when you press Return, no change will occur.

Use the "SORT" option to compare current command names.

COMMANDS WITHIN OTHER COMMAND NAMES can be a problem if you don't watch out. For example, notice how HGR2 comes *before* HGR in the command table. If it didn't, and you typed "HGR2", Applesoft would scan the table, find "HGR" first, quit scanning and "parse" HGR2 into two words, "HGR" and "2". If you want to create a command name (like "GONOW") that starts with the same characters as another command (say "GO"), always put the longer command first in the command table. Commands that *end* with the same characters (like PLOT and HPLOT) are no problem.

Tokens are explained more on page 18.

NOTE: You can test new commands and functions by quitting Create Newbasic (Main Menu option Q). Re-enter the program by typing "RUN" or "RUN CREATE NEWBASIC". Be sure to SAVE (Main Menu option S) any version of Newbasic that you want to keep.

Changing Error Messages

Selecting option **E** from the Main Menu will enter the ERROR MESSAGE EDITOR which displays and lets you change Applesoft error messages, the words your Apple prints when an error is encountered. Operation of the Error Message Editor is similar to that of the Command Editor. Differences are pointed out below.

EXPERIMENT: Change the message ?SYNTAX ERROR to "?TYPING GOOF!"—

1. Select the Error Message Change Mode, **E**, from the Main Menu. You will see the current 17 Applesoft error messages on the screen, as well as the words "ERROR", "IN" and "BREAK".

2. Use the Arrow Keys to move the cursor to SYNTAX (the 2nd error message down). Notice that word "ERROR" appears near the bottom of the screen and must be changed separately. (The "?" that precedes Applesoft error messages may be changed too. See the tip on page 32.)

3. Type **2** to edit the message. The word "**EDITING**" will appear in the upper-left corner of the screen, and the bar-cursor will change to a blinking square.

4. Type "TYPING" and hit Return.

5. Move the cursor to ERROR, third word from the bottom, type **2** to edit, type "GOOF!" and hit Return.

See page 32 for ways to test your new error messages.

You have officially changed the ?Syntax Error message to "?Typing Goof!". Later on, typing an illegal statement (my favorite is "sdjaflkjfd") will produce a well-deserved ?Typing Goof! message and a beep. The word "Goof!" will now follow *every* Applesoft error message. Since "?Illegal Quantity Goof!" and "?Out of Data Goof!" don't make a lot of sense, you may want to *not* change the word "Error".

ERROR MESSAGE RULES: The Error Message Editor is not concerned with spare characters, because there are no spare characters; each error message is already at its maximum length (the length of the cursor). Other rules are the same as for commands, except spaces *are* legal in error messages.

"**IN**" and "**BREAK**" appear in messages like BREAK IN 1234, and may be changed.

3: FIND (Commands only)

From the Command Editor, typing a **3** will print the word "**FIND:**" on the screen. Enter any command's name and the cursor will move to that command. If the search is unsuccessful, the cursor will stay where it was. It's not necessary to type an entire command. For example, if you type "X" and press Return, the cursor will position itself at the first command that starts with an "X".

4: SORT (Commands only)

Pressing **4** will temporarily sort (alphabetize) the commands on the screen. If you want to send this list to your printer, press **0** (see next page). Pressing any key (like the Space Bar) returns the command order to normal, the order (left to right) in which they are stored in memory.

Sorting lets you check to make sure you have not given the same name to two or more different commands. Doing so would disable one of those commands.

Giving a BASIC command a DOS command name could possibly have the same effect. To play it safe, never give a BASIC command a word that DOS uses— APPEND, BLOAD, BRUN, BSAVE, CATALOG, CHAIN, CLOSE, DELETE, EXEC, FP, INIT, INT, IN#, LOAD, LOCK, MAXFILES, MON, NOMON, OPEN, POSITION, PR#, READ, RENAME, RUN, SAVE, UNLOCK, VERIFY or WRITE.

5: TOKEN NUMBER (Commands only)

Pressing **5** while in the Move Mode will print the word "**TOKEN**" at the upper-left of the screen, erase the command at the cursor and replace it with the hex and decimal values for that command's "token". Tokens are the numbers Applesoft assigns each of its keywords. Pressing any key (like the Space Bar or Return) will return you to Move Mode.

6: HIGHLIGHT CHANGES

(Commands or Error Messages)

Pressing **6** will show you the changes that you have made. All non-standard commands or error messages (if any) will be highlighted in inverse. The word "**CHANGED**" will appear in the upper-left corner of the screen. Press any key to return to Move Mode.

7: HIGHLIGHT STANDARD

(Commands or Error Messages)
Pressing **7** will show you the standard version of each command or error message (if any) that has been changed. The word "**STANDARD**" will appear in the upper-left of the screen, and the original commands or error messages will appear in inverse. Press any key to return to Move Mode.

8: NORMALIZE (Commands and Error Messages)
Typing **8** followed by a **Yes** (when asked for approval) will normalize the names of all commands OR error messages (depending which is on the screen), setting them to their original state. If, for instance, you wanted to change new commands CLS and INV back to standard "HOME" and "INVERSE" you could use this function. **BEWARE** that all other command or error message changes will *also* be undone. **NOTE** that standardizing all command names DOES NOT standardize (or change in the least) the *function* of any command. For example, if you have turned on the ELSE function (page 6), the command name ELSE will be standardized to "SHLOAD" but the *function* of the command SHLOAD when executed in a program will still be that of ELSE. Confusing? Yes. Until you think about it for a week or so...

The last two keychart commands involve your printer. If you don't use a printer, don't use keychart options 9 and Zero. If your printer is connected to other than Slot 1, change the SLOT variable in Line 3 of the CREATE NEWBASIC program.

9: PRINT OLD/NEW

(Commands or Error Messages)
Selecting **9** after entering the Command Editor or Error Message Editor will send a listing of all standard and revised Applesoft commands or error messages to your printer. Hex and decimal token numbers are printed adjacent to commands. You must be in the Command Editor *or* Error Message Editor to use this function.

0: PRINT SCREEN

Selecting **0** (Zero) will dump the current text screen to your printer. The most common use for this feature is printing an alphabetized list of commands (keychart option 4), although it may be used anytime.

Beagle Basic and the Token System

Applesoft has a 107-word "command table" stored in memory that contains "keywords" (words like PRINT, HOME, FOR, GOSUB, and so on) that are used by Applesoft programs. For efficiency, every keyword is represented by a one-byte "token" or number. For example, instead of storing the command "PRINT" as five characters or bytes (P, R, I, N and T), it is stored as *one* byte— the value 186 (hex $BA).

A complete list of tokens may be acquired by selecting keychart option 9 while in the Command Editor.

When you type in a command like—

HOME: PRINT X; "ABC"

Applesoft converts all of the keywords into tokens by looking at its command table. Words that aren't recognized and characters between quote marks are represented by the ASCII value of each character. Like this:

HOME is tokenized as **151** ($97)
":" becomes **58** ($3A, the ASCII value of ":")
PRINT is tokenized as **186** ($BA)
X becomes **88** ($58, the ASCII value of X)
; becomes **59** ($3B, the ASCII value of ;)
" becomes **34** ($22, the ASCII value of ")
A becomes **65** ($41, the ASCII value of A)
B becomes **66** ($42, the ASCII value of B)
C becomes **67** ($43, the ASCII value of C)
" becomes **34** ($22, the ASCII value of ")

So a program is simply a series of tokens and ASCII values. What words are displayed when you list depends on the words in the current command table.

Beagle Basic lets you change the contents of the command table by renaming keywords and storing the new words in memory. Let's say you change the keyword "HOME" to "CLS". Now when you type "HOME", Applesoft won't recognize it, and CLS will clear the screen. BUT if you load a program that contains the old HOME command (actually just the *token* for HOME) and you list it, you will see that CLS has *automatically replaced* each occurence of HOME.

OR if you change HOME to CLS, and write a program full of CLS's, it will work fine with your version of Newbasic. If you now re-boot and use normal Applesoft, your program will still work, even though all of your CLS's will list as HOME's.

New FUNCTIONS Need Newbasic!

Changing names of commands is one thing, but adding new *functions* like ELSE or SWAP or TONE is another story. For these new functions to work properly, you MUST have the correct version of Newbasic loaded into memory. Otherwise, the machine language instructions that enable Applesoft to execute these functions just aren't there. Running a program with non-standard Applesoft commands without Newbasic loaded will produce unpredictable results.

Exiting Newbasic— APPLE II+ AND IIe

There are two ways to exit Newbasic and re-instate normal Applesoft:

1. Boot with a normal disk. This is the "cleanest" way to leave Newbasic, because it completely normalizes your system.

2. Or type "INT". This command normally would switch you to Integer Basic. With Newbasic in effect, INT instead switches you to Applesoft (without erasing your program) and prints "Language Not Available" on the screen (Integer wasn't found). If you want, you could use *DOS Boss* to rename "INT" to "ROM" and "Language Not Available" to "Applesoft Available" or something else appropriate.

NOTE: Pressing RESET will always leave you in *Newbasic* with your program intact. FP will also always leave you in Newbasic, but with your program gone.

Exiting Newbasic— APPLE II (Integer machines):

There are two ways to exit Newbasic and re-instate normal Integer Basic in your antique Apple:

1. Boot with a normal disk. This is the "cleanest" way to leave Newbasic, because it completely normalizes your system.

2. Type "INT". This command will switch you to Integer Basic and erase your program.

NOTE: FP will return you to (or leave you in) Newbasic, with your program gone.

GPLE and Newbasic

GPLE is compatible with Beagle Basic except for the following things. Let us know if you find more.

Thing #1: If you are using GPLE on the language card (GPLE version PLE.LC), it must be loaded *before* Newbasic is loaded.

Thing #2: Using the INT command with Newbasic and PLE.LC loaded won't exit Newbasic as described above. Don't use INT.

Thing #3: Newbasic's ESCape cursor won't function properly with GPLE in memory. Sorry.

Thing #4: CREATE NEWBASIC should not be run while GPLE.LC is in memory. Doing so will prevent parts of the program from working properly.

Normal Location DOS Required

DOS can't be in auxiliary memory (RAM card or language card) when using Newbasic.

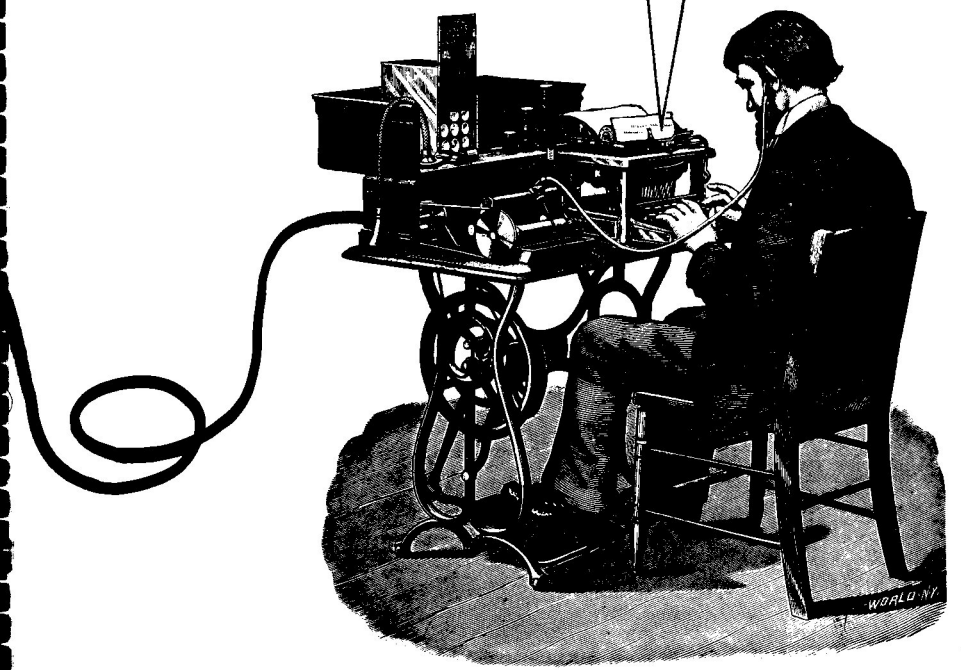
Dear Apple Programmer,

Hello out there. Beagle Bros' Apple TIP BOOK #6 starts on the following page. You will be pleased to know that all of the programs have been listed in standard 40-column format, so your listings, if typed correctly, will appear line-for-line as you see them in print. Good luck!

Oh, I almost forgot-- One of thf qsphsbnt xjmm dbvtf zpvs Bqqmf up ejtjoufhsbuf; J gpshfu xijdi pof.

Zpvs gsjfoe,
Vodmf Mpvjf

Printed in U.S.A.



Text Screen Formatter

by Mark Simonsen

Note: This program is already on the Beagle Basic disk.

Here's a bonus program that lets you format the 40-column text screen the way you want it and convert the finished product into VTAB, HTAB and PRINT statements that can be appended to any Applesoft program.

To begin, **BLOAD TEXT SCREEN FORMATTER** (or BRUN if you want) from the Beagle Basic disk. Then type "**CALL 25000**". The following commands will let you type and format text on the screen:

TO MOVE THE CURSOR
Left Arrow: Move cursor left.
Right Arrow: Move cursor right.
Up Arrow: Move cursor up (non-IIe; use control-A).
Down Arrow: Move cursor down (non-IIe; use control-Z).

TO MOVE THE ENTIRE SCREEN
(Keys are in diamond pattern.)
control-S: Move screen left.
control-D: Move screen right.
control-E: Move screen up.
control-X: Move screen down.

TEXT APPEARANCE
control-F: **F**lash.
control-I: **I**nverse.
control-N: **N**ormal.
ESC: Upper/lower case toggle.

Remove GPLE from your Apple for best results.

AND
control-@: Clear the screen.
control-C: **C**enter text line.
control-L: Clear to end of **L**ine.
control-P: Clear to end of **P**age.
control-W: **W**indow toggle
Return: Carriage Return.

QUIT AND CONVERT
control-Q: **Q**uit and convert.

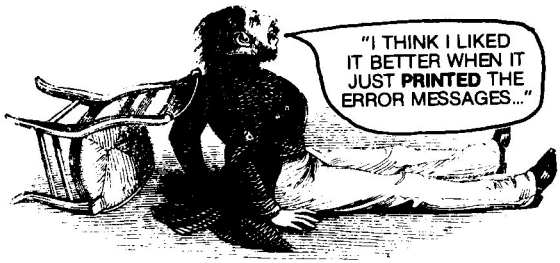
When you quit (control-Q), any program in memory will be replaced by Applesoft statements that will print the screen the way you had it formatted; type **RUN** to check it out. If you don't like what you see, add a **CALL 25003** as the last

program line and you will be put back in the Screen Editor when you RUN. To start over with a clean screen, **CALL 25000** again.

The Applesoft code (without any Calls) may be saved on disk or appended to another Applesoft program.

The default starting line and increment are both 10. To change them:

```
POKE 25006, START-INT(START/256)*256
POKE 25007, INT(START/256)
POKE 25008, INCREMENT
```



ONERR TRY AGAIN...

Don't you *hate* it when DOS gives you an error message like "File Not Found" just because you typed "RUN HELOO", or "I/O Error" because you put your disk in the drive sideways? The following program will put a small patch into DOS that, in the event of an error, causes the cursor to be put back on the *same line* as your previous typing statement. That way you can simply *trace over* your statement making corrections if necessary as you go.

Note: This program assumes that DOS is in the normal 48K location. And it wipes out the semi-useless CHAIN command.

```
10  POKE 42751,76: POKE 42752,240
    : POKE 42753,164
20  FOR I = 42224 TO 42235: READ
    BYTE: POKE I,BYTE: NEXT I
30  DATA 165,37,233,4,32,91,251,1
    08,94,157,0,0
```

Giving credit where credit is due— Don Worth and Pieter Lechner inspired us on this one. (They wrote *Bag of Tricks* and *Beneath Apple DOS*. Run out and buy one of each; we'll wait here...) We improved on their version by making it shorter and by moving it to another location in DOS so you can use it with *ProntoDOS* and other DOS modifications.

We read *Beneath Apple DOS* every day; sometimes more often.

CURSOR MADNESS

If Bert Kersey's text batons aren't enough for you—

```
5  HOME :A$ = "!/-" +  CHR$ (92)
30  VTAB 7: FOR X = 1 TO 4: HTAB
     9: FOR Y = 1 TO 10: PRINT  MID$
     (A$,X,1);: NEXT : NEXT
40  GOTO 30
```

—try running the following program; the screen will be *packed* with text screen twirlers. You probably won't want to quit the program for several days, but when you do, RESET is the only way out.

Warning: DO NOT run this program with the room lights off and the monitor contrast turned up full blast and your nose touching the screen and your eyelids propped open.

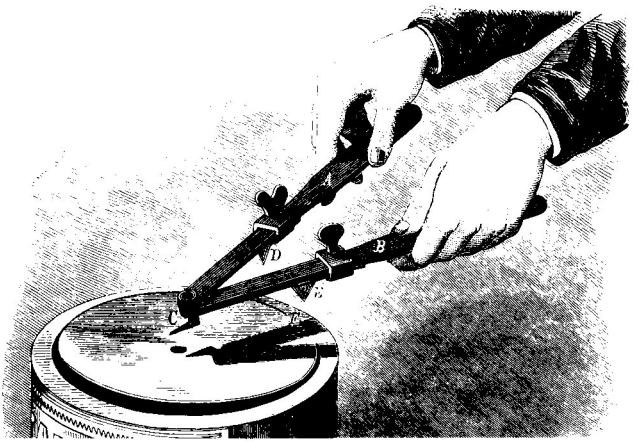
```
10  FOR I = 768 TO 814: READ V: POKE
     I,V: NEXT I: CALL 768
20  DATA 162,4,169,0,141,16,3,169
     ,4,141,17,3,189,42,3,141,1,6
     ,160,4,136,208,253,238,16,3,
     208,243,238,17,3,172,17,3,19
     2,8,208,233,202,208,217,240,
     213,175,173,220,161
```

WHAT'S HCOLOR=4 AND HCOLOR=7 AND HCOLOR=5 ALL OVER?

Who says that Apple II's only have six hi-res colors? Run this program and you'll become a non-believer.

```
10  TEXT : HOME : GR : HGR : REM
     GR SETS WINDOW BELOW HI-RES
20  FOR I = 1 TO 6: READ C(I): NEXT
     : FOR I = 1 TO 6: READ C$(I)
     : NEXT
40  FOR I = 1 TO 6: FOR J = 1 TO
     6
50  HOME : PRINT C$(J);: IF J <  >
     I THEN  PRINT " + "C$(I)
60  FOR Y = 22 TO 149 STEP 2: HCOLOR=
     C(I): HPLOT 32,Y TO 247,Y
70  HCOLOR= C(J): HPLOT 32,Y + 1 TO
     247,Y + 1: NEXT Y,J,I
110  DATA 0,1,2,3,5,6,BLACK,GREEN
     ,VIOLET,WHITE,ORANGE,BLUE
```

Color monitors are very helpful when viewing hi-res color.



ONERR REPAIR

Due to a quirk in the ONERR system, this program will crash after the 85th error. Hold control-C down (and REPT if necessary) to cause an error and a jump to line 30.

Please don't write us with the reasons for this bug.
Oh, never mind; go ahead.

```
10  ONERR  GOTO 30
20  GOTO 20
30  PRINT "ERROR # "ERR:ERR = ERR
      + 1: GOTO 20
```

Adding a CALL-3288 fixes the bug.

```
10  ONERR  GOTO 30
20  GOTO 20
30  PRINT "ERROR # "ERR:ERR = ERR
      + 1: CALL  - 3288: GOTO 20
```

CALL THIS NUMBER

Here's one that will really confuse the gang—

CALL 64246

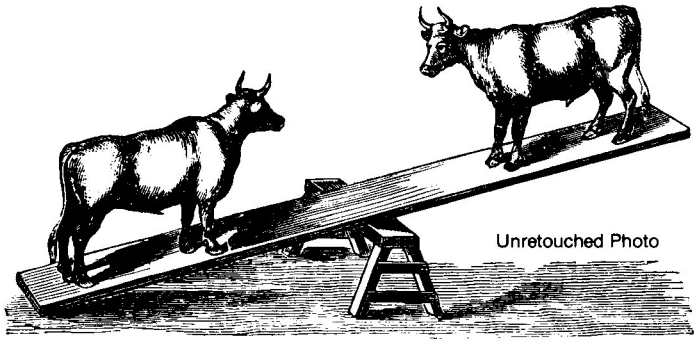
We would explain it to you, but no one has explained it to us yet. Apparently it has something to do with the position of the Moon...

IIe FLICKERBUG

Have you noticed the flicker you get on the IIe's text and hi-res page 2? This program shows it at its worst:

Apple II+'s don't seem to have this problem.

```
10  HGR : POKE 28,127: CALL  - 30
     82: HGR2
20  FOR I = 1 TO 10: LIST : NEXT
     : TEXT
```



MEM COMPARER

This program will compare two ranges of memory such as two hi-res pictures or two machine language programs, printing the bytes that are different.

OS and **OE** are "Old Start" and "Old End" locations. **NS** is "New Start".

```
5000 N = OS:LOC = 60: GOSUB 5020
5001 N = OE:LOC = 62: GOSUB 5020
5002 N = NS:LOC = 66: GOSUB 5020
5010  POKE 768,160: POKE 769,0: POKE
     770,76: POKE 771,54: POKE 77
     2,254: CALL 768: RETURN
5020  POKE LOC,N -  INT (N / 256)
      * 256: POKE LOC + 1, INT (N
      / 256): RETURN
```

HIDING AMONGST THE BUFFERS

One of the best places to put a small machine language program (besides under the bed) is between DOS and its buffers. That way, your code can't be zapped by FP, RESET and other would-be attackers. To make room for a program do this:

```
POKE 40193,PEEK(40193)-N
CALL 42964
```

This will allocate N*256 bytes of safe space. To de-allocate the space do this:

```
POKE 40193,PEEK(40193)+N
CALL 42964
```

If you've got GPLE or Double-Take or some other program hanging around in your Apple, no guarantees.

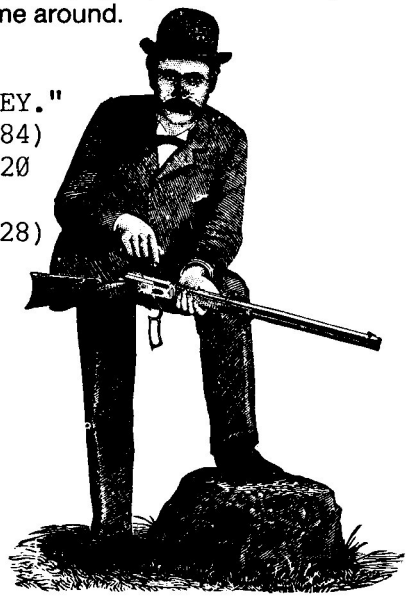
KEYPEEKER

To see if someone has pressed a key without having to wait until they do it (as is the case with GET) use some kind of loop with **K=PEEK(-16384)** in it. If K is greater than 127 then a key has been pressed, and you can jump out of the loop. Using **A$=CHR$(K-128)** will set up A$ as though you *had* used the GET function. Don't forget to do a **POKE -16368,0** (or Beagle Basic's **CLRKEY**; see page 9) to clear the keyboard buffer for the next time around.

Here's an example:

GPLE's typeahead buffer (certain GPLE versions only) prevents the keypeeker from working. We recommend running the "Config.GPLE" program to cancel the buffer's effects.

```
10  PRINT "PRESS A KEY."
20 K =  PEEK ( - 16384)
25  IF K < 128 THEN 20
30  POKE  - 16368,0
35 A$ =  CHR$ (K - 128)
40  FOR I = 1 TO 40
45  PRINT A$;: NEXT
50  GOTO 10
```



BLOAD DETECTORS

If you Bload a binary file *without* telling it where to load, with a command like "BLOAD FILE", it will load to the location in memory *from which it was saved.* (Read that sentence again.) If you *specify* a starting location in a command like "BLOAD FILE, A12345", the file will, of course, load to the specified location.

These two statements will tell you the starting location and length of a file immediately after a BLOAD:

```
ADDR = PEEK(43634) + PEEK(43635) * 256
LENG = PEEK(43616) + PEEK(43617) * 256
```

If you like to think in hex, **CALL -151** to enter the monitor, and type "**AA60.AA73**". The first two bytes printed will be the Length and the last two will be the start Address (in low-byte/high-byte order, of course).

PEEK-POKE MEM-MOVE

Check out our latest Peeks & Pokes chart. It has a memory move routine that uses Apple's built-in CALL -468. And it *works!*

One of our programmers exits to FID's menu by opening his drive door during disk access. He once went two straight weeks without destroying a disk.

FID-QUITTER

FID is fine, but you've got to live with some of its quirks. For example, it's hard to exit FID at certain times. Well, use that old program killer, control-RESET. Then, to re-enter FID, just CALL 2051.

TEXT SCREEN MEM VALUE

Here's a good way to determine the actual memory location of any text screen position:

```
10  TEXT : HOME : PRINT "TEXT SCR
    EEN LOCATION:"
20  INPUT "X=";X: INPUT "Y=";Y
30  VTAB Y: HTAB X
40 P =  PEEK (40) +  PEEK (41) *
    256 +  PEEK (36): VTAB 4: PRINT
    : PRINT "MEMORY LOCATION ";P
```

HI-RES MEM VALUE

And here's a program that determines the memory location of any hi-res plot:

```
10  TEXT : HOME : INPUT "HI-RES P
    AGE 1 OR 2?";PG
20  INPUT "X=";X: INPUT "Y=";Y
30  POKE 230,PG * 32: HPLOT X,Y
40  PRINT : PRINT "MEMORY LOCATIO
    N="; PEEK (38) +  PEEK (39) *
    256 +  PEEK (229)
```

DEC DUMPER

Warning: This is a potentially *boring* tip, depending on your level of programming. No charge for this one.

The Apple monitor has the ability to dump a range of memory in hex. Running the program below will set up a short machine language routine at 768 ($300) that will dump memory in *decimal* rather than hex. It is invoked by using the **&** command. For example, to look at hi-res page 1, enter the start and end addresses separated by commas after the "&" (like this: **&8192, 16383**).

```
10  FOR I = 768 TO 875: READ V: POKE
    I,V: NEXT I: POKE 1014,0: POKE
    1015,3
20  DATA 160,0,32,183,0,32,12,218
    ,165,80,133,60,165,81,133,61
    ,32,183,0,201,44,208,14,32,1
    77,0,32,12,218,165,80,133,62
    ,165,81,133,63,76,70,3,32,14
    2,253,173,0,192,201,131,240,
    55
30  DATA 165,61,166,60,32,36,237,
    160,0,169,173,76,237,253,165
    ,60,41,7,208,8,32,40,3,169,4
    ,141,108,3,173,108,3,24,105,
    4,141,108,3,133,36,177,60,17
    0,169,0,32,36,237,160,0,32
40  DATA 186,252,144,216,96,104,1
    04,96
```

CURSORS FOILED AGAIN

Run this program and watch that flashy cursor of yours screech to a grinding halt.

```
10  FOR I = 768 TO 806: READ V: POKE
    I,V: NEXT : CALL 768
20  DATA 169,11,133,56,169,3,133,
    57,76,234,3,72,41,63,9,0,145
    ,40,104,230,78,208,2,230,79,
    44,0,192,16,245,145,40,173,0
    ,192,44,16,192,96
```

Many were offended by the pre-IIe flashing cursor. We kind of liked it.

BRUN HI-RES PIX!

You know what happens if you unknowingly BRUN (instead of BLOAD) a hi-res picture... *Crashola!!*

And you know what a pain it is to select a picture's page number and full- or split-screen status with all of those dumb Pokes?

Boy, have we got a Tip for you!— Now you can update your hi-res pictures so they are not only BRUNnable, but they become VISIBLE when you BRUN them, on either hi-res page, split-screen or full. Never mess with Bloading and Poking again!

Just run this little program. It does all the magic for you.

This program stores machine language instructions in the mysterious blank spaces found in high-res picture data.

```
10  INPUT "NAME OF PICTURE? ";PN$
20  PRINT "LOAD TO PAGE 1 OR 2? "
    ;: GET P$: PRINT
30  IF P$ < > "1" AND P$ < > "2
    " THEN 20
40 P = VAL (P$):AD = P * 8192
50  PRINT "DISPLAY 4 LINES OF TEX
    T AT BOTTOM? ";: GET T$: PRINT
60  IF T$ < > "Y" AND T$ < > "N
    " THEN 50
70 T = 1: IF T$ = "N" THEN T = 0
80  PRINT "LOADING..."
90  PRINT CHR$ (4)"BLOAD "PN$",A
    "AD
100  PRINT "CHANGING..."
110 P1 = PEEK (AD):P2 = PEEK (A
    D + 1):P3 = PEEK (AD + 2)
120  POKE AD,76: POKE AD + 1,120:
     POKE AD + 2,32 * P
130  POKE AD + 120,169: POKE AD +
    121,P1: POKE AD + 122,141: POKE
    AD + 123,0: POKE AD + 124,32
     * P: POKE AD + 125,76: POKE
    AD + 126,248: POKE AD + 127,
    32 * P
```

```
140  POKE AD + 248,169: POKE AD +
     249,P2: POKE AD + 250,141: POKE
     AD + 251,1: POKE AD + 252,32
      * P: POKE AD + 253,76: POKE
     AD + 254,120: POKE AD + 255,
     32 * P + 1
150  POKE AD + 376,169: POKE AD +
     377,P3: POKE AD + 378,141: POKE
     AD + 379,2: POKE AD + 380,32
      * P: POKE AD + 381,76: POKE
     AD + 382,248: POKE AD + 383,
     32 * P + 1
160  POKE AD + 504,44: POKE AD +
     505,80: POKE AD + 506,192: POKE
     AD + 507,76: POKE AD + 508,1
     20: POKE AD + 509,32 * P + 2

170  POKE AD + 632,44: POKE AD +
     633,82 + T: POKE AD + 634,19
     2: POKE AD + 635,76: POKE AD
      + 636,248: POKE AD + 637,32
      * P + 2
180  POKE AD + 760,44: POKE AD +
     761,83 + P: POKE AD + 762,19
     2: POKE AD + 763,76: POKE AD
      + 764,120: POKE AD + 765,32
      * P + 3
190  POKE AD + 888,44: POKE AD +
     889,87: POKE AD + 890,192: POKE
     AD + 891,96
200  PRINT "SAVING..."
210  PRINT  CHR$ (4)"BSAVE "PN$",
     A"AD",L8192"
```

Remember, your program should list line-for line like the one printed here.

A three-byte catch: If you BLOAD a converted picture, the first three bytes (upper-left sliver of the screen) will be flawed. To fix the flaw, simply CALL 8192 (page 1 pics) or CALL 16384 (page 2). And, if you want to convert a BRUNnable picture back to normal, just BRUN it, then BSAVE it with normal procedures.

?ERROR !FIXER

With Newbasic loaded, you can replace the "?" in front of error messages to any character you want. Just run this program:

```
10  FOR I = 768 TO 782: READ BYTE
    : POKE I,BYTE: NEXT
20  PRINT "ENTER THE NEW CHARACTE
    R: ";: GET A$: PRINT A$
30  IF A$ < " " THEN 20
40  POKE 775, ASC (A$): CALL 768
100  DATA 44,131,192,44,131,192,1
    69,255,141,91,219,44,128,192
    ,96
```

ERROR TESTING

Test your new Newbasic error messages— Typing each of the commands below (without line numbers) will reveal the equivalent of the message shown. Each command assumes memory has been cleared. Type "FP" if it hasn't.

To get this message...	**type this:**
?Next Without For	NEXT
?Syntax	GARBAGE
?Return Without Gosub ..	RETURN
?Out of Data	PRINT: READ A
?Illegal Quantity	VTAB 25
?Overflow	PRINT 10 ^ 39
?Out of Memory	HIMEM: 0
?Undef'd Statement	GOTO X
?Bad Subscript	PRINT A(11)
?Redim'd Array	DIM A(1): DIM A(1)
?Illegal Direct	READ A
?Division by Zero	PRINT X/0
?Type Mismatch	A$=2
?String Too Long	FOR X=1 TO 256: A$=A$+"A": NEXT
?Formula Too Complex ..	IF "X" THEN IF "X" THEN ?X
?Can't Continue	CONT
?Undef'd Function	PRINT FN X(X)

SUBSTRING SEARCH

Some computer languages have a command that will search for a substring within a larger string. The subroutine starting at Line 100 gives Applesoft that ability. The lines before 100 are just an example of how to use it.

```
10  TEXT : HOME
20 S$ = "THIS SENTENCE TESTS THIS
      PROGRAM FOR YOU": PRINT S$
30  PRINT : PRINT : INPUT "STRING
      TO FIND:";FIND$: GOSUB 100
40  IF FOUND THEN  INVERSE : VTAB
     1: HTAB I: PRINT FIND$: NORMAL
     : END
50  PRINT "(NOT FOUND)": END
100  FOR I = 1 TO  LEN (S$)
110 K = I:FOUND = 1
120  FOR J = 1 TO  LEN (FIND$)
130  IF  MID$ (FIND$,J,1) <  >  MID$
     (S$,K,1) THEN J =  LEN (FIND
     $):FOUND = 0
140 K = K + 1: NEXT J
150  IF FOUND THEN  RETURN
160  NEXT I:I = 0: RETURN
```

The variable I is the starting position (in the long string) of the found substring.

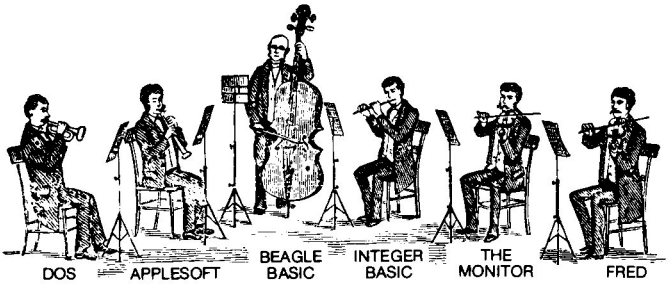
You know why the variable I is used so much by us programmers in for-next loops? We think it's because I stands for Increment.

PRINTER DE-BUGGER

If you've ever shouted obscene machine code at your printer for doing 47 line feeds, 192 carriage returns and 16 beeps when you listed a program with embedded control-characters or catalogged a disk with inverse file names, this program's for you. After you RUN this program (only once please), do a **PR#1** to turn on your printer, then **CALL 768**. All control characters will be sent to the printer as lower case. Inverse and Flashing characters will be printed as normal.

```
10  FOR I = 768 TO 810: READ V: POKE
     I,V: NEXT I
20  DATA  169,11,133,54,169,3,133
     ,55,76,234,3,201,141,240,25,
     201,32,176,2,9,192,201,96,17
     6,2,9,128,201,128,176,3,24,1
     05,64,201,160,176,2,9,96,76,
     2,193
```

Add this program to a routine that turns on your printer, prints, and then turns the printer off.



TONE TUNES

Here's a hit song or two you can type and play on your Apple. All use the new Beagle Basic TONE command, which you must have activated (see page 7).

Twinkle, Twinkle, etc.

```
10  DIM PITCH(21)
20  FOR I = 1 TO 21: READ PITCH(I
    ): NEXT
30  GOSUB 100
40  FOR I = 0 TO 1:TIME = 126
50  FOR J = 1 TO 7
60  IF J = 7 THEN TIME = 252
70  TONE PITCH(J + 14),TIME
80  NEXT : NEXT
90  GOSUB 100: END
100  FOR I = 0 TO 1:TIME = 126
110  FOR J = 1 TO 7
120  IF J = 7 THEN TIME = 252
130  TONE PITCH(J + I * 7),TIME
140  NEXT : NEXT : RETURN
200  DATA 63,63,127,127,140,140,1
     27,110,110,103,103,83,83,63,
     127,127,110,110,103,103,83
```

Entire Tone Range:

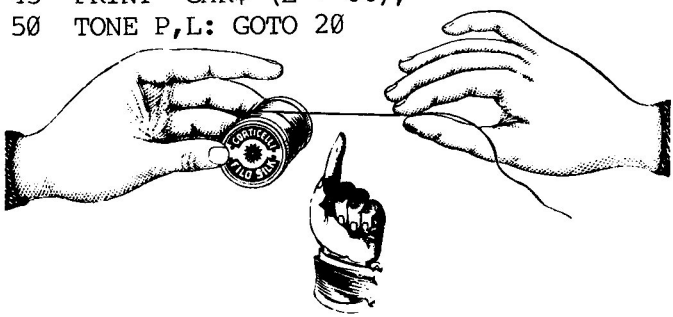
```
10 L = 5:A = 0:B = 255:S = 10
20  HOME : FOR P = A TO B STEP S
30  VTAB 9: HTAB 17
40  PRINT "PITCH=";P;" "
50  HTAB 17: PRINT "LENGTH=";L
70  TONE P,L: NEXT
90 S = S *  - 1:A = 255 * (S < 0)
     :B = 255 * (S > 0): GOTO 20
```

Scale:

```
10  FOR X = 1 TO 36: READ PITCH,K
    EY$: TONE PITCH,20: PRINT PI
    TCH;"=";KEY$: NEXT
100 DATA 13,G,25,G#,39,A,51,A#,6
    3,B,73,C,83,C#,93,D,103,D#,1
    10,E,119,F,127,F#,134,G,140,
    G#,147,A,153,A#,159,B,164,C,
    169,C#,174,D,179,D#,183,E,18
    7,F,191,F#,195,G,198,G#,201,
    A,204,A#,207,B,210,C,212,C#,
    215,D,217,D#,219,E,221,F,223
    ,F#
```

Messing Around:

```
10  HOME : INVERSE
20 P =  INT ( RND (1) * 40) + 1
30 L =  INT ( RND (1) * 23) + 1
40  VTAB L: HTAB P
45  PRINT  CHR$ (L + 64);
50  TONE P,L: GOTO 20
```



CENTERSTRING

This subroutine is for those of us that are too lazy to figure out the correct HTAB for centering words on the screen. To use it, set A$ equal to the string to be printed and then GOSUB 500.

```
10  HOME :A$ = "<----- CENTER ---
    -->": GOSUB 500:A$ = "ALMOST
     ANYTHING": GOSUB 500:A$ = "
    IN A FLASH!": GOSUB 500
499  END
500 H = 21 - ( LEN (A$) / 2): HTAB
    H: PRINT A$: RETURN
```

You could also write a routine that prints copy flush right.

BEAGLE BLACKJACK

Type in this program to make good use of the "SHAPE TABLE" file (actually a deck of cards!) on the Beagle Basic disk. Feel free to add enhancements such as betting, doubling down, cheating by looking at the deck, and so on. Call us when you're finished; Beagle Bros pays a hefty ¼% royalties on computer Blackjack games!

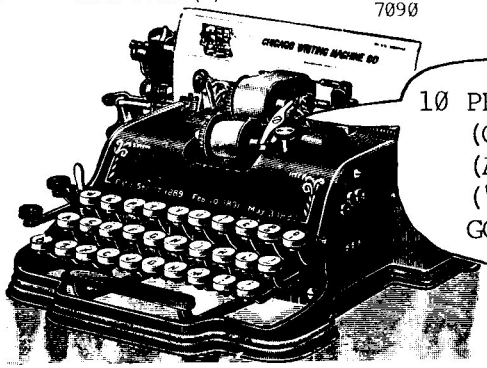
```
100  DIM DECK(51,1): TEXT : HOME
     : VTAB 9:PAUSE = 150
110  PRINT "B L A C K J A C K": PRINT
     : PRINT :PAUSE = 99: GOSUB 4
     000: INPUT "WHAT IS YOUR NAM
     E? ";NAME$: HGR
120  IF NAME$ = "" THEN  RUN
130  PRINT  CHR$ (4);"BLOAD SHAPE
      TABLE": POKE 232,0: POKE 23
     3,64: SCALE= 1
140  GOSUB 5000
150  POKE 28,42: CALL  - 3082: HCOLOR=
     3: HOME : GOSUB 160: GOSUB 1
     60: GOTO 360
160  GOSUB 6000
170 COUNT(1) = COUNT(1) + 1
180 X = (COUNT(1) - 1) * 50 + 20:
     Y = 10: GOSUB 1000
190  IF CARD > 10 THEN CARD = 10
200  IF CARD = 1 AND ACE(1) = 0 THEN
     CARD = 11:ACE(1) = 1
210 PNTS(1) = PNTS(1) + CARD: IF
     PNTS(1) > 21 AND ACE(1) = 1 THEN
     PNTS(1) = PNTS(1) - 10:ACE(1
     ) = 0
220  IF COUNT(1) = 5 AND PNTS(1) <
      = 21 THEN COUNT(0) = 0: GOSUB
     250: GOTO 7070
225  IF PNTS(1) > 21 THEN  VTAB 2
     2: PRINT "<BUST>":PAUSE = 99
     : GOSUB 4000
230  RETURN
250  GOSUB 6000
260 N1 = N1 + 1
270  IF N2 = 2 THEN COUNT(0) = CO
     UNT(0) + 1
280 COUNT(0) = COUNT(0) + 1
290 X = (COUNT(0) - 1) * 50 + 20:
     Y = 80: GOSUB 1000
300  IF CARD > 10 THEN CARD = 10
310  IF CARD = 1 AND ACE(0) = 0 THEN
     CARD = 11:ACE(0) = 1
320 PNTS(0) = PNTS(0) + CARD
330  IF PNTS(0) > 21 AND ACE(0) =
     1 THEN PNTS(0) = PNTS(0) - 1
     0:ACE(0) = 0
340  IF COUNT(0) = 5 AND PNTS(0) <
      = 21 THEN 7120
350  RETURN
360 COUNT(0) = COUNT(0) + 1
370 X = 20:Y = 80: GOSUB 2000: GOSUB
     3000: GOSUB 250
380  IF PNTS(1) > 21 THEN COUNT(0
     ) = 0: GOSUB 250: GOTO 7120
390  POKE  - 16368,0: VTAB 22: PRINT
     "ANOTHER CARD (Y/N)? ";: GET
     A$: PRINT A$: HOME
400  IF A$ =  CHR$ (27) THEN  HOME
     : VTAB 21: END
410  IF A$ = "N" OR A$ = "n" OR A
     $ =  CHR$ (32) THEN 7000
420  IF A$ <  > "Y" AND A$ <  > "
     y" THEN 390
430  GOSUB 160: GOTO 380
1000  GOSUB 2000: ROT= 0: XDRAW C
     ARD AT X + 1,Y + 1: XDRAW SU
     IT AT X + 1,Y + 9
1010  ROT= 32: XDRAW CARD AT X +
     39,Y + 49: XDRAW SUIT AT X +
     39,Y + 41: RETURN
2000  HCOLOR= 3: FOR I = X TO X +
     40: HPLOT I.Y TO I,Y + 50: NEXT

2010  HCOLOR= 0: HPLOT X - 1,Y -
     1 TO X + 41,Y - 1 TO X + 41,
     Y + 51 TO X - 1,Y + 51 TO X -
     1,Y - 1: RETURN
3000  HCOLOR= 0: FOR I = X + 1 TO
     X + 40 STEP 3: HPLOT I,Y TO
     I,Y + 50: HPLOT I + 1,Y TO I
      + 1,Y + 50: NEXT : RETURN
4000  POKE  - 16368,0: FOR I = 1 TO
     PAUSE: IF  PEEK ( - 16384) <
     128 THEN  NEXT : RETURN
4010 I = PAUSE: NEXT : RETURN
```

```
5000  FOR I = 0 TO 51:DECK(I,0) =
    0: NEXT
5010  FOR I = 0 TO 51
5020 J =  INT ( RND (1) * 52): IF
    DECK(J,0) THEN 5020
5030  HOME : VTAB 22: HTAB 1 + I -
     INT (I / 2) * 2: PRINT "<SH
    UFFLING>";
5040 DECK(J,0) = 1:DECK(I,1) = J:
     NEXT :DPTR = 0: HOME : RETURN

6000  IF DPTR > 51 THEN  GOSUB 50
    00
6010 CARD = DECK(DPTR,1):DPTR = D
    PTR + 1
6020 SUIT =  INT (CARD / 13 + 14)

6030 CARD =  INT (((CARD / 13) -
     INT (CARD / 13)) * 13 + 1.0
    5)
6040  RETURN
7000  IF PNTS(0) > 21 THEN 7070
7020  IF PNTS(0) >  = PNTS(1) THEN
    7120
7030  IF N1 = 1 THEN COUNT(0) = 0
7040  IF PNTS(1) > PNTS(0) THEN  IF
    PNTS(0) < 18 THEN N2 = N2 +
    1: GOSUB 250
7050  IF PNTS(1) > PNTS(0) AND PN
    TS(0) >  = 17 THEN 7070
7060  GOTO 7000
7070  VTAB 22: PRINT "<";NAME$" W
    INS>":PAUSE = 150: GOSUB 400
    0
7080 GAMES(1) = GAMES(1) + 1
7090  VTAB 24: PRINT "SCORE: DEAL
    ER ";GAMES(0)", ";NAME$;" ";
    GAMES(1);:PAUSE = 300: GOSUB
    4000: HOME
7100 COUNT(1) = 0:COUNT(0) = 0:PN
    TS(0) = 0:PNTS(1) = 0:N2 = 0
    :N1 = 0:ACE(1) = 0:ACE(0) =
    0
7110  GOTO 150
7120  IF PNTS(1) = PNTS(0) THEN  VTAB
    22: PRINT "<PUSH>":PAUSE = 1
    50: GOSUB 4000: GOTO 7090
7130  VTAB 22: PRINT "<DEALER WIN
    S>":PAUSE = 150: GOSUB 4000:
    GAMES(0) = GAMES(0) + 1: GOTO
    7090
```



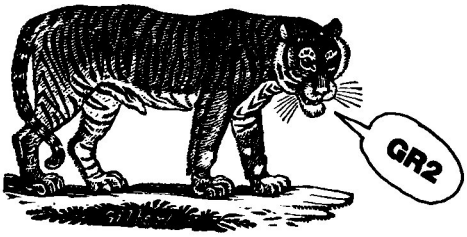
NO-RUN BOOT TIP

Did you know that if you press control-C while booting, the disk's Hello program won't be executed? We use this trick to keep Integer BASIC from loading when we boot the System Master disk, and to prevent GPLE from loading too.

GPLE NO-NO

Listen to us and there won't be any trouble— NEVER INIT A DISK WITH GPLE.48 ("PLE.48") IN MEMORY. Instead, before INITializing, re-boot and use the previous tip to prevent GPLE from loading. Thank you.

Strange problems if you don't follow our advice here.



GR2?

GR2 would have problems being a legitimate command. See page 9.

It's easy to find out what page 2 of lo-res graphics looks like. You don't even have to know any pokes for this one. Just type "**HGR2 : GR**". These commands uncover a minor a bug in Applesoft. GR doesn't set the display to page 1 as TEXT and HGR do; it assumes you're already on page 1. By the way, if you're still looking at that colorful garbage and want to get back to normal text page 1, just type "**TEXT**".

ESC-ARROWS

On an Apple IIe, hitting ESC followed by the four arrows is just like ESC followed by I, J, K, or M. Don't you wish they had put the Up-Arrow above the Down-Arrow, instead of to the *right* of it? Why don't you call Apple Inc. for us and complain? (Ask for Steve.)

GET IT RIGHT

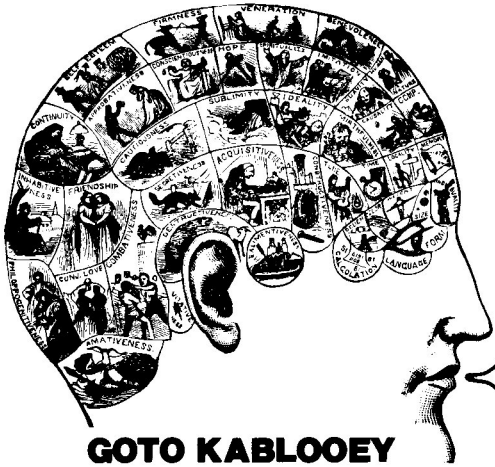
When your program requests numerical input via the GET command, use **GET A$: A=VAL(A$)** rather than **GET A**. This prevents an unwanted ?Syntax Error if a non-number is entered.

IIe AUTO-SHIFT

Here's another small difference between the II+ and the IIe. On a II+, if you type control-2 you get a 2. Likewise a control-6 yields a 6 (there's no such animal as a control-*number*). BUT on a IIe, control-2 comes out control-@ and control-6 is control-caret. No shift necessary.

LOWER-CASE POKER

Do a POKE 243,32 and list or run a program. With lower-case hardware, you will see that all upper-case has been demoted. If you don't have lower-case capabilities, you're caught looking at a bunch of garbage.



GOTO KABLOOEY

Everyone knows that any number larger than 63999 after a GOTO will result in a ?Syntax Error. But I'll bet you *didn't* know that if the number is in the range 437760 thru 440319...

CHANGE OF ADDREFP

If you've ever wanted to change the starting address for your Applesoft programs (so you could use text page 2, or for some other exotic reason), here's how it's done.

Make this the first line of the program to be moved. SAVE before you RUN.

The START of 16384 was chosen arbitrarily. FP will set the starting address back to normal, 2049.

```
Ø START = 16384: IF  PEEK (1Ø3) +
     PEEK (1Ø4) * 256 <  > START
     THEN  POKE 1Ø3,ST -  INT (S
    T / 256) * 256: POKE 1Ø4, INT
    (ST / 256): POKE ST,Ø: PRINT
     CHR$ (4)"RUN THIS PROGRAM"
```

GRAFFICKY TEXT

This program shows how you can make really effective bar graphs on the text screen.

```
1Ø  DIM N(23): HOME : NORMAL
2Ø  FOR I = 1 TO 23: HTAB 4Ø: PRINT
     ":";: NEXT : INVERSE
3Ø RAN =  INT ( RND (1) * 23)
4Ø N(RAN) = N(RAN) + 1
5Ø  VTAB RAN + 1: HTAB 1: PRINT  SPC(
     N(RAN)) CHR$ (RAN + 65);
6Ø  IF  PEEK (36) THEN 3Ø
7Ø  NORMAL : VTAB 22: PRINT  CHR$
     (7): END
```

Beagle Basic Index

Applesoft 2
BASIC, What is it? 2
BELL 10
Bell, Modify 11
Changing Commands ... 13
CLDN 10
CLLN 10
CLRKEY 9
Command Editor 13
Command Table 13
Create Newbasic 4
CRSD 10
CRSL 10
CRSR 10
CRSU 10
Cursor Commands 10
DOS 3
Edit 12
ELSE 6
Error Message Editor 15
Escape Cursor 11
Exiting Newbasic 19
Find 16
FP 19
Free Characters 13
GOTO/GOSUB 11
GPLE 20
G2 9
Hi-Res Commands 10
HSCRN 8
INT 19
Integer BASIC 2
Inverse REMs 5
Keychart 12
List Formatter 5
Main Menu 4
Move Mode 13
MIX 10
MODE 10
New BASIC Commands .. 6
Newbasic 3
Newbasic Loader 3
Normalize 17
Other Features 11
PAGE 10
Print Old/New 17
Print Screen 17
Quit 11
Reset 19
RESL 10
Save Newbasic 11
SCRLDN 8
SCRLUP 10
Sort 16
Spare Characters 13
Standard 17
SWAP 7
Text & Bell Commands .. 10
TONE 7
TXT2 9

PRINTED IN SAN DIEGO

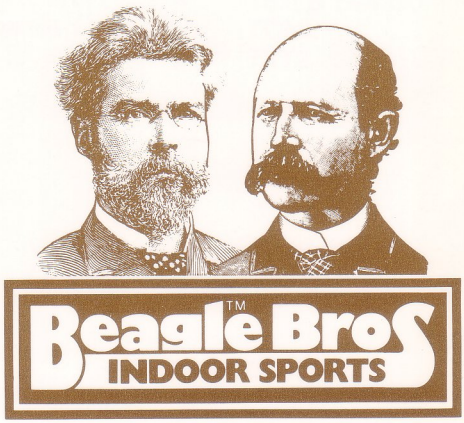
Warranties and Limitations of Liability

Beagle Bros warrants that this product will perform as advertised. In the event that it does not meet this warranty or any other warranty, express or implied, Beagle Bros will refund the purchase price of this product.

BEAGLE BROS' LIABILITY IS LIMITED TO THIS PRODUCT'S PURCHASE PRICE. In no case shall Beagle Bros or the author be liable for any incidental or consequential damages, nor for any damages in excess of the purchase price of this product.

This disk includes software, APPLE DOS 3.3, owned by Apple Computer, Inc. This software is used under license from Apple. Apple makes no warranties, either express or implied, regarding Apple DOS, its merchantability, or its fitness for any particular purpose.

"APPLE" is a registered Trade Mark of Apple Computer Inc.



Published by BEAGLE BROS INC.
4315 Sierra Vista, San Diego, California 92103
619-296-6400