Open-Apple

April 1985
Vol. 1, No. 3

Releasing the power to everyone.

Demoralized Apple II division announces enhanced IIe and an inventory backlog

It's important to remember, when dealing with large organizations such as Apple, Inc., that they are not single, indivisible monoliths. They are social organizations made up of many people. Those people don't all think alike. They meet, they argue; if a consensus develops, they decide. They are real people, with very different backgrounds and ideas, and with real feelings. Some of them are turkeys and some of them soar with eagles.

Though you often can't tell it from Apple's official corporate line, there are many people within Apple itself who feel as strongly about the Apple II as we do. Right now they need our support more than our sarcasm—morale in the Apple II division of Apple, Inc. appears to be at a new low point. In addition to Steve Wozniak's virtual departure last month, several key managers and engineers in the Apple II division have left as well. According to press reports, Apple's January stockholder's meeting was particularly demoralizing for the people in the Apple II division, since they had just finished their best quarter ever and were almost totally ignored.

To make matters worse, Apple announced on March 7 that, for the first time in its history, it will close down all of its manufacturing plants for a week this spring in an effort to work off excess inventories. Apple IIs are manufactured at plants in Dallas and Cork, Ireland. Some circuit boards for the IIs are manufactured in Singapore. The closing also includes Apple's Macintosh plant in California.

Ironically, March 7 was also the day Apple finally introduced its enhanced version of the Apple IIe. (Is this what they mean by "event marketing"?) Reports of Apple's imminent announcement of the enhanced IIe have been in the computer press for weeks. In fact, one of our subscribers wrote to say that he bought a new IIe in Ames, Iowa on March 2 and got the enhanced version. At least it's not vaporware.

The IIe enhancment package makes the IIe more closely compatible with the IIc and, believe it or not, the II-plus. On the other hand, many software packages written for the original IIe are incompatible with the enhancements.

Four chips make all the difference. An enhancement kit for older IIes is available from dealers for $70, installed. However, you don't get to keep your old chips and Apple won't let dealers de-enhance your IIe, so consider the upgrade carefully.

Apple sent out samples of this enhancement kit last fall to software developers so that we could test the compatiblity of our products. Only one of the four chips in the kit stayed inside my IIe after testing was completed. Though my own software checked out okay, there were several incompatibilities between the enhancement and other software I use frequently.

The one chip that stayed is a 65C02 microprocessor. It replaces the older 6502. The new chip does everything the old one will do and some new stuff besides. There is a bit of copy-protected software around that won't run on the 65C02—the problem is related to the protection scheme used.

Two of the chips in the kit are new ROMs. The machine language programs built into your computer (Applesoft and the Monitor) are changed when the new ROMs are used. In general, the changes are most welcome—if anything, they don't go far enough. I'll cover them in greater detail later in this letter. Their main function is to make the IIe a true 80-column, upper-/lower-case computer. Or, from a nastier viewpoint, you could say the new ROMs get the most notorious bugs out of the Apple IIe 80-column firmware.

But there is a price to pay for this bug stomping. Programs written to avoid these bugs in the original IIe may no longer work correctly. For example, the telecommunications software I use, *ASCII Express*, version 4.20, won't run on an enhanced IIe. So my enhanced ROMs went back into the box they came in, awaiting the day I need them bad enough to get an *ASCII Express* update or a new communications program.

The fourth new chip in the enhanced IIe is the character generator. This chip determines what the characters that appear on your display screen look like. The new chip includes "mousetext" characters in the 80-column character set where inverse capitals should be. These characters allow programmers to create Macintosh-like windows on the Apple II *text* (not graphics) screen. The mousetext characters are little graphics symbols—open- and solid-Apples, an hourglass, checkmarks, and the like.

The problem is that on most 80-column software written before the IIc was introduced, the mousetext characters appear on your display screen where inverse capitals are supposed to be. This occurs even with best-selling software such as Apple Writer and VisiCalc. If you are familiar with Apple Writer, imagine what the status line at the top of the screen looks like with all the capital letters replaced with graphics symbols. The letter under the cursor is also unrecognizable when it is a capital. Thus the mousetext characters make the enhanced IIe incompatible with a lot of pre-existing 80-column software.

Apple's press release on the enhancement package says that "over 95 per cent of the Apple IIe software currently available on the market will run on the enhanced IIe." I questioned Linda Merrill, the Apple II specialist in Apple's corporate relations department, very closely on this point—since a much lower proportion of my own software library worked without interference from either the mousetext characters or the ROM modifications.

Merrill said the key words in the press release are "currently available." Apple's 95 per cent figure doesn't include any programs that are no longer available or any previous versions of today's programs. What this means is that the 95 per cent figure is only relevant to people buying an Apple for the first time.

It's high time Apple's marketing mavens discovered that many of their customers aren't brand new. A very high proportion are either adding Apple IIs to an existing base (almost all schools would be in this category), or are buying a new computer to replace an older one. To those who are adding IIes to an existing base or upgrading an older IIe, Apple's 95 per cent compatiblity figure is meaningless, if not downright deceptive. Don't make the mistake of using it as an indicator of how much of your *existing* software library will run without interference.

When I asked Merrill about this, she pointed out that updates that solve the incompatibility problems are available for many older programs.

In general, you can expect most 40-column Basic and assembly language programs to run on the enhanced IIe without a hitch. Most 80-column programs written for the original IIe, on the other hand—particularly assembly language programs—have at least the nuisance of mousetext characters showing up where inverse capitals should be—a few have more serious problems.

Of course, the incompatiblity problems the enhanced IIe poses are nothing compared to what the world will experience if the continuing rumors about an "Apple II" that won't run DOS 3.3 come true. This month I managed to locate the source of these rumors—Apple's own *Apple II Forever College* (talk about double-speak!), where software developers with 3 days, $500, and a willingness to sign a non-disclosure agreement can go and see visions of the future, among other things.

Apple manuals liberated

I often give Apple, Inc. a hard time, so let me stop right here and hug the corporate body, remind it that I really do love it deep down inside, and praise it for some recent changes.

Some of the best books in print about the Apple II have also been among the hardest to obtain the last couple of years. These books are Apple's own manuals. At one time, new Apple owners got a virtual library free with their machines—an Integer Basic tutorial, an Applesoft tutorial, an Applesoft reference manual, a DOS manual, and a reference manual for the Apple II itself.

While none of these books held the last word about the Apple, together they constituted the finest documentation available for a personal computer. In a time when purchasers of other computers were loudly lamenting the poor instructions that came with their machines, the buyers of Apples were quietly learning how to use and program a real computer. This is one of the many reasons Apple thrives today while so many of its competitors have disappeared.

By the time the Apple IIe was released, most buyers were more interested in running spreadsheets and word processors than in learning Basic. Consequently, Apple decided to make the heavy-weight manuals extra-cost options and to pack light-weight owners/users manuals with the computers. This made business sense at the time, but it had the effect of making Apple's excellent manuals nearly impossible to obtain.

The reasons for this are complicated. First of all, only Apple dealers were allowed to sell them. Next, Apple's books were much more expensive than most computer books. Consequently, many dealers, some of whom weren't in the book business to begin with, didn't stock them. Dealers were also reluctant to place special orders for customers because Apple only sold books in packages of five of each title.

Those days are done. Apple and Addison-Wesley Publishing Company have reached an agreement whereby Addison-Wesley will distribute Apple's books. They will now be available to bookstores, as well as computer stores, in any quantity. Individuals can even buy direct from Addison-Wesley (Addison-Wesley, Reading, MA 01867, 800-238-3801; by credit card, you pay the postage; by check, they pay the postage).

Available right now are the Applesoft Tutorial (with disk), $29.95; Applesoft Reference Manual, $22.95; and Basic Programming with ProDOS (with disk), $29.95. Available sometime within the next few months will be the ProDOS Technical Reference Manual, the Apple IIe Reference Manual, and the Apple IIc Reference Manual. Prices on these three haven't been set yet, but the people I talked to at Apple and Addison-Wesley thought they would be less expensive (probably $20 to $25) than currently (the IIc manual costs $50 today, at Apple dealers nowhere).

Unavailable from Addison-Wesley are the DOS 3.3 Manual and the ProDOS Users Manual (each includes a system master disk). Apple has made arrangments for dealers to special order these one copy at a time, however, through service parts. Talk to your dealer's technicians.

More about mousetext

Mousetext characters premiered on the Apple IIc. Other than the system utilities program that comes with the IIc, however, very little software published to date actually uses these characters. Their purpose is to give the Apple II family the potential of a command structure and user interface similiar to that of the Macintosh—windows and pull-down menus.

Software developers haven't rushed to write Macintosh-like programs for the Apple II because, up till now, only the IIc could run such software. Most developers try to write for the widest possible range of machines. Even now, with new IIes having mousetext built in and old IIes having the potential of using it, there are more than a million Apple II-pluses and compatibles in the world that will never be able to display these characters. Many developers are loathe to cut this market out of their product's potential sales.

On the other hand, a few developers follow the theory that it's the people who are buying new computers who buy the most software. For example, *AppleWorks* doesn't seem to be suffering much from its inability to run on the Apple II-plus.

Investigating Apple's characters. From the letters I receive I can detect a lot confusion among ***Open-Apple*** readers about the character sets, including mousetext, available on the various types of Apple II. Let's study the whole situation, beginning with how characters are displayed on the Apple II-plus.

What you see on your computer's screen depends on what values are stored in a special area of your Apple's memory. This area, which is called the *display page*, extends from byte 1024 (in hex that's $400) to byte 2047 ($7FF). To prove this to yourself, type in the following commands. (If you have a IIe or IIc, press escape/control-q to make sure you are in 40-column mode (flashing checkerboard cursor) before you begin.)

```
HOME
POKE 1024,0
```

An inverse @ sign will appear in the upper-left corner of your screen. When a byte in the display page holds a zero, an inverse @ sign appears at that byte's position on your screen.

Since the screen is 40 characters wide, it follows that poking zero into the byte 40 characters beyond byte 1024 should put another inverse @ sign in the second line. Try it.

```
POKE 1024+40,0
```

Surprise. The @ sign shows up in the ninth line, not the second line. This is because the screen "mapping" isn't "linear", whatever that means (it means Wozniak saved a chip or two in his original design for the Apple II).

The easy way around the non-linear mapping problem is to build an array that holds the starting address of each line on the screen. The following program segment will do that:

```
100 DIM LADR(23)
110 FOR I=0 TO 23 : READ X : LADR(I)=X : NEXT
120 DATA 1024, 1152, 1280, 1408, 1536, 1664, 1792, 1920, 1064, 1192, 1320, 1448,
      1576, 1704, 1832, 1960, 1104, 1232, 1360, 1488, 1616, 1744, 1872, 2000
```

Now we can poke or peek at any byte on any line quite easily. If L is the line we want and C is the column we want, then *POKE LADR(L)+C,0* will put an inverse @ sign there. Now add the following lines to those above and run the program:

```
10  REM ***  ASCII DEMO  ***
200 HOME
210 FOR L=8 TO 16
220 POKE LADR(L)+19,0
230 NEXT
240 VTAB 23
```

This program will put a line of inverse @ signs down the middle of your screen. Press the return key a few times. Your line of @ signs will move up and off the screen. What's going on here is that the *scroll* routine in the Monitor is moving around those zeros you poked into the display page. The Monitor moves them in such a way that they appear to scroll up the screen.

Now let's modify our previous program a little. Keep lines 100 through 210 just as they are, but delete 210 through 240 and add the following:

```
210 FOR S=0 TO 7
220 L=S*2 + 6
230 FOR C=0 TO 31
240 POKE LADR(L)+C, S*32 + C
250 NEXT C,S
```

Save this program (we'll use it again later), then run it. You'll get the following eight-line display on your screen. Four of the eight lines start with our old friend the @ sign and are full of capital letters and symbols. The top two lines are displayed in inverse, the next two are flashing, and the bottom four are normal.

```
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_      <- inverse
 !"#$%&'()*+,-./0123456789:;<=>?      <- inverse
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_      <- flashing
 !"#$%&'()*+,-./0123456789:;<=>?      <- flashing
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_      <- normal
 !"#$%&'()*+,-./0123456789:;<=>?      <- normal
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_      <- normal
`abcdefghijklmnopqrstuvwxyz{|}~       <- normal
```

If you are using a IIe or IIc, the bottom line of the eight will have lower-case letters in it. You may also see lower-case letters if you are using an older Apple with a "lower-case chip." Otherwise, the bottom line will have symbols and numbers in it—lower-case letters weren't supported on the original Apple II and II-plus.

What our little program has done is poke all the possible values between zero and 255 into consecutive locations on our screen. The first line shows the values from 0 to 31, the next from 32 to 63, and so on.

Split the display into two parts in your mind. The first part is the top four lines, which contain special characters. The second is the bottom four lines, which contain normal characters. The top four lines hold the values from 0 to 127; the bottom four hold 128 to 255.

All about ASCII. The display on your screen shows the structure of the American Standard Code for Information Interchange, or ASCII. ASCII associates the upper- and lower-case letters of the alphabet, the ten numerical digits, an assortment of symbols, and an assortment of control characters with the numbers between 0 and 127. The top line of each group of four holds the 32 ASCII control characters. The second line holds 32 digits and symbols, the third holds 32 capitals and symbols, and the fourth holds 32 small letters and symbols.

On some computers, the 32 control characters also display special symbols. On the Apple, the characters that are displayed when values in the control-character range are stored in the display page (something that normally happens only by mistake) are duplicates of the capitals.

So far we've uncovered one difference between the Apple IIe and IIc and earlier Apples. The newer machines have lower-case letters as part of their normal character set. Older Apples do not, although modification packages are available to fix this.

Now let's explore an even bigger difference. If you have a IIe or IIc, enter the following command while the eight lines are still displayed on your screen:

```
POKE 49167,0          ($C00F) turns on alternate character set
```

This command turns on your Apple's *alternate* or *80-column character set.* The Apple IIe and IIc actually have *two* entire sets of characters, where earlier models had just one. The difference between the two sets is subtle, but you should be able to find it. The flashing characters have disappeared.

What you see in their place depends on what kind of computer you are using. On the original IIe, the flashing characters are replaced by upper- and lower-case inverse characters. Thus, on the original IIe, the alternate character set provides two complete groups of ASCII characters, one group normal, one group inverse.

To turn the alternate character set back off, use:

```
POKE 49166,0          ($C00E) turns on standard character set
```

(You can also determine from inside a program which set of characters is currently displayed. *IF PEEK(49182)< 128 THEN* the standard set is on. In hex that location is $C01E.)

Now here's the part I don't understand. Apple decided to add 32 mousetext characters to the alternate character set of the IIc and enhanced IIe. Instead of replacing one or the other of the two groups of control characters with mousetext, Apple's engineers put it over the top of the inverse capitals.

On a IIc or enhanced IIe, the alternate character set looks like this:

```
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_      inverse
 !"#$%&'()*+,-./0123456789:;<=>?
------mousetext characters------
`abcdefghijklmnopqrstuvwxyz{|}~
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_      normal
 !"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
`abcdefghijklmnopqrstuvwxyz{|}~
```

This scheme breaks the rules of ASCII and seems pretty dumb to me. Programmers putting characters directly on the screen have to use control characters to get inverse caps and inverse caps to get mousetext. If any of you loyal readers know why Apple put the mousetext characters where they did, it would sure make me feel better to hear about it. It was by placing them where inverse capitals should be that Apple's engineers managed to make the IIc and enhanced IIe incompatible with most 80-column, pre-IIc software.

Those of you who don't have IIcs and have never seen the mousetext characters might like to look at the following table, which I stole off the back of the latest Beagle Bros *Peek, Pokes and Pointers* chart. The characters include a solid Apple and an open Apple, a mouse pointer, an hourglass, inverse and normal check marks, a running human (F and G together; usage unknown), and various kinds of arrows, lines, and boxes for making window frames.

@=	H= ←	P=	X=
A=	I= ...	Q=	Y=
B=	J= ↓	R=	Z=
C=	K= ↑	S= —	[=
D= ✓	L=	T=	\=
E=	M=	U= →	]=
F=	N=	V=	^=
G=	O=	W=	_=

To print mousetext characters from within a Basic program, begin by turning on the 80-column mode with a *PR#3.* Then print an escape with the command *PRINT CHR$(27).* Use the *INVERSE* command and print the letter or symbol shown on the above chart to get the mousetext character you want. You can also print in inverse capitals and lower case, if you like. Use the *INVERSE* command just as before, but first turn the mousetext feature off with a *PRINT CHR$(24)* (control-X). (Note: You cannot enter the escape and control-X that turn mousetext on and off from the keyboard. They must be printed to the screen by a program.)

From 80-column mode it is also possible to change to inverse characters by printing a control-O (*CHR$(15)*); change back to normal with control-N (*CHR$(14)*). The following program uses this trick to allow you type the mousetext human at various places on your screen:

```
10  REM *** HUMANTEXT DEMO ***

100 HOME : PRINT CHR$(4);"PR#3" : PRINT
110 MT$=CHR$(27);CHR$(15);  : REM turn on mousetext and inverse
120 NR$=CHR$(24);CHR$(14);  : REM turn off mousetext and inverse
130 MAN$=MT$+"FG"+NR$       : REM this string prints the mousetext man

140 VTAB 23 : INPUT X,Y     : REM get screen coordinates from human at keyboard
150 IF X<1  THEN X=1        : REM check to make sure they are legal
155 IF X>79 THEN X=79
160 IF Y<1  THEN Y=1
165 IF Y>24 THEN Y=24

170 HTAB X : VTAB Y         : REM move there and print
180 PRINT MAN$
190 GOTO 140
```

By assigning each mousetext character its own string variable, and including the control characters for turning mousetext on and off as part of the string, you can print mousetext very easily.

More about the new IIe ROMs

The primary advantage of the enhanced IIe ROMs is improvements in the 80-column firmware. We've just seen how characters are displayed on a 40-character screen. To fully explore the new enhancement, lets look at how Apple's 80-column screen works.

If you have an Apple IIe or IIc, enter *PR#3* and rerun our little ASCII DEMO program (you saved it like you were supposed to, right?). You should notice two differences from how it looked before. First, there will be blank spaces between all the characters; secondly, the alternate character set will be displayed automatically without the need for any special pokes. Entering Apple's 80-column mode always turns on the alternate character set. You can, of course, switch back to the standard set with *POKE 49166,0,* as mentioned earlier, if you want to.

The "display page" memory that our program pokes the ASCII values into resides between byte 1024 and byte 2047, as mentioned earlier. This space encompasses 1,024 bytes. Since our screen is normally 40 columuns wide by 24 lines high, 960 (40*24) bytes of this area are needed to represent the 40-column screen. Obviously, when we switch to 80-columns, we'll need another 960 bytes.

Rather than using some of the 64K of memory already built in for this, Apple's engineers decided to use additional memory on a card in the auxiliary slot for storing this second 960 bytes. When 80-column mode is turned on, the display circuitry in the Apple uses the 960 bytes in main memory for the odd columns of the 80-column screen and the 960 bytes in auxiliary memory for the even columns (assuming the first column is zero).

Since our program only pokes numbers into the main display page, then, our ASCII chart shows up with spaces between the characters. When 80-column mode is on, access to the two 80-column pages is gained with pokes to 49236 ($C054, main memory on) and 49237 ($C055, aux memory on).

To fix our "ASCII DEMO" program so that it works correctly in 80-column mode, make the following changes:

```
change the upper end of the loop in line 230 to "15" (not 31).
add "*2" at the end of line 240

add:
235 POKE 49237,0 : REM do even columns
245 POKE 49236,0 : REM do odd columns
247 POKE LADR(L) + C, S*32 + C*2 + 1
```

After making the changes, you have a program that will display the ASCII chart correctly on the 80-column screen.

Something flashy. Now, just for fun, press escape-4 or escape-control-q to get a 40-column screen and run the program again. Don't panic. The interference you see is normal. In 40-column mode, the switches at 49236 and 49237 switch

between *display page 1* and *display page 2*. Display page 2 resides between bytes 2048 ($800) and 3071 ($BFF). When display page 2 is turned on, what you see on your screen in a *meaningless* ASCII representation of the "tokens" in your Basic program. Display page 2 is supported by the Apple *hardware*, but not by the Apple *software* (that is, the Monitor). Consequently, it is rarely used as a display page. Instead, this memory area is widely recognized by assembly language programmers as the place where "free memory" starts. Basic stores your programs here.

While we are detouring from our main subject, let's also answer the question of what happens to the extra memory inside the display pages. As we saw earlier, there are 1,024 bytes inside each display page, but only 960 of these are used to hold characters on the screen. The 64 extra bytes are in eight scattered groups of eight bytes and are reserved for the exclusive use of programs controlling devices connected to your computer. These bytes are usually referred to as the *I/O scratchpad* RAM or the *screenholes*. Don't mess with them unless you love trouble.

At any rate, now you know how the 80-column screen works. You also probably realize that putting text onto the screen yourself gets quite messy. This is why computer manufacturers write *operating system* software for their computers to do this work. In the original 40-column Apple II, the routines that display charcters on the screen are a part of the Monitor. With the 80-column IIe, however, there wasn't enough room in the space allocated to the Monitor, so Apple built in additional ROM, made it look like a device in slot 3, and called it the *80-column firmware*. Note that this firmware is stored in the ROMs that come with the Apple IIe and IIc and not, as many people think, in the "80-column card" (really nothing but a memory card) that goes in the IIe's auxiliary slot.

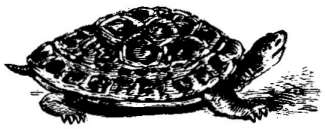
The built-in routines are used by Basic and by many assembly language programmers. However, high-performance commercial software usually bypasses the Apple's built-in routines in favor of specially-written ones. The original 40-column routines didn't have any provisions for scrolling up, for example. Consequently, Apple II-plus programs such as VisiCalc and Apple Writer included their own "screen drivers" (software for displaying characters on the screen).

The 80-column routines built into the original IIe have given software developers fits. They are relatively slow. They disable interrupts for relatively long periods of time. Applesoft's HTAB, TAB, SPC and comma tabbing don't work right. The 80-column routines support extended escape sequences, but you can no longer GET an escape from within Applesoft. All of these problems have been solved in the enhanced IIe ROMs. We may even start to see high-performance software using Apple's firmware instead of special routines.

Applesoft and the Monitor accept lowercase input on enhanced IIes, just as on the IIc (it's about time!). The enhanced IIe Monitor also allows direct entry of ASCII characters into memory by preceeding the character with an apostrophe. You can, for example enter *300:'O'p'e'n'-'A'p'p'l'e* and have the ASCII values of those characters stored in memory. As mentioned here last month, the enhanced IIe Monitor also supports a mini-assembler and has a new search command.

Interrupt handling on the enhanced IIe is greatly improved. The operating system itself now saves the memory configuration at the time of the interrupt and turns on the main banks of memory before passing control to a program's interrupt handler. It also puts everything back like it was at the end of interrupt. This may be so much hocus-pocus to many of you, but it means that it's much easier for developers to write software for things like networks and high-speed modems.

It's difficult to say at this point what the full implications of the enhanced IIe software will be. We'll continue to study it and watch for further developments.



Go, Logo, Go

Support of the mousetext characters is built into Apple's 128K Logo. However, the technique isn't the same as that used with Basic. You cannot change the screen to inverse by printing a control-O, as you can from Basic or assembly language.

Instead, you access mousetext (as well as inverse characters) by printing ASCII codes. Use the primitive CHAR for this. Try these procedures:

```
TO MOUSETEXT
MAKE "CODE 192
REPEAT 32 [TYPE CHAR :CODE MAKE "CODE :CODE + 1]
PRINT "
END
```

```
TO HUMAN :POS
SETCURSOR :POS
TYPE CHAR 198 TYPE CHAR 199
SETCURSOR [0 22]
END
```

The MOUSETEXT procedure PRINTs the value of CODE as an ASCII character using Logo's CHAR primitive. CODE will start at 192 and work its way up to 223. The result will appear on your screen as the 32 mousetext characters. The Logo reference manual has a small chart of what CHAR will do on page 83. Ignore it and check out Appendix F (pages 281-283) instead. This shows what will acually be printed for each value given to CHAR. The main two ASCII groups of 128 I talked about in the previous mousetext article are reversed with this version of Logo.

The HUMAN procedure PRINTs the mousetext human at whatever screen coordinates you give. Note: Logo demands that you specify these coordinates in brackets, like this *HUMAN [10 10]*.

By the way, has it ever occured to you to write a procedure like this one?

```
TO E
EDIT [E MOUSETEXT HUMAN]
END
```

It's a nice procedure for developing programs. You simply type E and you'll have all the procedures you're working on loaded into the editor.

Another tip: Under most conditions on the Apple, you can stop a screen that is scrolling faster than you can read it by pressing control-S. Press control-S (or any other key) a second time to restart. However, this doesn't work with Apple's 128K Logo. Instead, control-S(PLITSCREEN) causes the standard Apple 4-line text window to appear at the bottom of the graphics screen. (Control-L gives you the text-windowless FULLSCREEN.)

But there *is* a key for stopping scrolling screens. It's control-W(HOA), just above the standard control-S on your keyboard.

Seen In Print

800K Apple II disk drives are a firm part of Apple's plans, according to documents obtained by *Micro Marketworld*, a publication for computer retailers and wholesalers. In its March 11 edition (page 5), the magazine said the drives would use 3.5 inch disks and be available in single- and dual-sided versions. The documents obtained by the magazine were "from a recent Apple Computer Inc. marketing strategy meeting."

"A former apple executive" reportedly confirmed for the magazine that the enhancements outlined in the documents were in the works at Apple.

"The papers make it clear that Apple will concentrate on developing—either internally or externally—compatible, add-on products for its AppleWorks package; mouse-based products that will take advantage of icon and pull-down menus; and networking products designed to run on a II version of AppleTalk," the magazine said. Also mentioned were 256K (expandable to 1 megabyte) RAM cards for both the IIe and IIc and a 10 megabyte hard disk for the IIc that would support ProDOS, Pascal, and CP/M. Interestingly, two things apparently not mentioned in the paper were DOS 3.3 and the 65816 chip.

How can the Sider hard drive be so cheap? I still don't know the whole answer, but the *Wall Street Journal* published an article on February 22 (page 29) about circuit design advances that are bringing down the price of hard drives. Xebec Corp, the company that makes the Sider, is highlighted in the article, but the Sider itself is not.

The Computer Shopper looks like an advertising rag; I've never given it much attention. Flipping through the April issue, however, I found a column by Don Lancaster, called **Ask the Guru**. Lancaster has written books on electronics and computers for years and is currently into Apple IIs and Apple Writer IIe, among other things. Very little of the editorial material in the *Shopper* relates to Apples, but what is there is surprisingly good. For example, Lancaster gives a fast and simple assembly language scheme for generating a list of 2.1 billion random numbers (page 94—Applesoft's RANDOM function can't do anywhere near this well). Also in the April issue—how to read mainframe 9-track tapes with an Apple II. (Computer Shopper, 407 S. Washington Ave., Titusville, FL 32796, $15/yr.)

Ask (or tell) Uncle DOS

Don't despair...

Since I was a kid, one of my favorite parts of life has been getting something in the mail. This newsletter business has put me in a postal paradise. I love getting your letters, and you've sent me several stacks of them. I've even been getting mail in my Source and CompuServe mailboxes after years of always finding them empty!

The biggest problem I have is that all your letters won't all fit in the time and space I have available. I can't make the type any smaller (several of you insist it's too small already), so I've been printing the letters that say or ask things that I think need to be said or asked.

I try to respond personally to the letters that don't make it into ***Open-Apple****. I'm sure there are a few of you out there who feel like you might as well have sent your letter to the Easter Bunny for all the response you've seen so far. In part that's because some of you ask* **hard** *questions, and I haven't figured out the answers yet.*

But, in truth, it's also because some of your letters are buried at the bottom of one or the other of the various stacks of paper in my office. Some weekend when my mother's in town she's probably going to sneak in here and throw everything out and that will be that.

I recommend you prepare any letter you send in on a word processor; save the file. That way you can send me a second (or a third, or a...) copy if I haven't responded in a few weeks. Questions that include stamped self-addressed envelopes or that appear in my electronic mailboxes almost always get an immediate reply; I try to scribble down an answer before I lose the envelope or the file.

At any rate, I just wanted you to know this is fun, and that I'm as reliable as the next human being but no match for the speed and accuracy of our silicon friends. Keep those cards and letters rolling in.

The hard part of hard disks

I read your review of the *Sider* hard disk with great interest. I was disappointed, however, that you didn't include any comment about whether there is any problem installing software programs on it, e.g. Wordstar. Apple's *Profile* and Quark's *QC10* come with software called the *Catalyst* for loading even copy-protected programs. Is the *Sider* capable of doing this with the help of a copying program?

Samuel Chu
Nashville, Tenn.

The Sider comes with an enhanced FID utility for moving DOS 3.3 programs and files to and from the hard disk. Apple's FILER utility for ProDOS works fine with a hard disk (well, as fine as it does for floppies, anyhow). The Sider does not come with any software for moving copy-protected programs. However, please understand that even the **Catalyst** *can't move* **all** *copy-protected programs to a hard disk, but just the handful that have been protected with the* **Catalyst** *protection scheme.*

Few commercial DOS 3.3-based programs were written with a hard disk in mind. Of course, if a program is on a protected DOS 3.3 disk, there's little hope of using it with a hard drive, anyhow. To use a hard drive, you must usually boot with the drive's version of DOS and then start up the disk by running a HELLO program. Few protected programs allow this. And, to use a hard disk effectively, a DOS 3.3 program must allow a user to specify volume numbers when accessing files in addition to slot and drive parameters.

Most ProDOS programs, on the other hand, are specifically designed to work with a hard disk. To this end, you'll find many of these programs aren't even copy protected, which makes moving them to a hard disk quite simple. **AppleWorks** *and the ProDOS version of* **Apple Writer** *are two programs in this category. (Note: to get* **Apple Writer** *started once you move it to a hard disk, you must first set the prefix to the subdirectory the* **Apple Writer** *files are in, then enter "-AW.SYSTEM".)*

I suspect Wordstar could be configured to run on a hard disk, but I don't know enough about CP/M to PIP.

Sider sounds

In your review of the Sider hard disk you state that, in general, you're pleased with the unit, aside from the documentation. Enclosed is an article that appeared in the February 1985 issue of *Minn'app'les*, the newsletter of the local Apple users group. The reviewer had a few problems with his unit that you did not experience. I thought you'd find it interesting.

Marc S. Renner
St. Paul, Minn.

Your reviewer returned his Sider because, "I am one of the ten per cent of the population acutely sensitive to some kinds of noise." The Sider is not a silent partner. It has a disk inside it that is spinning constantly and I think there's a fan in there somewhere too. Unlike your reviewer, I prefer an environment with a lot of white noise so that I can't hear the buffalo chips drop in my otherwise quiet Kansas setting. If you are the kind of person who can't stand the noise a fan makes, you wouldn't like the Sider or, I suspect, any other hard drive.

Save DOS 3.3 and more

I'm writing to tell you that I've joined the campaign to save DOS 3.3.

I do have a few other campaigns in mind, they all have to do with enlightening those people who write software for the Apple IIs.

I am sure if informed they would write software that would allow:

1. Use of more than 143K of storage per file.

2. Use of more than 48, or 64, or 128K of RAM (many of us have much more, but only Magi-Calc lets us use it without a preboot).

3. Use of more than 132 columns when printing (a 15 inch printer will do almost twice that in compressed mode).

William L. Terry
Pleasanton, Calif.

A flood of 128K questions

1. I have a 128K IIe. How can I use some of its memory as a ramdisk?

2. Where can I get information on using pathnames with ProDOS? Apple doesn't seem to divulge such things; at least the dealer doesn't know.

3. Why is a *soft switch* used to change from text to graphics, but a *pointer* used to switch from screen to printer? If I understood this, I might be able to figure out how to switch to high-res mixed text and graphics on page 2. Or at least know why it's not possible.

4. I'm also confused on just how the Apple can use its extra 64K of memory without losing itself in the wilderness. It can't look at more than 64K addresses at one time, because it only has a 16-bit address bus. So if it is looking at a location in one 64K bank, it must be blind to things in the other bank. But it can't lose sight of its operating system, its application program, or its I/O section. How does it work?

T. Stewart Harris
Stamford, Conn.

1. A ramdisk is built into ProDOS. It has the volume name /RAM. Boot up a ProDOS disk and then enter CATALOG /RAM *and you'll get a glimpse of it, but it will be empty. You can use Apple's FILER to transfer files over to it. Ramdisks have also been designed for DOS 3.3. Beagle Bros sells one called* DiskQuik *and there are others.*

2. Actually, Apple has published quite a bit on pathnames. Ask your dealer's technical staff, not salesmen, for this kind of information. If you bought your IIe within the last year, you got a manual in the box with the disk drive called the ProDOS User's Manual. *Chapter 4 of this book describes pathnames. If you bought your IIe earlier, you'll have to buy this book. Your dealer's techs can get it for you through service parts.*

3. Switching from text to graphics involves telling the Apple's hardware *to change the signal being sent to the display screen. You talk to hardware by poking soft switches. Switching from the screen to a printer involves telling the Apple's operating system* software *to send output someplace different. The software does this by changing internal pointers. I used to think you could get mixed text and graphics on high-res page 2, but only while the 80-column firmware was active. I would do a PR#3 to turn on 80-columns, HGR2 to get to high-res page 2,* POKE 49235,0 *to turn on mixed text and graphics, and press return a bunch of times to get the cursor into the window. Further testing, however, has convinced me that even though I expressly said HGR2, what is displayed on the screen is hi-res page 1. With 80-column mode active, you can't display hi-res page 2 at all. You can use POKE 49235,0 to turn on mixed text and graphics on page 2 if only 40-columns are displayed on your screen, but then the text window has information from text page 2, which the operating system doesn't support. In other words, what you want to do can be done if you figure out a way to use text page 2 without overwriting your program, but I doubt you'll find it worth the trouble.*

4. A complete description of the memory switching on a 128K Apple would take an entire article; maybe I'll do one one of these months. However, the reason bank switching works in spite of the problems you mention is that the entire 64K is not switched in and out at the same time. It has independent sections. Memory can be set up so that part of the 64K address space is assigned to main

memory and another part to auxiliary memory. This is always an intermediate step between having a program running totally in one bank or the other. Programs that use both banks must also include instructions for turning the main bank back on before calling the operating system. It's pretty complicated, but it works.

Where's the 65816?

Your mention of the 65816 chip in the January issue (page 3) is a subject I keep reading about, but nothing seems to get done. I can understand why Apple itself is dragging its feet about offering anything like the 65816 to beef-up the existing Apple IIs, but I don't understand why some of the smaller companies who design and make the various add-on cards don't put out such a system.

Parker Mitton
Jamesburg, N.J.

Everyone has been so excited about these chips for so long we've lost sight of the fact that they aren't in full production yet. But they almost are. I suspect you'll see 65816 cards for the Apple II within 6 to 12 months. Brady Books is supposed to be publishing a programmer's reference manual, by David Eyes, to be available in July or so.

An abbreviated catalog

1. What do I poke into DOS 3.3 to eliminate all the information in a catalog except the filename?

2. How do you pronounce "Weishaar"?

3. How can I find other intermediate programmers who are interested in developing and writing programs for the education and rehabilitation of adults who suffer from visual, reading, and other language problems resulting from stroke or head trauma?

Rich Katz
Sunset Software
11750 Sunset Blvd., Suite 414
Los Angeles, Calif. 90049
215-476-0245

1. POKE 44507,16 : POKE 44508,54 deletes everything but the filenames from a DOS 3.3 catalog.

2. Back in the old country they pronounce it "vice-har" (rhymes with far). Around here we respond to just about anything. The name means "this person is far younger than he looks."

3. Maybe printing your address and phone number here will get you a few leads.

Egging us on

Whatever kind of paper you print it on, please keep up the editorials like the one in the January issue. Some of the dingdongs in the computer industry are going to do it in if someone doesn't get their attention.

Jim Freeburg
Port Orchard, Wash.

We ate bees for breakfast

How do you tell the Bs from the 8s in the source code listing for Basham's INCINERATOR, which appears in the March issue?

Cousin Weakeyes
aka Edwin J. Martin, Jr.
Kansas City, Mo.

My fellow newsletter publisher Bob Sander-Cederlof at **Apple Assembly Line** *called to POKE fun at me and ask the same question. Bob says he remembers seeing a listing in a magazine once where the Bs and 8s were perfectly distinguishable but were all reversed. Apparently the typesetter couldn't read the original manuscript.*

I thought we had all these problems solved at **Open-Apple** *because listings go direct from my computer to the typesetter's computer and are untouched by keyboard interfaces. However, it's now clear that what the world really needs is a decent typeface for use with computer programs.*

The face has to be what is called a "mono" font, which means all the characters are the same width. That is pretty rare in typesetting shops. Next, the face should include all the ASCII characters. I can't find any with that feature—my typesetter has programmed his machine to switch to a special symbols font for the odd characters that appear in programs. Third, the Bs and 8s should be distinguishable. Maybe I could use lower case letters for hexadecimal digits. Then at least it would be the b and the 6 that got mixed up, not the B and the 8.

If anyone out there knows some typeface designers, tell them to give me a call. I've got a thing or two I'd like to tell them.

/RAM and double high-res

Here's a tidbit your readers may find interesting. Both the ProDOS /RAM disk and double high resolution graphics use the auxiliary memory space from $2000 to $3FFF. Does this mean they can't be used together?

The answer is no. Data from the very first file stored in /RAM are placed at $2000. Thus, if the first file stored in /RAM is a hi-res picture, it will be stored in the double-high-res auxiliary page 1 space. Thus, the following command, if it's the first save to /RAM, will move whatever is in normal hi-res page 2 to double-high-res auxiliary page 1:

```
PRINT CHR$(4);"BSAVE /RAM/PIC.AUX,A$4000,L$2000"
```

The number of available blocks in /RAM will be reduced from 120 to 103, but /RAM will still be available. Subsequent saves to /RAM/PIC.AUX will update the double-high-res auxiliary page without destroying anything else in /RAM.

Robert C. Moore
1204 Marton St.
Laurel, MD 20707

Neat trick! Notice that this not only allows /RAM and double-high-res to work together, it actually provides a way you can draw the auxiliary half of a double-high-res picture in HGR2 using normal Applesoft commands, then move it to auxiliary page 1 without descending into the murkey world of assembly language.

For those of you who have never tried this before, let me give you a few hints. To use double-high-res graphics, you must have a IIc or a 128K IIe with a revision B motherboard (and unless your IIe was purchased right after the IIe was introduced in early 1983 and has never been upgraded, you'll have revision B).

To turn on double-high-res graphics, you must be in 80-column mode. Not just using the 80-column firmware, mind you, but with 80-columns displayed on your screen. Next POKE 49246,0 to turn on double-resolution. (POKE 49247,0 will turn it off.) After that HGR will display double-high-res. HGR will clear the high-res page in main memory, as always, so if you have something stored in /RAM/PIC.AUX it will appear to be in a jail cell.

AppleWorks, IIcs, and Okis

Robert Moore (the person who sent that last item) also sent me a three-page instruction sheet he's developed on how to get AppleWorks, a IIc, and an Okidata 92 serial printer working together. It's too long to print here, but it's quite good—he has written down the DIP switch settings and everything. He says he has also written up a double hi-res graphics sceen dump routine for the IIc/Oki 92 combination. For more information, I suggest you send him a stamped envelope addressed to yourself. His address follows that last letter.

ROM swap

Can you tell me if I can directly replace the Basic ROMs in my IIe with those of a IIc? I'm primarily interested in having the feature of being able to enter Basic commands in lower case, and not have to worry whether or not the caps lock key is locked down. Are there other advantages or disadvantages that might result from such a swap?

Paul Pagel
Enfield, Conn.

The enhanced IIe ROMs have the features you're looking for. The ROM in the IIc is a single chip, while in the IIe it's two chips, so they are not directly interchangable.

The Deacon's Macintosh

Despite Deacon Muskrat's talk about the Mac being a HeDaP machine, it looks like you used it on these order cards. Those fonts look like genuine Geneva and Cairo. Or was this done by that "terrible" program Fontrix?

Alan Millar
San Mateo, Calif.

The Deacon and I don't agree completely about the Macintosh. It is an excellent graphics machine and does "typesetting" good enough for order cards and other odd jobs. The user interface works well for people who don't use a computer everyday and for graphics oriented tasks. But holy snails, the machine is **slow**. *(Can that explain why decent documentation on how to program the beast is also taking so long to appear?)*

Incidentally, one of the best rumors I've heard lately is that the Macintosh operating system (the Finder*) is being rewritten so disks are ProDOS compatible. I doubt the rumor is true; I don't think Apple's 32-bit gang has enough humility to let it happen.*

Probing POKE 72,0

You have a good article on the system monitor in the February and March ***Open-Apples***. As an old Apple user, I humbly suggest, however, that you should have included a POKE 72,0 command prior to the CALL -151 (or CALL -155) just to be on the safe side of things.

C600G might deserve some attention also for a way to boot from the Monitor if all else fails.

Phil
76324,720

I can tell from your letter that you really are an old-timer. *Let's tell everybody a little bit about POKE 72,0. (It also appears, incidentally, in the S.H. Lam technique demonstrated in the February* ***Open-Apple****, page 13.)*

Byte 72 ($48) is where the contents of the 6502's status *register get stored when a machine language*

break instruction occurs. See page 20 of last month's letter for more detailed information on this. Every time a G(o) command is issued from within the Monitor, the contents of byte 72 are reloaded into the status register.

Among other things, the contents of the status register determine whether the microprocessor will do arithmetic in straight binary or in "binary-coded decimal". Binary-coded decimal is an interesting mode that makes it easier for assembly-language programmers to handle decimal numbers. However, the mode is rarely used; most machine language code I've seen—including all of the Monitor and DOS—was written to run in straight binary.

Thus, if the binary-coded decimal bit in the status register gets turned on by accident, a marvelous thing occurs. Your microprocessor starts giving the "wrong" answers to the most elementary calculations and your computer goes crazy.

Dying to try it? Here's how:

```
]CALL -151     (enter Monitor)

*48:8          (mess up byte 72 ($48) with an 8)

*3D0G          (return to Basic)
```

The results are pretty messy. To recover, press Control-Reset.

All versions of DOS use byte 72 as a temporary storage area. But all versions since DOS 3.2 reset this byte to zero after using it. Very early versions of DOS left it full of trash. With DOS 3 and DOS 3.1, it was necessary to poke a zero into byte 72 at some point before leaving the Monitor or your Apple could get a massive headache. With later versions of DOS, the poke usually isn't necessary. That's how I can tell you've been around Apples awhile.

Are you kidding?

I really am enjoying your newsletter and the two issues I have so far received have had lots of interesting info I can use. In fact, I was really elated when I read in the February issue the statement "as the owners of 40-track drives know, this upper limit (143,360 bytes) is all myth. DOS 3.3 will work, without modification, on disk drives having as many as 50 tracks and 32 sectors".

I was elated because quite a few database programs limit the number of records according to the storage capacity of the disk. I have one program (protected) that if the above statement were true, I could have more records and would not have to do any mods to the DOS. However, of two people that I talked to that had 40-track drives one said that there were 3 pokes that were necessary and the other wondered why he had got a whole disk full of patches to DOS for the 40 tracks. Did you literally mean what you said in your statement quoted above? I am anxious to hear. I am all set to go out and buy a 40-track drive so I can have more records in my database.

Dave Harvey
70140,162

Standard, absolutely unmodified DOS 3.3 can read, write, and store files on disks with up to 50 tracks. This is because DOS looks on each disk to find out how many tracks and sectors the disk has. This information is kept in the Volume Table of Contents, or VTOC (say vee-tock, located at track 17, sector 0), in bytes 52 and 53 ($34-$35).

Also in the VTOC is a map of the disk that shows which sectors are free and which are in use. This map allows disks to have up to 32 sectors. There is enough room in the map to squeeze in 50 tracks. Beyond that DOS 3.3 can't go without major modifications.

Before you run out and buy a 40-track drive, however, I recommend you test your database software on one of your friends' drives. Many programs assume DOS 3.3 disks will always have only 35 tracks. Such programs make internal calculations on that basis and refuse to use additional tracks even when they are available. I suspect this will be the case with most copy-protected database software, but I could be wrong.

To **initialize** *disks with other than 35 tracks you* **do** *have to make three pokes to DOS (for more on this, see the next letter). As for your friend with the disk full of patches, I don't know what those patches were for, but they aren't necessary to get DOS to use a 40-track drive.*

If your database program requires you to initialize data disks from within the program, you almost certainly won't be able to use the extra capacity 40-track drives offer.

In addition to drives with more than 35 tracks, I have also heard of drives that have more than 16 sectors on one side of one disk. While this is rare, DOS 3.3 can use them.

Typically, however, high-capacity drives don't use the RWTS section of DOS 3.3. This is the part of DOS that actually reads and writes magnetic patterns on disks. Most high-capacity drives bypass RWTS with a patch at $BD00 that causes all calls to RWTS to first jump to a program on the high-capacity drive's disk controller card. This program checks to see if the RWTS call is for the high-capacity drive or for an Apple floppy. If for the floppy, control returns to RWTS, which accesses the disk normally.

If the call is for the high-capacity drive, on the other hand, machine language programs in the disk controller ROM take the place of RWTS. These programs transfer data between the disk itself and the rest of DOS, just as RWTS itself normally does.

The actual, or "physical", number of tracks and sectors on these disks is irrelevant, because the disk controller programs can make it **appear** *there are however many DOS needs. Large DOS 3.3 volumes on the Sider hard drive, for example, appear to DOS to have 50 tracks of 32 sectors. I don't know how this data is actually stored on the Sider, but it could, for example, be in 16 tracks of 100 sectors. The point is that the Sider's control programs pass the data back to DOS as if it was arranged in 50 tracks of 32 sectors, and DOS,* **without further modification**, *uses all 409,600 bytes of that space.*

40-track patches explained

Check out the letter in the February 1985 *Nibble*, page 111 (upper right corner) and page 112. Forty tracks! Why has this been a secret for so long?

John Redfield
Fort Smith, Ark

[The letter, written by Yin H. Pun, says that most Apple-compatible drives manufactured after 1982 should have the ability to use 40 tracks. "...these include the original DISK II, Quentin, and Micro-Sci. Most slim line drives based on TEAC or Shugart mechanisms should also work. The only drives that do not work are ones based on a Shugart SA390 mechanism, including older DISK IIs." Pun's letter also gives the patches that must be made to DOS 3.3 to get it to initialize 40-track disks and the patches COPYA needs to duplicate them.]

Most Apple-compatible drives do, in fact, have the ability to use more than 35 tracks. I'm not yet convinced they can routinely use 40, however. The drive in my IIc, for example, bombs out after 39.

If you want to test the maximum capacity of your own drives, here's what you need to know. First of all, as mentioned earlier, standard-absolutely-unmodified DOS 3.3 can read, write, and store files on disks with up to 50 tracks. What an unmodified DOS **can't** *do is* **initialize** *disks with other than 35 tracks. But the modifications for this are pretty simple. Boot a DOS 3.3 disk and then make the following pokes:*

```
POKE 48894,T : REM T=the number of tracks you want.
POKE 44725,T*4
```

The first of these pokes modifies the RWTS disk-format routines so that magnetic patterns for "T" number of tracks are placed on the disk. The second patch modifies the routine that builds a new disk's free-space map so that additional tracks are marked as free.

It is also necessary to adjust the byte in the new disk's VTOC that tells DOS how many tracks are on the disk. DOS never does this automatically, but always uses the value found on the last disk accessed. This causes all kinds of confusion and consternation.

For example, if you catalog a ProDOS or Pascal or CP/M disk just before initializing a DOS 3.3 disk, the newly initialized disk will fail to work correctly. It is natural to catalog a disk before initializing it to make sure it's the erasable disk you think it is. But the sector that holds DOS 3.3's VTOC holds data on disks from these other operating systems. There's no telling what will be in bytes 52 and 53 (the ones that tell DOS how many tracks and sectors are on a 3.3 disk). Whatever junk is there, however, will go onto the new disk as its number of tracks and sectors. Consequently, it will fail to work correctly.

Likewise, if you catalog a 35-track disk just before trying to initialize a 40-track disk, the new disk's VTOC will **say** *it has 35 tracks, even though your modified DOS actually puts more tracks on it. Thus, just before trying to initialize a disk, you must either catalog or otherwise access a pre-existing disk with the correct number of tracks or do this:*

```
POKE 46063,T : REM T=the number of tracks you want.
```

To determine the maximum track capacity of your own disk drives, use the following program:

```
10 REM   *** Maxtrack ***

100 HOME : VTAB 12 : D$=CHR$(4)
110 IF PEEK(978)<>157 THEN PRINT "PROGRAM WORKS WITH
    48K DOS 3.3 ONLY." : END
120 INPUT "INSERT AN ERASABLE DISK AND HIT
    <RETURN>";A$

200 T=50 : PASS=0
210 PASS=PASS+1
220 POKE 48894,T
230 POKE 44725,T*4
240 POKE 46063,T
250 PRINT D$; "INIT FILE IN TRACK 18"

260 ONERR GOTO 300
270 FOR I=19 TO T-1
272 : PRINT "FIRST ";I;" TRACKS OK."
274 : PRINT D$; "SAVE FILE IN TRACK ";I
280 NEXT

300 IF PASS=1 THEN PRINT "LET ME TRY AGAIN." : T=I+1
    : GOTO 210
310 PRINT "THIS DRIVE CAN USE ";I;" TRACKS." : END
```

Run the program with an erasable disk in the drive. The program will attempt to initialize the disk with 50 tracks. After the disk has been initialized, it begins saving files (copies of itself) onto the newly-initialized disk. Because of the way DOS always does things, the first file saved will go in track 18 and succeeding files will go in succeeding tracks.

As the files are saved, the program reports how many tracks seem ok. Eventually an I/O ERROR will occur. The program will then reinitialize the disk with the apparent maximum number of tracks your drive will allow and test them. The program ends by reporting the maximum number of tracks your drive can handle. (The last file on the disk will be in the last track; your disk has one track more than this because of track 0.)

Owning a drive that can use more than 35 tracks isn't of itself a good reason to actually use more. You will find some compatibility problems. FID will handle these disks without complaint, but the ProDOS CONVERT program, which was apparently written in a cave somewhere without access to non-Apple drives, refuses to read DOS 3.3 disks that don't have **exactly** *35 tracks. A similar thing happens with the IIc's System Utilities.*

COPYA normally copies only the first 35 tracks of a disk. You can fix it by loading it, adding the following line to it, and resaving it.

```
95 T=40 : POKE 44725,T*4 : POKE 48894,T : POKE 770,T
    : POKE 863,T
```

Of course, if your drive uses more or less than 40 tracks, change "T=40" to reflect your situation.

Since an old horse like DOS 3.3 can easily adapt to drives with more than 35 tracks, you'd think ProDOS would be able to as well. You'd be wrong. Although bytes 41 and 42 ($29-$2A) of block 2 (track 0, sector 11) on ProDOS disks tell how many blocks the disk,..er volume, has, you can't just initialize a disk with extra tracks and increase this number. ProDOS is actually written to reject any read or write requests to floppy disk blocks beyond number 280 (the last one in the 35th track).

I was discussing this problem with Dennis Doms, a reader here in Kansas City, and he pointed out that **Beneath Apple ProDOS** *(pages 7-25 and 7-26, but don't look in the index cuz it isn't listed there) gives patches that make ProDOS version 1.0.1 work with 40 track drives. There's also a patch for version 1.0 of the FILER that makes it initialize 40-track disks. Dennis figured out the patches for the latest version of ProDOS (version 1.1.1) and the FILER (v 1.1) and here they are:*

```
REM for PRODOS version 1.1.1 only
UNLOCK PRODOS
BLOAD PRODOS,TSYS,A$2000
CALL -151
56E3:39
300G
BSAVE PRODOS,TSYS,A$2000
LOCK PRODOS

REM for FILER version 1.1 only
UNLOCK FILER
BLOAD FILE,TSYS,A$2000
CALL -151
426A:39
79F4:2B
300G
BSAVE FILER,TSYS,A$2000
LOCK FILER
```

According to my technical contacts at Apple, the "correct" way to go about getting ProDOS to work with 40-track drives is to have ProDOS-compatible ROMs on your disk controller card. ProDOS will then recognize the drive as being capable of using 40 tracks when it starts up and no patches will be necessary.

When it Rana, it Poura

Sorry, but you are wrong about the Rana Elite II version of DOS 3.3 (see *Rana, A.P.P.L.E., Abacus* in the March issue, page 22). They do not have different VTOC's for each side of the disk. Rather, they use the normal Apple DOS VTOC, which was apparently written to permit the use of 2-sided drives if Apple itself ever got around to it. Rana's two-sided drives can also access 40 tracks rather than 35, completely normally. Thus DOS 3.3 treats the Elite II as a 40 track, 32 sector drive. That is 320K per disk.

Rana's version of DOS is a very minor modification of DOS 3.3. It's compatible with normal 35 track, 16 sector disks. Users performing normal storage operations (even including direct calls through RWTS) should have no trouble doing so. Your append fix routine even works with it, proving that it differs only in trivial ways from Apple's version of DOS.

We have stayed with DOS rather than going to ProDOS for a lot of reasons: not only that we have painfully earned knowledge about what is going on, but more importantly that we need the memory space. By relocating DOS to the language card, we gain memory for Basic programs. Incidentally, Cornelius Bongers' **DOS MOVER** (see *Call -A.P.P.L.E.*, July/August 1981, page 9; and November/December 1981, page 81) works fine with Rana's version of DOS.

We have added a bit of code (the himem for Basic is at $BB00) that allows Applied Engineering's 1 megabyte *RAMWORKS* card to act like three (large) disk drives. We have made each drive look just like a Rana Elite II disk, namely 40 tracks by 32 sectors. This allows very simple copying of disk contents to and from the ramdisks. And there is still extra space on the card for the normal Apple AUX bank, which is left for 80-column text, double-res graphics, and anything else that needs to use it.

Our DOS also supports the Quark 10M hard disk, which is broken up into 32 volumes, each of which is also the same size (320K) as the Rana volumes. With a re-map table, we can assign any of the physical devices (ramdisk 1, 2, or 3, active hard disk volume, and floppy disk) to any slot/drive address. For instance, at start up the floppy is S6,D1, but later on accesses to S6,D1 may be sent to ramdisk 1 and those to S6,D2 to ramdisk 3. This just requires poking the correct numbers in a table in normal memory (the space between $BB00 and $BBFF), so it is very quick. Under program control it can be completely transparent to the user.

So what we end up with is a system that has on-line three 320K ramdisk volumes, with the ability to mount or dismount any of them to hard disk or to a floppy. We also use an *Accelerator* card (which is completely compatible with the *RAMWORKS* card), so it is a pretty powerful system.

We use all this stuff to put together Apple-based scientific instruments that we sell. Our instruments measure and process images (for example, from light or electron microscopes, or satellite images). We need a lot of space for the digitized images, as well as for the programs that operate on them, for the data files holding measurements of the features present in the images, and for statistical analysis. We use the extra slots (AUX has *RAMWORKS*, 3 has the *Accelerator*, and 6 has a controller for both the Quark and Rana drives, so there are still five slots left) for video interfaces, display interfaces, graphics tablets, printers, and so on.

We have used the Rana drives for several years, since they first appeared, and like them very much. It is true, however, that no one at Rana knows, or will admit to knowing, anything. Whoever wrote the original DOS modifications and Rana's stand-alone programs for disk formatting and copying departed long ago. Fortunately we were able to get the source code then and have since been able to maintain the software ourselves. The drives are great—fast and reliable, with a lot of storage. I can only blame lack of salesmanship for the failure to make them the mainstay of Apple users.

We also have one of the Rana 8086/2 boxes, with two Elite II compatible drives and a more-or-less (mostly less) IBM-PC compatible computer inside it. Nice idea. Terrible execution. The drives are half-high instead of the full height drives used in the normal Elite II, and are very unreliable for reading or writing in either the Apple or IBM format. (We have been told that the problem is a low read voltage, but in 12 months they have made no move to fix anything.) And this product again shows that Rana has no knowledgeable technical folks (or they lock them up very well). We have identified several fatal incompatibilities with IBM programs, such as the system crashing totally if any attempt to generate any sound (even a beep) occurs in a program, or if inverse characters are sent to the display. Incompatibilities are a fact of life, but crashes are unacceptable. The response from Rana has been no response at all, except that we can return the system if we want to. Curious attitude for a company, isn't it?

John Russ
Scientific MicroPrograms
213 Merwin Road
Raleigh, N.C. 27606

In addition to the Elite II, Rana also makes two other drives, the Elite I and the Elite III. The Elite I is a standard single-sided drive with 40 tracks. The Elite III is a double-sided monster with 80 tracks per side —652K bytes of storage. It must be the Elite III that uses the highly modified version of DOS 3.3 with two VTOCs (or did I dream that?). Obviously I didn't check out last month's remarks about Rana drives as thoroughly as I should have. I appreciate your setting me straight on this.