Open-Apple™

November 1986
Vol. 2, No. 10

ISSN 0885-4017
newstand price: $2.00
photocopy charge per page: $0.15

Releasing the power to everyone.

ProDOS bug found in Australia

Since January we've been tracing the source of a delicate little bug that likes to destroy ProDOS 5.25 inch floppy disks. AppleWorks and *Apple Writer* data disks seem to be the most commonly destroyed items. If you've experienced this bug, you know what misery is. It looks to me like the source of this bug has finally been found—in West Heidelberg, Australia. A vaccine has been developed and a dose has been included herein just for you.

The first report we had of this bug appeared in the January 1986 issue of ***Open-Apple***, page 103. Subscriber J. Ernest Cooper reported that 24 new Apples at his high school in Michigan were destroying an abnormally large number of data disks. On examining the disks, Cooper found that each was ruined the same way—track 0 was destroyed—and that each could be recovered by reformatting track 0 and rebuilding the directory.

By February (page 2.6), we had more reports of ProDOS track 0 crashes. This time the programs involved were AppleWorks and *PFS:Write*. In March (page 2.12) and in May (pages 2.29-30) we had follow-up letters from Cooper and another subscriber that indicated the problem might be related to weak power supplies.

Then, a little over a month ago, subscriber David Grigg in Research, Australia, sent me a photocopy of an article, which originally appeared in the AUSOM user group newsletter, by Stephen Thomas of Maclagan, Wright and Associates in West Heidelberg. The article identified some subtle differences in various versions of Apple's "floppy driver," the machine language module that actually turns on 5.25-inch floppy drives and reads and writes on disks. Thomas blamed the track 0 crashes on these changes.

One of the major products of Maclagan, Wright and Associates is the Digicard Classroom Network, which can connect as many as 28 Apple IIs to central disk drives and printers, as well as allowing each Apple II to use its own private disk drives. Soon after the release of AppleWorks 1.2, the most common inquiry on Maclagan Wright's support phones had to do with destroyed data disks.

Thomas observed that disks seemed to be destroyed only under ProDOS 1.1.1. Neither ProDOS 1.0.1 nor DOS 3.3 were causing any problems. So Thomas began investigating the differences between the three.

Like all devices connected to an Apple II, floppy drives are controlled by a machine language module that reads and writes to 16 special bytes associated with the slot the disk controller is in. These bytes lie in the address range from $C080 to $C0FF. The 16 bytes from $C080 to $C08F are associated with "slot 0" (which holds built-in "language card memory" on current Apples), the 16 bytes from $C090 to $C09F are associated with slot 1, and so on. By putting the value "s0" in the X register (where "s" is the slot number), an assembly language programmer can easily access the correct byte for any slot with indexed addressing. This means using commands such as *LDA $C080,X*, which determines which byte to tickle by adding the contents of the X register to $C080.

For Disk II-type floppy drives, the first eight of the 16 input/output bytes control the stepper motor that moves the disk arm across the disk. The next two turn the drive motor, which actually spins the disk, off and on. The next two are for selecting drive 1 or drive 2. The next, $C08C,X, is the byte from which disk data can be loaded when reading a disk. When writing to a disk, tickling $C08C,X tells the drive to write *right now*. Byte $C08D,X is where you store a byte you want to write on a disk, but it doesn't actually get written until you hit $C08C,X. Byte $C08E,X puts the disk in read mode; $C08F,X puts the disk in write mode.

The first difference Thomas noted between DOS 3.3 and ProDOS is that DOS 3.3 *always* hits byte $C08E,X to put the drive in read mode before touching any of the other soft switches. ProDOS, on the other hand, doesn't take this safety step. ProDOS assumes the drive has been left in read mode by the previous access. This turns out to be the ProDOS floppy driver's Achilles' heel. In terms of the software, read mode is a valid assumption. In terms of the hardware, however, maybe it's not, and when it's not, disks are destroyed.

Determining just how the softswitch at $C08F,X gets hit to accidentally turn on write mode involves some black magic. Thomas noted that one of the differences between ProDOS 1.0.1 and 1.1.1 was a section of code, added right to beginning of the driver, that looks like this:

```
D6C3- 9D 80 C0  STA $C080,X
D6C6- 9D 82 C0  STA $C082,X
D6C9- 9D 84 C0  STA $C084,X
D6CC- 9D 86 C0  STA $C086,X
```

This code makes no sense in terms of the stepper motor, which these softswitches control. What's going on here is that the ProDOS floppy driver, as a safety step, is disabling any 3.5 inch drives that might be connected in a daisy chain with the floppy that is about to be accessed. When 3.5 and 5.25 drives are connected in a daisy chain, they both must be accessed through the same 16 softswitches.

In attempting to figure out what this code was for, Thomas noticed that these four commands cause an electron war on the Apple's "data bus". The data bus is the set of "wires" that is used to transfer data between memory locations and the microprocessor.

The four STA instructions all cause the microprocessor to place the contents of the A register on the data bus. Meanwhile, the four accesses to even bytes in the disk controller's 16-byte range all cause it to "drive" the data bus as well. For those of you who are interested in such stuff, I'll let Thomas explain why.

"Referring to the circuit diagram on page 145 of the DOS 3.3 manual, you will see that the two output enable pins of the 74LS323 (G1 and G2) are

"MY GOSH BARBARA, IF YOU DON'T THINK IT'S WORTH GOING A COUPLE OF WEEKS WITHOUT DINNER SO WE CAN AFFORD A 'IIgs', JUST SAY SO."

connected to the DEV* and AD0 lines of the Apple bus. The effect of these connections is that any reference to an even location in the disk controller's 16-byte I/O space will cause the '323 to drive the data bus. If the reference in question happens to be a write, the output of the '323 will "fight" with the bus buffers on the main board, which are also enabled during a write cycle.

"The exact mechanism by which a data bus fight gives the address-bus-driven controller hiccups is fairly obscure, but is probably something to do with the high-current data bus and supply rail glitches (caused by the fight) being inductively coupled onto nearby address-bus tracks. The Apple address bus is very noisy at the best of times and it would not be hard for an inductively coupled spike to be the straw that breaks the latch's back.

"As for power supplies....it's conceivable that an under-voltage power supply would lower the noise immunity of a system, making it more susceptible to noise inductively coupled onto its address bus. In any case, when you get rid of the bus fights you get rid of the problem. The patches we've developed have completely cured every ProDOS-wipes-tracks problem we've tried it on."

Thomas's patch has two parts. The first consists of changing the store (STA) instructions shown above to read instructions (LDA), so that the microprocessor drops out of the data bus fight. In terms of disabling the 3.5 drive, both commands work; in fact, the UniDisk firmware itself always uses reads. The second part of Thomas's patch consists of adding a LDA $C08E,X command near the beginning of the ProDOS 1.1.1 floppy driver to make sure the drive is always in read mode before its motor is turned on.

Here's a small program that makes the patches Thomas suggests. It works, and seems to be needed, only with ProDOS 1.1.1. May it save us from the anguish of crashed disks many times over.

```
10 REM Stop ProDOS track 0 trashing
20 REM patches by Stephen Thomas
30 REM         MacLagan Wright & Associates
40 REM         West Heidelberg, VIC  Australia

100 TEXT : HOME : D$=CHR$(4) : E=0
110 VTAB 12 : PRINT "Now patching ProDOS."
120 ONERR GOTO 500
130 IF PEEK(116) < 96 THEN E=1 : GOTO 500 : REM make sure there's room
140 PRINT D$;"UNLOCK PRODOS"
150 PRINT D$;"BLOAD PRODOS,A$2000,TSYS"

200 FOR ADR=22211 TO 22220 STEP 3 : REM change four STAs to LDAs
210 : IF PEEK(ADR) = 189 THEN E=2 : GOTO 500
220 : IF PEEK(ADR) < > 157 THEN E=3 : GOTO 500
230 : POKE ADR,189
240 NEXT

300 ADR=20484 : V(0)=189 : V(1)=142 : V(2)=192 : REM chg NOPs to LDA $C08E,X
310 FOR I=0 TO 2
320 : IF PEEK(ADR+I) < > 234 THEN E=3 : GOTO 500
330 : POKE ADR+I,V(I)
340 NEXT

400 PRINT D$;"BSAVE PRODOS,A$2000,TSYS"
410 PRINT : PRINT "Patches completed."
420 END

500 PRINT CHR$(7);"Error! No patches were made."
510 ON E GOTO 530,540,550
520 PRINT "PRODOS file not found." : END
530 PRINT "Not enough room to load PRODOS." : END
540 PRINT "File has already been patched." : END
550 PRINT "This version of ProDOS not 1.1.1."
```

Maclagan Wright's (123 McEwan Rd, West Heidelberg, VIC 3081 03-458-1211) Digicard Classroom Network consists of a master controller that houses two floppy disk drives and two printer ports (one parallel, one serial) and that connects up to 28 Apple IIs via a flat ribbon cable. The system supports both simultaneous downloading of software to all networked computers at once or sequential access for all users to the two networked disk drives. All users have sequential access to the printers and all have the ability to privately use drives directly connected to their own computer.

In the U.S., Maclagan Wright's stuff is distributed by Major Educational Resources Corp, 10400 Ridgland Rd, Suite 5, Hunt Valley, MD 21030 301-628-1527. The Digicard network is priced at $2,495 for the master controller and $195 per station for connection cards, hardware, and cable.

Maclagan Wright has also developed what it calls the Digicard Shared Resource Controller. This device allows a group of networked computers to share a 20 to 70 megabyte hard disk, three printers, and a clock/calendar. It can be used with the Digicard network, or it can be directly connected to as many as 32 mixed Apple II, Macintosh, and IBM PC computers using twisted pair cable.

Apple is having trouble getting the new IIgs into production. Apparently at least one of the new chips in the machine is giving them the fits. Dealers have demonstration units that will knock your socks off (the sound is so good you'll assume it's not coming from the computer; graphics taken with a video camera are television-quality), but it doesn't look like they'll have any units to sell until November or later. Availability of the IIgs upgrade kit for the IIe has been changed from "January 1987" to "first-half 1987."

A computerized index of 1985 issues of *Open-Apple* is available at an introductory price of just $5 from Terry Tufts at T-Squared Enterprises, 1015 S Ridge Ave, Arlington Heights, IL 60005 (add $3 for overseas shipping). This is a new member of Tuft's *Periodical Index* series. Also available is an index covering the 1981 through 1983 issues of ***Call -A.P.P.L.E.*** ($14.95).

The *Periodical Index* is an author/title index, but it's also much more. Each article and letter-to-the-editor entry includes a one- or two-line description that can be searched for key words. Each entry is also categorized by major (business, education, games, home/hobby, programming, hardware, misc), minor (communications, data base, financial, math, medical, management, music/sound, spreadsheet, word processing, misc), and additional (announcement, application, editorial, opinion, review, tutorial, misc) keys.

I'd like to see Tufts sell subscriptions to new, all-inclusive updates to this index that would come out quarterly. He's interested, and thinks he should charge $17.95, but isn't sure enough of you are interested to make the project worthwhile. You'll have to convince him.

***The Main Menu* is a new monthly AppleWorks newsletter** from the folks who publish *inCider* ($49.97/yr, P.O. Box 802, Peterborough, NH 03458). The first issue was dated October 1986. It includes a review of the new AppleWorks 2.0, the news that Rupert Lissner has changed his name to Robert, several pages of information on fixing AppleWorks printer problems, an introduction to the use of text files and pathnames with AppleWorks, an Applesoft program that will turn "comma-delimited ASCII files" into text files that can be loaded into the AppleWorks database, and a survey of the AppleWorks-related programs (desk accessories, spelling checkers, mail merge, macros, work-alikes) that are currently available.

This issue also says Lissner's favorite AppleWorks command is open-apple-ditto. "He uses it in spreadsheets to copy the entry above the cursor into the current cell; existing entries are replaced by the one above." Wow, I didn't know that command worked in the spreadsheet. I immediately tried it and found out Lissner must have his own personally-patched version of AppleWorks; it certainly doesn't work in mine. Open-apple-ditto is one of my favorite commands, too, but it only works in the multiple-record layout of the data base. It will copy the contents a field into the record below it. Put the cursor in the cell you want to change and press open-apple and the apostrophe (even though Apple's manuals call this open-apple-', *do not* press the shift key). See ***Open-Apple***, December 1985, last two paragraphs of page 92, for more on this command.

Those of you interested in Apple II clones, such as the Lazer, the Basis, and the Franklin, should join A.S.C.I.I. ($20/yr, 6B2 East Wind Rd, Tecumseh, MO 65760 417-679-3526). Doug Trueman, editor of A.S.C.I.I.'s newsletter, *The Clone Ace*, is one of the few true clone experts around. The newsletter appears somewhat sporadically (the August issue was mailed September 24), but includes more news and technical information about clones than you're likely to find anywhere else.

Penguin Software has changed its name to Polarware after objections from a book company named after an Antarctic bird. Penguin, I mean Polarware, was an early leader in removing copy-protection from software and in lowering software prices. Polarware specializes in graphics programs and in games, but also sells a data manager, a terminal program, a disk of ampersand utilities, and other good stuff. Among the best is the stock of books remaining when *Softalk* went out of business. The price of these has been reduced to just $2.95—too cheap, I think—for classics like *Assembly Lines*, by Roger Wagner (an introduction to 6502 assembly language on the Apple); *Applesoft Isn't Hard*, by Doug Carlston; and *Graphically Speaking*, by Mark Pelczarski (where have I heard of him before?). Write or call in an order and tell them to send you their summer newsletter and price list, too—P.O. Box 311, Geneva, IL 60134 312-232-1984.

If you'd like to hear the difference between *digitized* speech and *synthesized* speech, call this number: 800-843-9363. The phone will be answered by a $795 device called the Computerfone II, which is attached by means of an RS-232 serial cable to an Apple IIe. It not only answers the phone and talks to you in your choice of digital or synthetic voices, it also accepts touch-tone input, can digitally record and play back your voice, and can make calls as well as accept them.

Digital speech sounds as good as a tape recording. The problem with it is that it must be pre-recorded. Synthesized speech sounds like a robot talking, or worse. Its advantage is that it can convert text you already have—any word processor file, for example—into something that can be understood over the phone. You can download either kind of speech from your Apple to the Computerfone II while the device is in use. Combine this with the device's ability to make digital recordings, which can be saved on the Apple's disks, and you have what's called a "voice store and forward" system.

The Computerfone II can be used to make every touch-tone phone in the world a terminal for your computer. And it allows your computer to call any phone in the world, touch-tone or not, and leave a message ("A ground-floor window was just broken on the north-east corner of the home at 123 Main St. Please send help." or "Because of the snow storm, tonight's user group meeting has been rescheduled for next week." or "This is an electronic purchase order. Please send the following items to:..." or "This is your computer. The staff meeting starts in two minutes. If you are going I'll take your calls. Press 1 for going, 0 for not going.")

On incoming calls from remote touch-tone phones, users could request computerized information such as prices, inventory levels, or shipment status; or could give instructions, such as bank bill-payment, product orders, or service dispatch requests; or could leave messages. The system can also notify an operator that a caller wants to talk to a real person. We'll all be at the beach soon, however, because the computers will be doing all the work. By the way, the computer that answers 800-843-9363 belongs to Suncoast Systems, P.O. Box 7105, Pensacola, FL 32514, makers of the Computerfone II.

Ask (or tell) Uncle DOS

Default of AppleWorks

I have been unable to determine how to change the default option settings for the AppleWorks word processor, aside from loading an otherwise empty file that contains the margin and character-per-inch settings that I prefer and changing its name. Am I missing something or has Apple really provided no utility function to allow this? Do you know of any patch I could apply to the AppleWorks program files to change the defaults?

Douglas J. Sietsema
Culver City, Calif.

*I passed your question on to **AutoWorks** author Alan Bird. Alan says that the default characteristics assigned to a new AppleWorks word processor file are embedded in a section of assembly language code rather than being grabbed out of an easily-modifiable table. Alan figured out how to change a few of the defaults, but they only work with AppleWorks version 1.3. He says to proceed like this from the Applesoft prompt:*

```
BLOAD SEG.M1,T$00,A$2000,L$33,B$4E64

POKE 8205,platen width in tenths of an inch
          (80=8 inches)
POKE 8217,left margin in tenths of an inch
          (must be 10--1 inch--or more)
POKE 8227,85=unjustified; 74=justified
          67=centered
POKE 8231,characters per inch AND right margin

BSAVE SEG.M1,T$00,A$2000,L$33,B$4E64
```

*Alan pointed out that a much simpler solution is to use **AutoWorks** or another AppleWorks-macro package to create a macro that sets up the default parameters you want.*

Alan also sent along a small program that patches AppleWorks so that it doesn't stop on the way to the desktop during the startup sequence. These patches will be welcomed by those who have a clock card or who boot AppleWorks off a 3.5 drive or a hard disk. Here's Alan's program:

```
10 FI$ = "APLWORKS.SYSTEM" : REM use full pathname
20 PRINT CHR$(4);"BLOAD";FI$;",TSYS,A$2000"
30 A = PEEK(8250)
40 V = (A=202) + 2*(A=169) + 3*(A=180) : REM version
50 IF NOT V THEN PRINT
      "Unrecognized version of AppleWorks." : END
60 PRINT "Patching AppleWorks V 1.";V
70 IF V=1 THEN A1=13139 : A2=13522 : REM V1.1
80 IF V=2 THEN A1=13135 : A2=13518 : REM V1.2
90 IF V=3 THEN A1=13193 : A2=13855 : REM V1.3
100 POKE A1,44 : REM no space bar for AppleWorks
110 POKE A2,208 : POKE A2+1,19 : REM no Return
      for date
120 PRINT CHR$(4);"BSAVE";FI$;",TSYS,A$2000"
```

Make sure you have a backup of APLWORKS.SYSTEM before running this program.

A completely bold suggestion

I have come up with a superior solution to the problem of using the ImageWriter's boldface mode throughout an AppleWorks document. The idea of defining a custom printer with the BOLD END code set to nothing, which you described in your September 1985 issue (page 72), prevents you from using the ImageWriter's proportional fonts. AppleWorks doesn't support proportional fonts on custom printers.

Instead, I used a disk sector editor to search the AppleWorks disk for the ImageWriter's $1B $22 end boldface code. (I used *DiskWorks*, a freeware program from Living Legends, by the way. It's fast, easy to use, and only costs $10. I highly recommend it.) I was amazed to find this code appears only once anywhere on either side of the AppleWorks disk—it's in the file called SEG.PR. I simply changed the two bytes to null codes ($00 $00). Now when I ask for boldface at the start of a document I get it throughout the document. Voila!, correspondence quality.

The only disadvantage of this technique is that it doesn't allow you to embed boldface characters into a document any longer, which wasn't much of a loss for me. To do a standard-quality document, simply turn the printer off and back on. This resets the ImageWriter to its default settings, which include normal print. I find this a lot faster than your trick of printing an empty spreadsheet with a special printer setup string (ESC ") to turn off boldface mode.

I also have a few questions. Is there any way to get the old character set on a IIc? I am thinking of some type of hardware change similar to what Don Lancaster suggests for the IIe in his book "Understanding the Apple IIe." I have a copy of *TK! Solver* that would be a lot friendlier with the old character set. The company that currently owns this program doesn't care about supporting the Apple version, based on a couple of recent calls on this subject.

Do you know of any cheap sources for the 256K x 8 RAM chip required to get the last bank of memory on an Applied Engineering ZRAM II?

Do you have any plans to get a BIX ID for ***Open-Apple***? I find the questions being asked and answered in their Apple II forum are a generally more interesting than the run of the mill MAUG queries. The help is fast and accurate. I had a problem running *Triple Dump* on the IIc and a BIX Apple moderator went to the program's author for a fix. Something about unclaimed interrupts in the IIc. I fixed it myself by changing ProDOS so it wouldn't crash when an unclaimed interrupt occurred, using a patch I got from MAUG. I later received a letter from the author with two new *Triple Dump* disks patched to eliminate the problem. I'm impressed with both BIX and Beagle Bros.

Dean Shutt
75126,2240

*I don't know of any hardware modifications that would prevent the IIc from displaying MouseText characters when in 80-column mode, but our readers might. It may be easier to come up with a software patch for **TK! Solver** that would make it use the standard Apple character set, rather than the MouseText-using "alternate character set." You just have to find a way to tickle byte 49166 ($C00E), after **TK! Solver** turns on 80 columns, to switch to the standard character set. There's a disadvantage, too, however—when you do this what was supposed to*

show up on the screen as inverse will change to flashing. Lower case inverse changes to flashing punctuation marks. Thus, even with this trick, your choice is between having inverse upper case or inverse lower case readable. The alternate case will either be MouseText (if you go for readable lower case) or punctuation marks (if you go for readable, but flashing, capitals).

We haven't yet seen a source for the 256K x 8 chip you mentioned.

***Open-Apple** isn't up on BIX yet (we're only barely up on the Source and CompuServe), but you make it sound enticing. I don't have accounts for Genie, Delphi, BRS After Dark, AppleLink, PMS (KC Apple user group's board), or MCI Mail yet, either. I do have a Dow Jones account, but haven't used it for over a year. I talked Viewdata out of a free account, but it closed down before I had time to log on. Bill Basham has been trying to get me to look at his **Diversi-Dial** program and Gary Little saw that I got a copy of Pinpoint's new **Point-to-Point**, which he wrote. My wife wants me to find a way to transfer files between Apples and the 14-ounce Sharp 1500A handheld computers her research group uses to collect data.*

*But before I do any of that I probably ought to read Alfred Glossbrenner's **Complete Handbook of Personal Computer Communications: Everything You Need to Go Online with the World**, which looks to me like an incredibly useful book. Before I do that, though, I probably should write a program that will clone me on any Apple II. That way I could get lots more stuff done and could be having fun on many different parallel planes of the same universe at the same time. Meanwhile, anybody want to guess which item on the above "communications to-do list" will get done first?*

Wait a second, here

How do you get the computer to keep time (say, 5 seconds) while waiting for an input to stop it, as in a game?

Herman Kyrklund
Big Springs, Neb.

Try this subroutine:

```
100 REM *** GET with timer ***
110 A$="" : T=D
120 FOR I=0 TO D : REM D controls delay time
130 IF PEEK(-16384) < 128 THEN 160
140 A$=CHR$(PEEK(-16384)) : POKE 49168,0
150 T=I : I=D : REM T indicates response time
160 NEXT
170 RETURN
```

Before calling this routine you must give D a value. This will control how long the routine delays. The larger the value you give, the longer the routine will wait for a key press. Setting D to 360 gets you a delay of five seconds on a standard Apple. On a IIgs or other accelerated Apple, of course, a five-second delay would require a bigger D—1200 for Applied Engineering's TransWarp, for example.

When the subroutine returns to its caller, the variable T will hold a number that indicates how long it took the user to respond. The larger the number, the longer it took. If T is equal to D when the subroutine returns (or if LEN(A$)=0), then no key was pressed. If T is less than D (or if LEN(A$)=1), then a key was pressed and A$ holds its value.

*The magical peeks and pokes in this routine are well known by many **Open-Apple** readers, but for those of you who are new around here, I should explain that byte 49152 (commonly known as KBD) is a hardware softswitch that holds the ASCII code of the last key that was pressed. When a new key is pressed, the value in this byte rises above 127. After you have collected the keypress, you poke byte 49168 (known as KBDSTRB or the keyboard strobe), which is another hardware softswitch. This tells KBD to drop back below 128 so that you will be able to sense the next keypress.*

As this subroutine is written, it will respond to a key press that occurred before the subroutine was called (and T will come back equal to 0). If you want to prevent this, add a POKE 49168,0 to line 110.

I found this little subroutine tester to be fun (see how fast you can press the key, etc.):

```
10 D=360 : GOSUB 100
20 PRINT T, A$
30 GOTO 10
```

More disk recovery

In response to "disk resurrection" in your September issue (page 2.59)—I do data disk recovery for a local Apple dealer. Most of the problems have been with AppleWorks data disks. I've recovered files ranging from a 46K word processor file to a 34K spreadsheet to a 53K database. The problems I've encountered range from the directory needing some fixing up to the more difficult task of rebuilding the file.

The last two disks I worked on held about 250 blocks of database records. AppleWorks could not identify the volume. Most of the records were still on the disks but they had been written over and some of the individual record's pointers were left off, missing, or messed up. So after rebuilding the volume directory, using new filenames, and the index blocks of the new files, I still had many fields within the individual records that needed to be redone.

Without Apple's documentation on AppleWorks file structure a person would have a very difficult time rebuilding these files. And the *ProByter* program from Beagle Bros is invaluable. They are very helpful if you call with any questions. In your article you mention *Bag of Tricks 2*. Even if you rebuild the volume directory with this program you may still have a long way to go before getting useful data back out of the disks.

I do provide a disk recovery service. If anyone has any questions please have them contact me.

Pete Johnson
P. O. Box 5
New London, MN 56273

I have taught classes in disk recovery to school computer coordinators in southern Minnesota, and for the past several years have done recovery for various schools and individuals. I do not advertise. My observation is that trying to do such work as a business would be difficult due to the wide variety of disk problems. How does one guarantee success when too often someone has put a fingerprint in the window and data and disk are gone? Some problems can be remedied in minutes; some problems, especially in ProDOS disks, can take prohibitively long. I enjoy the process, but have not figured a way to do it in such a way to give consistent satisfaction to myself and my customers.

I have a question I can't find an answer to. Within a program I can do a POKE 1012,0 to force control-reset to reboot. However, if control-reset is hit before that line is executed, everything stops. Is there a DOS location that can be altered so that control-reset will force a reboot even as DOS is booting?

Jim Aufderheide
New Ulm, Minn.

Losing data is so disheartening that its always worthwhile to try to recover a disk, but as you point out, the results can never be guaranteed. When I've destroyed data in the past, it's usually been by accidentally initializing the wrong disk. This happens when the lights go out inside your head—when you realize what you've done you feel like leaving them out and going to bed to cry for 40 days and 40 nights.

*Under DOS 3.3, put a zero at $9E37 and initialize a new disk. The new disk's DOS will reboot whenever you press reset. Under ProDOS, the proper locations to change keep moving around because of all the versions, but essentially what you do is search for the sequence $49 $A5 and change $A5 to $00. You'll find the $49 $A5 sequence in both PRODOS and BASIC.SYSTEM and you'll have to change both. If you've purchased Glen Bredon's **ProSel**, which I seem to recommend monthly, you'll find a program on there called BLOCK.WARDEN that will search for a sequence of bytes within a file and allow you to change them. Many other programs also do this, or you could use the Monitor's search command if you have an enhanced IIe (see March 1985, page 20, for more).*

The last 48K

Some time ago I purchased an accelerator for use in my Apple II-Plus. I am generally quite pleased with its performance. However, there is one small factor that is bugging me about this device. According to the user's manual that came with it, the plug in card, which is installed in slot 0, "...contains 80K of fast (150 ns) RAM. This memory serves 3 functions: (1) 48K of the accelerator's RAM replaces the Apple's main (lower) RAM, (2) 16K acts as a built in language card, and (3) the remaining 16K serves to store ROM language in fast RAM. Memory on the Apple's main board is not used..."

It is really bugging me that I now have 48K of (original) Apple RAM "lying fallow" in my machine. Is there some software and/or hardware way of accessing this memory, even if it is only to use it as a RAMdisk?

E. J. Martin
Kansas City, Mo.

Just to check our memory (so to speak), Dennis booted up with the Applied Engineering TransWarp; disabled it, then examined the Apple's memory to see if it really was "unused". It wasn't; a copy of the program last loaded was in the on-board memory. Neither Dennis or I really understand how these boards work, but it appears to us that what you want to do is impossible. (And there is no better way that we know of to find somebody who can do it than to make that statement.)

Nevertheless, let me get philosophical on you for a moment—for thousands of years the factor that has limited the potential of individual humans has been the scarcity of finished material goods—food, tools, and things like that. Nowadays the factor that limits us is time. Your question comes from the "scarcity of materials" mindset. Our answer comes from the "scarcity of time" mindset—it would be far more efficient to buy another memory card for your computer, if you really need it, than to take the time to figure out how to squeeze the last drop of blood out of the RAM you already have.

Cheap hard drive alternatives

Hard drives of 20 to 30 megabytes are available for $300 for IBM-type machines, without controllers. Is

there any way to get/make a controller which will interface my Apple IIe to one of these hard disks?

Lance Eggleston
Hamburg, N.Y.

We haven't heard of anything along these lines. Consider, however, that the $300 you are quoting is the rock bottom price for a bare hard drive. Besides missing a controller, you'll also be missing a power supply, a cabinet, and utility software.

First Class Peripherals (3579 Highway 50 E, Carson City, NV 89701 800-538-1307), on the other hand, is running a "second anniversary special" of $500 for their 10 megabyte Sider between now and the end of the year. It includes all that stuff, a manual, and a technical support number. Unless you want to start with an IBM-type drive just to prove it can be done, I can't see the point of trying. The extra $200 seems quite reasonable for a controller, power supply, cabinet, software, manual, and support.

Unless you really have tons of data that must be stored all in one place, such as the public library's card catalog, or the Power and Light's customer billing list, or Heinz 57's inventory, consider very carefully whether you really need a hard drive at all.

Applied Engineering's RAMfactor card or Cirtech's Flipster will give you a bootable 1 megabyte RAM-based drive that will work with any Apple II except the IIc, that can be partitioned into various operating systems, that is faster than a hard drive, and that is ***silent.*** *(IIc users can get Apple's new IIc memory card, which does all this stuff except the partitioning.) Of course, if the power goes out, you lose the data on the card, but you can solve that with the uninterruptable power supply (see January 86, page 98) you were eventually going to have to buy for your hard drive anyhow, and just start leaving your computer on all the time (turn the monitor off, though). A cheaper alternative would be AE's battery backup option for its card.*

Put your most-often-used programs on the RAMdisk, and use 800K drives for data storage. I think that for the typical Apple II user this is a faster, quieter, and more reliable system than one that includes a hard drive. Hard drives certainly have their uses, however, if you need to get more than 800K in one file or if your program is constantly accessing your data disk.

FILE NOT FOUND tricks

On page 16 of *The DOSTalk Scrapbook* you recommend that the DOS command VERIFY be used in Applesoft programs to confirm that a file exists before trying to OPEN and READ it. Otherwise, if the file doesn't exist, the OPEN command will create a bogus file that takes up a sector of the disk and adds a line to the catalog.

This is OK, but if the file does exist, VERIFY will proceed to check the integrity of each sector in the file. This can take a noticeable amount of time unless the file is quite small.

Instead of VERIFY, I use UNLOCK as the first reference to a file. If the file isn't there, UNLOCK produces an error message without creating a new file (just like VERIFY); but it doesn't waste a lot of time if the file is there. If you have a read-only data file, use LOCK.

Paul Nix
Summit, N.J.

You raise a good point; VERIFYing a 250-sector text file can take awhile under DOS 3.3. Under Basic.system, on the other hand, VERIFY just checks the directory for the name, but doesn't scan the file, and takes almost no time at all.

Under DOS 3.3, another possiblity is to just begin READing the file. You can skip the OPEN command completely. If the file doesn't already exist, you'll get a FILE NOT FOUND error, not a new bogus file like OPEN will give you. For more on this, see page 219 of **The DOStalk Scrapbook** *or the March 1985* ***Open-Apple****, page 23.*

Dennis came up with a completely new and different suggestion. He says to try the command:

```
1000 PRINT D$;"RENAME TESTFILE,TESTFILE"
```

This works under either DOS 3.3 or Basic.system and doesn't change the status of the file. Using an ONERR GOTO trap, you can find out several things about the status of the file, at least under DOS 3.3:

```
---DOS 3.3---
FILE NOT FOUND      file doesn't exist
FILE LOCKED         file locked
WRITE PROTECTED     file unlocked, disk locked
no error            file unlocked, disk unlocked

---Basic.system---
PATH NOT FOUND      file doesn't exist
no error            file exists
```

IIgs, *Apple Writer, General Manager*

The write-ups of the IIgs that I have been able to get my hands on fail to answer the most important consideration from my point of view. I don't care a fig for graphics or sound. I use Apple IIs to provide accounting, inventory management, and database management for a small non-profit historic restoration and its gift shop. We use a IIe in the office and II-Plus in the shop, transferring data on a floppy disk. Since I was unable to find appropriate software, I have had to write it myself as a self-taught Applesoft programmer. The organization's growth will soon require a major rewrite of my programs that I had hoped the IIgs, with its larger memory, might avoid.

However, it seems that compatibility with existing programs is achieved by simply including an Apple IIe in the innards of the IIgs. My question is this: can I run my existing Applesoft programs without major modification (perhaps using upgraded ProDOS and Basic.system files), *and use the increased memory for variables?*

Will the IIgs run *Apple Writer II*? If so, do you get to use the additional memory in the editor? I frequently use *Apple Writer* to edit my programs and data files (even my random access files, which I originally opened as sequential files consisting entirely of spaces and thus can load and save in *Apple Writer* with no problem).

A highly useful discussion, for me anyway, would describe the merits of the IIgs versus a IIe with an accelerator board and memory cards, from the point of view of users who program in Applesoft to handle large amounts of data. I'm already using Beagle Bros *Extra K*, and it isn't enough. If the IIgs will run my Applesoft programs using its 256K or more of memory, I would find it very attractive. If not, then I will probably have to extensively revise my programs to throw data to and from the disk.

The query regarding mailing labels (September 1986, page 2.59) prompts me to share my own experiences. For some time I have been printing zip-code ordered "Cheshire" labels (44 addresses per page in a four-across, 11-down format) for our organization on ordinary 15" computer paper. Our local letter shop will slice these up and paste them on envelopes for about $6 per thousand. The savings in time and postage are considerable.

I spent some time determining how to do this. I use *General Manager* (now dropped by Sierra On-Line and without a source of user support as far as I know). This program can sort records on any field, from a data base that is much larger than will fit into memory at one time, and it allows the user to write Applesoft programs that can use the data base records.

Using *General Manager*, I was able to write a program that not only prints four-up labels, but also neatly adjusts each label's format for missing fields, extra long fields, common typos, and so on. I have never understood the inability of most data base programs to print more than one-up labels, or the usual requirement of loading the entire database into memory to perform sorting operations. The apparent failure of *General Manager* and the enduring popularity of *PFS File* and similar programs is a continuing source of amazement to me.

But I am uncomfortable running a program for which no outside support is available. Do you know of any? Is there another, preferably ProDOS-based, program that will do what it does, and that can read data from a text file so that, if and when the time comes to switch, our records would not have to be re-entered by hand?

Lawrence S. Pratt
Great Barrington, Mass

Your suppositions about the IIgs are correct. The additional memory the IIgs hardware supports cannot be accessed by Applesoft except as a RAMdisk, at least at the moment. Which is not to say that wizards like Alan Bird and Mark Simonsen, authors of Beagle Bros' ***Extra K****, couldn't write a* ***More Extra K*** *or somesuch that would take advantage of additional memory. If I was writing such a program I would make it so it could use any RAMdisk, however, not just the IIgs.*

Bird has already written a program called ***ProBasic*** *(see September 1986, page 2.62) that is Applesoft compatible and that allows numeric arrays to reside in disk files. This is an easy way to use vast amounts of RAMdisk RAM from Applesoft. The April 1986* ***Open-Apple*** *also addressed using RAMdisks with Applesoft from the point of view of CHAIN, STORE, RESTORE, and program/data overlays.*

Likewise, the IIgs will run ***Apple Writer****, but* ***Apple Writer*** *won't take advantage of the extra memory in a IIgs without an update. I don't know if Paul Lutus and Apple have an update forthcoming, although one would certainly be welcome. Meanwhile, don't overlook the incredible power that* ***Apple Writer*** *already has. If there is a field in your file that is different for every record, you can use that field as a marker and load a file of any size—no matter how big—into* ***Apple Writer*** *in pieces.*

Consider these options in today's ***Apple Writer****:*

```
[L]oad: /pathname
[L]oad: /pathname\
[L]oad: /pathname#

[L]oad: /pathname!start marker!!
[L]oad: /pathname!!end marker!
[L]oad: /pathname!start marker!end marker!
[L]oad: #!start marker!end marker!
```

The first is the normal load command. The second allows you to peek into a file without actually loading it into memory, according to the manual (I've never gotten this one to work, myself). The third

allows you to load a catalog into memory.

The four commands in the second set allow you to load part of a file into memory. The first loads the part of the file between a marker and the end of the file; the second the part from the beginning of the file to the marker. The third will load an interior section between two markers. All of these can have "a", "n", or both appended on the end of the command. "A," with a start marker and an end marker, will load **all** sections of the file between such markers, "n" will prevent **Apple Writer** from loading the markers themselves.

The final command loads a section of **the file in memory** into the file in memory—it's a copy command.

Apple Writer's save command isn't quite so flexible. To rebuild the edited pieces into one large file, sequentially save the pieces into the same file, placing a "+" at the end of the pathname. For example:

```
[S]ave: /pathname+
[S]ave: /pathname!end marker!+
```

The first will append the entire file in memory onto the end of /pathname. (**Apple Writer** will ask if you want to delete the old file named /pathname. Trust me, say yes, but please begin with files you have backups of.) The second appends only that portion of the file in memory between the cursor and !end marker!.

I've said it before, but I'm on a roll so I'll say it again—if you have an Apple II you should have AppleWorks. People who are using II-Pluses or clones should look at our September 1985 issue, page 57, to learn how to get AppleWorks running on their machines. Since 1985, AppleWorks has been an **essential characteristic** of the Apple II. It defines what an Apple II is. It is as much a part of the Apple II as is Applesoft or the Monitor itself (that's why **Open-Apple** never italicizes any of those words).

AppleWorks is a very fast, very flexible program that will do 95 per cent of what beginners want to do with computers, 80 per cent of what intermediate users want to do with computers, and 50 per cent of what advanced users want to do. It is incomparably easier to set up a mailing list with AppleWorks than it is with **General Manager.** It is incomparably easier to teach a novice how to write a letter with several character widths, underlining, and boldface using AppleWorks than it is with **Apple Writer.**

Nevertheless, as your letter points out so well, **Apple Writer** is a must-have program for intermediate-level-and-up Apple users, especially programmers. Don't think of it as a word processor, but as a text file editor. And given Word Processing Language, it's really a **programmable** text file editor. It's an extremely powerful program, and I probably don't talk about it nearly enough here in **Open-Apple.**

Since you're worried about support for **General Manager,** I suggest that you begin by trying to move your mailing list over to AppleWorks. Based on what I've just said, I hope you're convinced you should have AppleWorks anyhow, so this suggestion costs nothing. It will read text files right into the data base, so all you have to do is convert the files from DOS 3.3 to ProDOS. Depending on the number of names in your list, you will either have to use several files or you will have to buy a RAM card and expansion software so that all your names can fit in one file.

You will be amazed at what AppleWorks can do. Not only can it sort your mailing list on any field, it will sort 5,000 names in less than two minutes. That's power, and that's why it's memory-based, rather than disk-based as the **General Manager.**

OK, I grant you AppleWorks can't print four-up labels, but it can print one-up labels to a text file, making most of the neat adjustments you are used to. Writing an Applesoft program that reads this file four labels at a time, truncates and pads the strings as needed, and prints them four across is almost trivial. I don't know why there aren't half a dozen public domain programs around to do it (maybe there are).

It is entirely possible that after you've tried AppleWorks, however, you will find it unsuitable for what you want to do. No problem. Unlike the many other programs that turn out unsuitable for a particular project, AppleWorks won't collect dust. You'll find other uses for it.

Since I myself have yet to have a data base problem that I couldn't solve with AppleWorks, I am unfamiliar with the more advanced ProDOS-based data managers that are available, so I'll throw that part of your question out to our readers. What are you folks using and is it any good?

Finally, as to your question about the relative advantages of an accelerated, memory-carded IIe compared to a IIgs. If you already have the IIe, and if you only use Applesoft, AppleWorks, and **Apple Writer** (a big if), I think you are better off sticking with the IIe. Applied Engineering's TransWarp accelerator will make the IIe run faster than the IIgs and the strengths of the IIgs aren't relevant to those particular programs. If you are buying a new computer, on the other hand, and you can get one, go for the IIgs. Price-wise the IIgs, with all its built-in equipment, is an incredible bargain compared to the IIe.

Inverse logic

I have a problem with the Apple IIe 80-column card. I am using an enhanced IIe. The problem is that when INVERSE mode is on and the screen scrolls, the new blank line at the bottom of the screen will be in inverse. This doesn't happen when the card is turned off. For example, with the cursor at the bottom of the screen, enter INVERSE : PRINT "HI" : NORMAL. Try it with the 80-column card both on and off and look at the difference.

I am able to get around the problem, somewhat awkwardly, by HOMEing the screen and VTAB-HTABing to the lower part of the screen, and being sure that no scrolling takes place. Is there a better solution?

David Robins, M.D.
San Francisco, Calif.

There are definitely differences between how Apple's original 40-column screen firmware and its later 40/80-column firmware work. The "clear-to-end-of-line" routine in the newer firmware checks the inverse flag. If inverse is on, the inverse space-character is used to clear the rest of the line. The older firmware (as well as most non-Apple 80-column cards) always clears with a normal space-character. During a scroll, the clear-to-end-of-line routine is used to clear the entire bottom line of the screen. Thus, if the screen scrolls while inverse is on, you get a bar across the bottom of your screen.

The way to avoid the problem from Applesoft is to turn off inverse before allowing the screen to scroll. Compare your example with this one: INVERSE : PRINT "HI"; : NORMAL : PRINT. Note the semicolon after the first print statement—it keeps the screen from scrolling while in inverse mode.

Down the wrong path

How can I run two or more kinds of system file from one ProDOS disk? For example, I have the following files in one disk, like this:

```
MY.DISK                volume directory
   PRODOS                 system file
   MD.SYSTEM              system file
   SUB1                   directory file
      PRODOS                 system file
      BASIC.SYSTEM           system file
      STARTUP                Basic program
```

When I type "-/MY.DISK/SUB1/PRODOS", I always get the "MD.SYSTEM" file being executed. Why, and how do I solve the problem? (I am trying to execute the STARTUP file.)

Yuh-yuan Wu
Changua City, Taiwan

The problem you're having is caused by executing the PRODOS file. When the file PRODOS is run it always looks in the main or **root** directory for the first system file (type "SYS") with a name ending in ".SYSTEM". In your case, the first file fitting this criteria is the file "MD.SYSTEM". On this particular disk, this will always be the system program PRODOS executes at startup—no matter which subdirectory PRODOS itself comes from—thus, there is no reason to have more than one copy of PRODOS on a disk.

To run the STARTUP file in SUB1, enter instead:

```
PREFIX /MY.DISK/SUB1
-BASIC.SYSTEM
```

When Basic.system starts up, it looks in the active, or **prefixed**, directory for a file called STARTUP; if it finds one, it executes it. The reason you must set the prefix to /MY.DISK/SUB1 is that if you don't, the active directory will still be /MY.DISK, and there is no STARTUP file in that directory. This is why simply typing "-/MY.DISK/SUB1/BASIC.SYSTEM" won't work, either.

Sorting long DOS catalogs

I use large volumes on a Sider—400K, DOS 3.3 and ProntoDOS. Is there a program that will alphabetize the catalog in these large volumes?

Clarke Hottel
Mattapoisett, Mass.

Neither Dennis nor I know of a commercial program that will do this. If you're interested enough to write such a program yourself, a good starting place is the **DOStalk Scrapbook,** by (ahem) Weishaar and Kersey. Pages 192-205 include sample programs that load the image of a standard DOS catalog (sector-by-sector) into the Apple's memory so that it can be rearranged. A specific example of alphabetizing catalog entries is not included, but Applesoft routines to swap two filenames and to sort by file type are; all you need to do is to modify these to do a complete sort on a longer catalog.

You'll need to modify the book's standard "read or write a catalog" subroutine (described on pages 192-193) so that it begins by looking in track 17, sector 0 (the Volume Table of Contents), second and third bytes, for the track and sector of the first catalog sector. The track and sector of the second catalog sector are stored in the second and third bytes of the first one, and so on (pages 164 to 171 describe this). Just keep reading until the track byte comes up zero, which means you're at the end of the catalog.

The sorting routines will have to be generalized to

allow for catalogs of various lengths. And put in a trap somewhere so that a really long catalog doesn't get loaded in over the top of something important.

Does IBM sell Apples?

Your comments about rural America's access to Apple products are very true (September 1986, page 2.57). Apple terminated its exclusive Apple dealer in nearby Asheville, N.C., and I now have only a multiple line (IBM and others) dealer to attempt to do business with. I "almost" perceive a contempt from the dealer toward a business person who would consider Apple as an answer, rather than the "true business" stuff from IBM and others that he sells. Statements come up such as "what kind of accounting program could be of any value on an Apple, when it costs under $200." I personally do not look forward to working with this kind of dealer. I wonder where Apple's thinking was parked when they selected an IBM dealer to be their sole representative here.

Henry Bacarisse
Mars Hill, NC

Multiplan to SuperCalc 3a

Is there anyone out there who knows how to convert Microsoft's Multiplan (DOS 3.3) files to SuperCalc 3a (ProDOS) files? If so, please tell me and save me a lot of work.

Meir Wiengarten
Brooklyn, N.Y.

No one here knows. It has always struck me as funny that in the IBM-compatible arena Microsoft is one of the companies primarily responsible for making things compatible, while in the Apple II kingdom, most of Microsoft's stuff is in a world of its own.

Promal praised

In the September 1986 ***Open-Apple*** (pages 2.60-2.61) you talk about various languages available for the Apple II. You mention Kyan Pascal, ZBASIC, Micol BASIC, and Applesoft compilers. I was really surprised that you failed to include what may be the premier Apple II development language—PROMAL (Programmer's Micro Application Language) from Systems Management Associates (PO Box 20025, Raleigh, NC 27619 919-878-3600).

PROMAL is a block-structured language, like Pascal, without all the silly syntax requirements. It is compiled, includes a complete development environment, is ProDOS based, and is source code transportable to the Commodore 64/128 and IBM-PC. The end user version is only $49.95, the developer's version is $99.95 (no royalties!), and you get a 30-day trial.

PROMAL is one of the best kept secrets around. I urge all your readers who are serious about their programming to give it a try.

Richard Rettke
Appleton, Wisc.

Dennis replies: We still haven't covered all the languages for the Apple II; for the 6502 there's still LISP, FORTRAN, Forth, C, and the various versions of Logo; via CP/M there's COBOL, APL, and PL/1 (just for starters). But we're glad to include the mention of another inexpensive development language.

Modularity (what you call "block-structure") is the strong point of languages such as Pascal and Modula-2; the ability (nay, requirement) of building a program from small, individual procedures makes it a lot easier to follow what's going on.

The knock against Pascal has usually been the use of the ";" delimiter to denote the separation of Pascal statements. Indentation methods have been stressed for use with Pascal, but are not required, so that the following two programs are identical to the compiler's eyes:

```
        procedure writename;

        begin
           write('This sentence is ');
           writeln('no longer split!')
        end
```

and

```
        procedure writename; begin write('This
sentence is '); writeln('no longer split!') end
```

The first is a lot easier for humans to read, though. You can get away with the non-indented form in Pascal because ";" separates the statements and "procedure," "begin," and "end" are recognized as reserved words by the compiler. Such things as the lack of a need for ";" before the "end" terminating the procedure can be confusing.

*According to some reviews (see **inCider**, June 1986, pages 98-99) and the PROMAL ad, PROMAL uses the indentation scheme alone for parsing its modules. This gets rid of the need for things like ";", but won't let you write versions of a program similar to the second example above. But, then, who wants to write unreadable programs? Some people will no doubt complain that remembering where to indent is hard, though, as the **inCider** reviewer did.*

The 3.5 disk notcher

After months of waiting and drooling over it, I am happy to announce that I am the proud owner of one of Apple's new UniDisk 3.5's. On the specification sheet for the drive, it states that any disk used should be double-sided. As you well know, a standard 5.25 inch floppy can be notched, enabling you to use both sides. I've had no problems with the floppies I've notched. Is it possible to use both sides of a single-sided, double-density 3.5 inch disk with a UniDisk 3.5? Or do I have to go out and buy double-sided disks for it? Also, could you please explain what "double-density" and "single-density" refer to on disks?

Randy Vose
Shorewood, Ill.

Using both sides of single-sided 3.5 disks is even easier than with the 5.25s—you don't need to do anything but slip them in the drive. I've been using "single-sided" 3.5s almost exclusively and I've yet to get an I/O ERROR. This whole disk business is very confusing. I assume "double-density" and "single-density" refer to the number of bits that can be stored on a unit of disk media, but I don't know how many it is or at what point it starts to make a difference. The important thing is that, when it comes to the Apple II, the least expensive disks you can find seem to work just fine.

How many drives have we here?

Is there a way to check, from Applesoft, how many disk drives are hooked to an Apple II?

Doug Tuccinardi
Wappingers Falls, N.Y.

No, and yes, but mostly no. You can find all the disk controller cards in a computer's slots without much trouble, but determining how many drives are hooked to each controller is difficult to impossible, as is finding "phantom" controllers.

Under either DOS 3.3 or ProDOS you can figure out which slots hold disk controllers by checking the slot ROM for the correct ID bytes ($20 at $Cs01, $00 at $Cs03, $03 at $Cs05, where "s" is the slot number). Here's how:

```
10 FOR SLOT=1 TO 7
20 ADR=49152 + (SLOT*256)
30 IF PEEK(ADR+1) = 32 AND
      PEEK(ADR+3) =  0 AND
      PEEK(ADR+5) =  3 THEN
        DEVICE(SLOT) = 1 :
        PRINT "DISK CONTROLLER IN SLOT ";SLOT
50 NEXT
```

These are the same ID bytes the Monitor uses to find a disk drive to boot when you turn on your computer. That search begins at slot seven and works down. The Monitor passes program control to the first device it finds that has the proper ID bytes. The device responds by booting the disk in drive 1. Usually the first device found is in slot 6, but if you put a disk controller in slot 7 it will (usually) be the one that boots.

Usually, not always, because Apple recently changed this identification protocol slightly. Before the enhanced IIe, there was a fourth ID byte ($Cs07=$C3). All floppy disk controller cards and many third-party hard disk controllers have this fourth ID and will autoboot on both older and newer machines. Apple's own UniDisk 3.5 and ProFile hard drives, however, autoboot only on the enhanced IIe, IIc, and IIgs.

The reason for this is that the UniDisk 3.5 and the ProFile have only the first three of the four ID bytes. The UniDisk, in fact, uses the fourth as a protocol converter ID byte ($Cs07=$00). The reason Apple changed from four ID bytes to three was that our old friend Apple Pascal assumes that any device that has all four ID bytes is a floppy disk. To keep Apple Pascal from mistaking a newer device for a floppy it was necessary to change the protocol.

(Third-party developers, such as the Sider hard disk people, responded to this problem by patching Apple Pascal, which has to be updated anyhow to respond even to Apple's own devices. This solves the problem as long as you are using the patched version of Apple Pascal that's stored on the hard disk, but can still create problems with mistaken identity when you use standard Apple Pascal booted from a floppy.)

Everything but Apple Pascal determines what kind of disk drive is connected to a controller by looking at byte $CsFF. A zero here means the device is a 16-sector floppy. An $FF would indicate an old 13-sector floppy, if such a thing still existed. Any other value indicates routines for reading from and writing to the device are stored on the controller card itself. (The entry point for these routines must be at $Cs00 plus the value stored at $CsFF.)

Using this information, we can add a line to our earlier program so that it returns DEVICE(SLOT) equal to 0 if there is no disk controller, to 1 for a floppy controller, 2 for an intelligent controller, and 3 for a protocol converter, like this:

```
40 IF DEVICE(SLOT) = 1   AND
      PEEK(ADR+255) > 0  AND
      PEEK(ADR+255) <255 THEN
        DEVICE(SLOT) = 2 :
        IF PEEK(ADR+7) = 0 THEN
          DEVICE(SLOT) = 3
```

A significant problem, however, is that this program

can't identify disk drives in "phantom" slots. It will never find the /RAMdisk ProDOS automatically puts in slot 3, drive 2 on 128K Apples, for example, because that device doesn't have a controller card. Likewise, controller cards that support more than two drives are supposed to make drives three and four ***appear*** *to be drives one and two in slot one or two—but this program will never find them.*

An even more significant problem is that even if you know which ***slots*** *hold disk controllers, there is no reliable way to determine how many* ***drives*** *are connected to any one controller. The Apple II was designed without an electrical signal for this.*

One solution many have tried is to attempt to read a disk and assume there is no drive connected if an I/O ERROR results, but this doesn't work reliably. Often an I/O ERROR will mean that there is no disk in the drive or that the drive door is open, rather than that the drive doesn't exist.

Central Point Software used a test in version 5 of Copy II Plus that consisted of starting a floppy drive, reading the data latch, and watching for changing data. The method was abandoned in version 6 because it wasn't fully reliable.

The latest version of Glen Bredon's ***ProSel*** *includes a program called SCAVENGE that tries to read all the drives ProDOS thinks are connected to a machine. The ones that can't be read are removed from the ProDOS device list. On my system (which has both floppy and UniDisk 3.5 controllers with only one drive each) I have SCAVENGE run automatically when I boot. This makes ProDOS lots faster when it starts searching through the drives for a certain disk (it doesn't try to access non-existent devices and wait the several seconds it takes for them not to respond). However, SCAVENGE also usually removes my floppy from the device list because I have left the door open, and I have to run it a second time after closing the door.*

In summary, a sure fire way to identify how many drives are really connected to an Apple II would be quite useful, but no one I know of has come up with one yet.

Open-Apple

[redacted]

is written, edited, published, and

Business Consultant Richard Barger
Technical Consultant Dennis Doms
Circulation Manager Sally Tally

Open-Apple
P.O. Box 7651
Overland Park, Kansas 66207 U.S.A.

ISSN 0885-4017. Published monthly since January 1985. World-wide prices (in U.S. dollars; airmail delivery included at no additional charge): $24 for 1 year; $44 for 2 years; $60 for 3 years. All back issues are currently available for $2 each; a bound, indexed edition of Volume 1 is $14.95. Index mailed with the February issue. Please send all subscription-related correspondence to:

Open-Apple
P.O. Box 6331
Syracuse, N.Y. 13217 U.S.A.

Subscribers in Australia and New Zealand should send subscription correspondence to ***Open-Apple***, c/o Cybernetic Research Ltd, 576 Malvern Road, Prahran, Vic. 3181, AUSTRALIA.

Open-Apple is available on disk for speech synthesizer users from Speech Enterprises, P.O. Box 7986, Houston, Texas 77270 (713-461-1666).

Unlike most commerical software, ***Open-Apple*** is sold in an unprotected format for your convenience. You are encourgaged to make back-up archival copies or easy-to-read enlarged copies for your own use without charge. You may also copy ***Open-Apple*** for distribution to others. The distribution fee is 15 cents per page per copy distributed.

WARRANTY AND LIMITATION OF LIABILITY. I warrant that most of the information in ***Open-Apple*** is useful and correct, although drivel and mistakes are included from time to time, usually unintentionally. Unsatisified subscribers may return issues within 180 days of delivery for a full refund. Please include a note from your parents or children confirming that all archival copies have been destroyed. The unfullilled portion of any paid subscription will be refunded on request. MY LIABILTY FOR ERRORS AND OMISSIONS IS LIMITED TO THIS PUBLICATION'S PURCHASE PRICE. In no case shall I or my contributors be liable for any incidental or consequential damages, nor for any damages in excess of the fees paid by a subscriber.

Open-Apple is neither affiliated with nor responsible for the debts of Apple Computer, Inc.; "tinaja questing" is a trademark of Don Lancaster.

Source Mail: TCF238 CompuServe: 70120,202

Several sorts of questions

Is there a way to EXEC a ProDOS system program (such as FILER)? When I try it, it seems like Basic.system goes away.

Does anyone have a machine-language array sorter that works with Beagle Bros *Extra K*?

Beware! The improved directory sorter from Copy II Plus version 6.6 will not sort large ProDOS directories (more than 255 files). It sorts the first 255 and then banishes the rest. Then, just for fun, it puts some permanent and illegal file names at the end of the sorted directory. I have notified Central Point and they are looking into it. Glen Bredon's CAT.DOCTOR has a limit of 110 files, but rather than crashing it tells you you have too many.

In response to George Rolla's question on sorting street address by name then number (June 1986, page 2.39); this is how I do it:

```
A$(0)="Main Street   123 Smith John"
A$(1)="Main Street  1107 Jackson George"
A$(2)="Main Street    11 Jones Paul"
```

Notice how we leave spaces when the number of digits is less than five. Then:

```
MID$ (A$(x),9,5) = the house number
LEFT$ (A$(x),8) = the street name
RIGHT$ (A$(x),15) = the customer name
```

Glenn Calderone
Fountain Valley, Calif.

EXEC is a creature of Basic.system. When you execute any other system program, such a FILER, that system program replaces Basic.system totally. Thus, you can't use EXEC to feed commands to ***any*** *system program other than Basic.system itself.*

We haven't yet heard of a machine language sort for ***Extra K****. See the next letter, though, for a faster Applesoft-based sort that will work.*

What are you doing with more than 255 files in one subdirectory? What are you doing with more than one screenful of files in one subdirectory? Dennis just tested Copy II Plus version 7.0 and says it alphabetized the first 255 files, left the rest unalphabetized, and did not trash the subdirectory.

Sorting out sorts

Bubble sorts give me the hives. An insertion sort is much faster, just as short, fairly easy to understand, and "stable," meaning you can use it in a multi-key sort, such as the street-name-and-numbers problem in your June 1986 issue (page 2.39-40).

Here are two examples. The first is for sorting a one-dimension array A(x), the other is for working with several lists kept in a two-dimension array. The second uses a directory to save some time shuffling things about:

```
900 REM --------------------------------------
901 REM  INSERTION SORT, one dimension array
902 REM    A(x)=array to be sorted
903 REM    E1=first element to be sorted
904 REM    EN=final element to be sorted
905 REM    also uses E, ET, T
906 REM    to sort strings use A$(x) and T$
909 REM --------------------------------------
910 FOR ET = E1+1 TO EN
920    LET T = A(ET)
930    FOR E = ET-1 TO E1 STEP - 1
940       IF A(E) <= T THEN 970
950       LET A(E+1) = A(E)
960    NEXT E
970    LET A(E+1) = T
980 NEXT ET
990 RETURN
```

```
900 REM --------------------------------------
901 REM  INSERTION SORT, two dimension array
902 REM    A(k,x) = array to be sorted
903 REM    D%(x) = index to sorted array
904 REM    K = key field to be sorted
905 REM    E1 = first element to be sorted
906 REM    EN = final element to be sorted
907 REM    also uses E, ET, T
908 REM    to sort strings use A$(k,x)
909 REM --------------------------------------
910 FOR ET = E1+1 TO EN
930    LET T = D%(ET)
940    FOR E = ET-1 TO E1 STEP -1
950       IF A(K,D%(E)) <= A(K,T) THEN 970
960       LET D%(E+1) = D%(E)
965    NEXT E
970    LET D%(E+1) = T
980 NEXT ET
990 RETURN
```

Jim Parr
Bloomington, Ill.

Dennis says he must have been out of sorts the day he recommended the bubble sort for the street-name-and-numbers problem. "I was relying on a comment I read in Sara Baase's ***Computer Algorithms*** *that an insertion sort's worst-case execution time was the same as a bubble sort. Unfortunately, I forgot that the worst case seldom occurs. The insertion sort does operate significantly faster than the equivalent bubble sort in real situations."*

Dennis also points out that ***Peeking at Call - A.P.P.L.E., Vol. 3 (1980)*** *includes an article called "Comparing Ten Sort Algorithms" by David Weston. In this comparison, the insertion sort was the fastest for any kind of list with less than 20 items, the fastest for any size list that had been pre-sorted (the test added five random items to a sorted list), and fastest all-around sort that was both "stable" and didn't require a major increase in memory use.*

A sort called Butterfly-Hart is both stable and more than 20 times faster than the insertion sort on long (more than 2,000 items) random lists. However, it requires two additional arrays the size of the array being sorted.

In your second example, we should point out that what actually gets sorted is the directory array, D%(X). To sort a 10 by 20 string array on field 2, then on field 1, and print it you would do something like this:

```
400 E1=1 : EN=20
410 K=2 : GOSUB 900
420 K=1 : GOSUB 900

500 FOR I = 1 TO 20
510    FOR J = 1 TO 10
520       PRINT A$(J,D%(I));",
530    NEXT
540    PRINT
550 NEXT
```