Open-Apple™

April 1987
Vol. 3, No. 3

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

Releasing the power to everyone.

Sending files by carrier bunny

Gary Little, author of Brady Book's *Inside the Apple IIc, Inside the Apple IIe,* and *Apple ProDOS: Advanced Features for Programmers;* author of Addison-Wesley's upcoming *Exploring the Apple IIgs;* contributing editor to *A+;* and author of Pinpoint's *Point-To-Point* communications package (I was very impressed with Little until I found out he's also a lawyer), recently developed a special file format, called Binary II, to make Apple II communications easier. He has placed the format, as well as two Applesoft programs that convert ProDOS files into and out of the format, in the public domain. If we can talk everyone in the Apple II kingdom into using this format for programs and data they post on bulletin boards and in commercial data bases, we'll all be speaking the same language.

The Apple II kingdom needs Binary II for communications because it allows all the various types of files we have (ProDOS SYS, BIN, BAS, DIR, AWP; DOS 3.3 B, A, and T to name a few) to be easily transferred from one computer to another by modem and phone line. In addition to simply transferring the data in the files, it also leaves each file's attributes (lock/unlock status, file type, filename, creation date, and so on) unchanged. Since modem-phone transfers usually take place with the help of a local bulletin board or commercial data base service that doesn't know SYS from SOS, Binary II uses a standard binary file format, which such boards and services can't complain about. It also allows programs that require several different files to be bundled together and transmitted in a single package.

Until Binary II appeared, transfers of Apple II files were usually done by converting all other types of files into text files. This has been the case even though many of the available Apple II communications programs actually include protocols for sending files without doing this. The problem with previous protocols was that they were all different from one another. In order to use one you had to have the same communication software at each end of the phone line. This is at least inconvenient and sometimes, as when using a commercial data base vendor as an intermediary, impossible.

The convert-it-to-a-text-file method, on the other hand, works, but just barely. It's a pain in the chips for the sender, it usually triples the size of the file, which means more money and time for transmission, and it requires that the receiver EXEC or otherwise process the file to convert it back to its correct format. Sometimes receivers find they don't have enough information to do this. Even when conversion works, other problems arise. For example, AppleWorks word processing files lose their formatting with this procedure. And it's nearly impossible to send an AppleWorks spreadsheet file this way.

Little's file format encapsulates one or more files of any type within a Binary II file. When stored on a ProDOS disk, a Binary II file looks like a standard BIN file. Any communication software can send or receive this type of file using XMODEM or any other transfer protocol. Other operating systems and computers think Binary II files are standard binary files. Binary II files, however, always have a name that ends with ".BNY" so that we Apple II people can easily recognize one of them on an alien bulletin board. BNY is pronounced "bunny" because after you bring one home it can quickly produce a litter of little files.

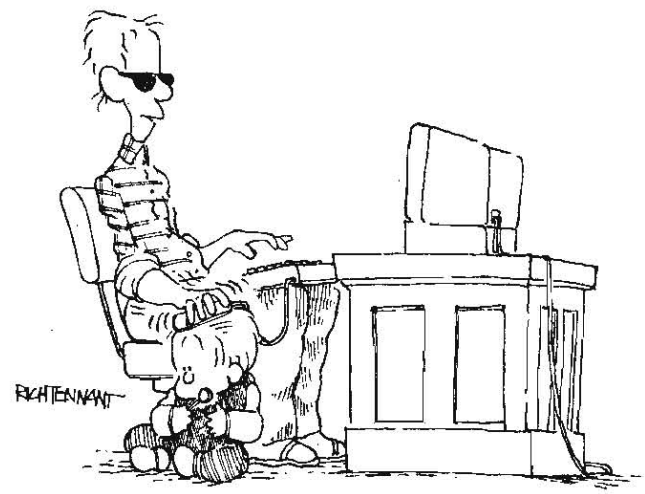
Every BNY file begins with a 128-byte "header" that holds all the attributes for the first file that has been encapsulated, as well as a few bytes of information about the BNY file itself (how many files are in it, how much disk space they require, and so on). After the header you'll find the first file. If the file won't divide into even 128-byte chunks, bytes holding zeros are added to the last chunk until it's an even 128 bytes long. If there is only one file inside the BNY file, the BNY file ends here. If there are more files, they follow in the same order: a header, the file itself, and filler zeros. A BNY file can even hold one or more subdirectories. A BNY file is never more than 511 bytes-per-file longer than the original files.

To create a BNY file you can use Little's public domain program or version 1.5.1 or higher of Little's *Point-To-Point.* The public domain program asks for the names of the ProDOS files you want to encapsulate (they must all be in the same subdirectory) and creates a BNY file that holds all of them. The BNY file can then be sent to a host computer. The host unwittingly stores the file and allows others to download it. Those who download can then split the BNY back into exact duplicates of the original ProDOS files with Little's second public domain program.

Since Little has placed the Binary II format in the public domain, anybody can write a communications program that automatically sends files in BNY format and that automatically splits BNY files into duplicates of the original files as downloading progresses. But *whether other publishers do this or not,* you can use *any* communications software to send and receive BNY files. If your software doesn't specifically support BNYs, however, you must use Little's public domain programs or the equivalent before sending and after receiving.

Little's programs are stored (as text files that can be EXECed into Applesoft) in the MAUG library on CompuServe. Search for the files BI2UP.EXE and BI2DWN.EXE. When you EXEC these they become BINARY.UP, which is for wrapping files in a BNY, and BINARY.DWN, which is for unwrapping a BNY. If you'd like complete documentation on the internal structure of Binary II files, also download BINARY.DOC.

The AppleWorks 2.0 upgrade program is scheduled to end on April 30, 1987. To upgrade an earlier version, ask your dealer for a special mailer. Enclose $50 and your original disks. Apple will send you new manuals and disks. I strongly suggest you keep a copy of your earlier version of AppleWorks in a safe place—it will do a few things 2.0 can't do, as you'll learn when you read this month's letters.



"COME ON, MOM. WHY DOES THE MOUSE PLUG INTO THE KEYBOARD IF YOU'RE NOT SUPPOSED TO USE IT LIKE THIS?"

Need a program for teaching physics with Apple IIs? *Datasheet* is a very high-quality public domain program that was designed with that in mind. I recently saw the program being used by students and was very impressed. It provides a 6-column by 160-row "spreadsheet" for data entry and for statistical analysis (means, standard deviations, least squares and power series fitters). It includes facilities for automatic data acquisition from lab instruments and can plot hand-entered or acquired data, even in real time. And it has survived the test of heavy use by students at a college that has labs full of Apple IIs—the State University of New York at Geneseo. For more information about the program contact Kenneth Kinsey or David Meisel, Dept of Physics & Astronomy, SUNY, Geneseo, NY 14454.

Polarware/Penguin Software hasn't changed the commercial version of *The Graphics Magician* for some time, but they've continued to enhance the version that they use in-house and license to software developers. The current developer version creates graphics in a special format that makes them quickly compatible with and take full advantage of the strengths of both standard and double-high resolution Apple II formats, as well as of Macintosh, IBM-PC, Commodore 64, Atari ST, and Amiga formats. Compatibility with Apple IIgs super high-res is upcoming. For more information about the developer edition of *The Graphics Magician* contact Mark Pelczarski at Polarware (Box 311, Geneva, IL 60134 312-232-1984).

Those of you interested in genealogy might want to check out a $35 program called *Personal Ancestral File* published by the Church of Jesus Christ of the Latter-day Saints. I haven't actually seen the program, but a mini-review of it in the Appleton, Wisconsin user group newsletter made it sound very interesting. For more information, call 801-531-2584 and ask for an order form. Make sure you tell them you're interested in the Apple II version of the program.

Software Publishing Corporation, creators of the PFS series of products, is apparently giving up on the Apple II market. According to a letter that PFS Product Support sent to one of our subscribers in mid-February, "The PFS products have not been tested on the Apple IIgs. It is not planned to update the products for that machine, or to drop copy protection on the current releases of the program."

***II Computing* has ceased publication.**

Ask (or tell) Uncle DOS

You are encouraged to turn to February's page 3.7, pencil in hand, and change the two references to POKE 47246, one half-way down the first column and the second near the top of the middle column, to POKE 47426. The two and one of the fours were accidentally transposed.

Insert system disk and...

I'm confused by a murky problem with my Apple IIe and IIgs, and I'm hoping you might be able to shed some light on the cause.

With the Apple IIe I use a ProFile hard drive. My problem is that if I turn on my modem (a Hayes Smartmodem 1200) after booting the ProFile, or turn off the modem before turning off the computer, my screen almost always goes to garbage. It shifts to 40 columns of gibberish and at the bottom of the screen is the message, "Insert system disk and restart—Err 01". As you can imagine, I've learned to be very careful to turn my modem on first and off last. It does not happen consistently, which is the most puzzling part of all. I think, but am not certain, that it happens only if I have loaded *ASCII Express*; but in that case it happens whether I've actually used the modem or not.

Recently I added an Apple IIgs to my collection of Apples and I'm suddenly having a similar problem. After doing quite a bit of work successfully, I suddenly had, during the middle of saving an AppleWorks document, the same problem as above, with the same error message. It happened another time when ESCaping from an AppleWorks document back to the main menu. I lost the documents both times. Before I got my 3.5 drive for the gs I was using AppleWorks successfully from a 5.25 drive with ProDOS 1.1, and had no problems.

I'm suspecting there is some problem with interrupts, but in one case I'm using ProDOS 1.1 and a modem, in the other case ProDOS 8 with a clock. Do you or your subscribers have any diagnoses, suggestions, or solutions?

C. L. Roberts
Lafayette, Calif.

I suspect interrupts, too. I think the problems you mention with the IIgs are the more serious ones and I can confirm that they are real—they've happened to me, too. I suspect more people aren't complaining simply because current versions of AppleWorks are supplied with ProDOS 1.1.1 rather than with a version of ProDOS 8. ProDOS 1.1.1 disables interrupts. ProDOS 8 does not. Then again, maybe you and I are the only two people in the world who have experienced this strange behavior.

One solution is to stick with ProDOS 1.1.1. On the IIgs, however, this means you lose the clock. A second solution is to use the newer versions of ProDOS on the IIgs but manually disable interrupts. This means you lose the control panel (unless you temporarily turn interrupts back on before pressing control/open-apple/escape). You also lose Appletalk and some other IIgs features. Consequently, disabling interrupts can't be a permanent solution, but I think it will be helpful while Apple comes to grips with the fact that it has a problem with interrupts and comes up with something to do about it. Although the 6502 microprocessor and its progeny have always supported interrupts, Apple II firmware and operating systems have a long history of interrupt-related bugs.

There is simply no reason ProDOS should bomb the whole system in response to an unclaimed interrupt. It would be much more reasonable to let the user know that an unclaimed interrupt occurred and then to continue trucking. If such interrupts continue to occur the user should have the option of disabling interrupts so that work can be saved, reset can be pressed, and the system restarted. Most unclaimed interrupts are odd, one-time affairs, such as the interrupt you're getting when you turn your modem on or off. To destroy a user's work in response to something like this is absolutely user-hostile and not the kind of treatment users have come to expect from Apple IIs.

In addition, all programmers—those at Apple and the rest of us, too—must become more aware of interrupts and more careful not to change the microprocessor's interrupt status accidentally. For example, on a IIgs turn interrupts off and press control/open-apple/escape. Then proceed to use your computer. If the control panel suddenly appears, it means the program you are running just turned on interrupts. I've found several that do so for no apparent reason.

*Here's a couple of short programs you can use to change the interrupt status of your computer until Apple gets ProDOS's response to unclaimed interrupts tamed. Delete line 40 from both of them if you don't use a program selector such as **ProSel**, **Bird's Better Bye**, or **Squirt**.*

```
10 TEXT : HOME : VTAB 12
20 PRINT "Disabling interrupts."
30 POKE 768,120 : POKE 769,96 : CALL 768
40 PRINT CHR$(4);"BYE"
```

```
10 TEXT : HOME : VTAB 12
20 PRINT "Enabling interrupts."
30 POKE 768,88 : POKE 769,96 : CALL 768
40 PRINT CHR$(4);"BYE"
```

*Another possibility, using AppleWorks on the IIgs anyhow, is to use **Super MacroWorks**. Among its many other features, it gives AppleWorks 2.0 the ability to recover from a control-reset. Control-reset will recover your Apple from the unclaimed interrupt error and has the additional beneficial effect of turning off most interrupting devices.*

*With AppleWorks 1.3 you can use the rescue routine from the June 1986 **Open-Apple** (page 2.33) after pressing control-reset. Living Legends Software (1915 Froude St, San Diego, CA 92107, 619-222-3722) sells a similar routine on disk that can be permanently placed on your AppleWorks disk and run automatically. It's called RESET.SYSTEM—several of our subscribers have recommended it.*

AppleWorks phone dialer

Here's a simple AppleWorks database report format you can use to automatically dial phone numbers, assuming you have a modem that will respond to ASCII command strings. This usually means an external, rather than a slot-resident, modem.

The first step is to tell AppleWorks you have a "printer" in the slot that holds the serial card for your modem. Do this in the usual manner—from the Main Menu select "Other Activities," then "Specify Information About Your Printer," then "Add a Printer," then select any of the listed printers, such as the ImageWriter. Give it a useful name such as "modem," or "dialer," or "dial phone." Specify the slot your modem is connected to. At the next menu all that needs to be changed is the code for "interface cards." Change that code to "none" by simply pressing shift-6 (the caret) to end the code-input sequence.

Secondly, create a database file that holds your list of phone numbers. This list should have separate

categories for name, area code, and phone number. It could have any additional categories you desire. You have to separate the area code from the rest of the number to make it easy to dial locally.

The third and final step is to create three report formats that will be used to actually "print" phone numbers to the modem and cause it to dial the phone. The first will be used for long distance calls outside your area code, the second for long distance calls within your area code, and the third for local calls.

Use the "labels" format for all three of these reports. First create a new labels-type report and call it "dial other area." Using open-apple-O(ptions), change the Paper Length to 1 inch, get rid of the report header (PH), and tell AppleWorks you want to send Special Codes to the printer (SC). Then enter the codes "ATDT1" for touchtone or "ATDP1" for pulse dialing. ESCape out of options and use open-apple-arrows to move the area code and phone number to the top line of the report. You should open-apple-D(elete) most of the other categories and blank lines, but I've found it useful to display at least one other category, such as Name, on the line below the line with the phone number. The carriage return at the end of the first line should keep any stray numbers in one of these extra categories from being dialed, but if you reach Khaddafy when trying to call your mother, you may have to delete the extra categories.

Now ESCape back to the "report menu" and chose item four, "Duplicate an existing format." Name the next report format "dial this area." Delete the area code category from this format. Use this format for long distance calls within your own area code.

Again ESCape back to the "report menu" and duplicate the "dial this area" report format. Name the duplicate "dial locally." The only change you need to make is to remove the "1" from the Special Codes in open-apple-O(ptions), leaving ATDP or ATDT. That is the end of the creation process.

To actually dial a number while using AppleWorks, open-apple-Q(uick change) to your phone number database, open-apple-P(rint), choose the appropriate dialing format, and use open-apple-R(ecord selection) to choose a specific number to be dialed. Usually "1. Name, 7. Contains" is enough, but you may have to "and" that with something else in order to get just one record.

Once you have a record selected, open-apple-Z(oom) so that you can check that it's the right one. You can use open-apple-1 and open-apple-9 to make sure that only one record has been selected. From there, press open-apple-P(rint), choose the printer called "dial phone" or whatever, and print 1 copy.

When the dialed phone starts ringing, pick up your receiver and open-apple-P(rint) the phone number a second time. This second printing gets the modem to hang up.

Jim Hercules
Johnstown, Ohio

Wow! It works as advertised.

There's one small problem that might crop up on the Apple IIgs or other machines that have a disk device, as well as a modem, that appears to be connected to slot 2 (this happens automatically on the IIgs when more than two devices are connected to the slot 5 Smartport). According to Steve Stephenson, author of Checkmate Technologies' AppleWorks expansion software, AppleWorks won't recognize a printer if it is assigned to the same slot as a disk device. Thus a slot-2 modem may be inaccessible from AppleWorks on the IIgs, even though Basic.system would have no problem with both a modem and a disk device "in" slot 2.

I'm especially impressed with your open-apple-Z(oom) and open-apple-1 through -9 trick. I hadn't previously noticed that those commands work on the "Report Format" screen.

Another way to make a dialer would be to create just one "dial phone" report format. Embed ATDP or ATDT within its Special Codes and put all the other required "dial strokes" in a special category in each record. This requires duplicated information in each record, but simplifies the calling process.

It also allows you to use long distance companies that require access codes. Around here, for example, we make long distance calls by first calling a local seven-digit number, then entering a six-digit access code, then entering the ten-digit phone number we want. These 23-digit calls are particularly fun to automate. However, the Special Codes section of open-apple-O(ptions) accepts only 13 characters, which isn't enough for the "AT" code and the long distance company's local number and access code.

Instead, I tried putting all 23 digits in a special category called "dial strokes." It was also necessary to add a couple of commas between the long distance company's phone number and its six-digit access code. These commas make the modem pause, which gives the long distance company time to answer the phone before AppleWorks starts punching in the access code.

The next step, of course, is to write some macros that automate the whole calling process. Solid-option-M could call your mother, solid-option-B your boss, and solid-option-L Don Lancaster's free Apple helpline (602-428-4073).

The year 2000 problem

In using the AppleWorks database for genealogy, I have found that AppleWorks will not accept a year ending in 00 in a "date" field. If I change the "Date" field to "Dte" it will accept everything, but without the automatic date procedure. The automatic dating procedure is really great except for that little hitch.

Another problem I've encountered occurs when more than 15 categories have been deleted from a report format. If you then try to reinsert a category, you'll find the selections for "insert a line above the cursor" and "insert a line below the cursor" actually run off the right side of screen. What's worse, if you move the cursor to either of these choices the program hangs and the only way out is rebooting.

This does not diminish nor detract from AppleWorks, I love it and use it every day for everything.

One idea I've found useful is to create a "NBR" field as the first category of every file and use it to assign each record a unique number beginning with 10001. I find using this category to select specific records really saves time and frustration compared to using the name of a person or place or whatever.

Another idea I've used effectively is to create two fields for addresses, one for the house number and one for the street. By sorting them both I end up with a file in the correct logical order. Similarly, my genealogy file has a lot of Thornburgs. Since AppleWorks will only sort on either the first or the last name, I transferred all the Thornburgs to a temporary file, sorted the first names, and then reinserted the Thornburgs into my original file and had all the names in first name and last name order properly.

Jim Thornburg
Alexandria, Va.

Wait a second, you're doing more work than you have to. Although AppleWorks will only sort one field at a time, you can do what other programs call "multi-key sorts" by sorting all the fields you are interested in, one-at-a-time, in least-important to most-important order. For example, first sort house numbers, then street names. Or first sort first names, then last names. The last sort you do is the primary sort and, for example, all your Thornburgs will appear together. Within the Thornburgs, however, records will be arranged the way they were by the previous sort—first names in this case.

I hadn't noticed AppleWork's inability to accept a year ending in double zeros before. That will be a major problem real soon now. I don't think changing the name of the category is a good final solution because then you lose the ability to do chronological sorts. How do you handle dates before 1900? It seems to me that AppleWork's two-digit years would be problematical in genealogical applications.

***I have** encountered the problem with AppleWorks hanging when you try to insert a blank line in a labels-style report format. This bug has been fixed in AppleWorks 2.0. If you use an earlier version of AppleWorks, try to remember to save your work before modifying report formats. If you should encounter the bug with unsaved work on the desktop, press control-reset and see the last paragraph of the answer to this month's first letter.*

Row too long to hoe

I have a large spreadsheet in AppleWorks. When I was developing it I got a message that some cells were lost in one row. After saving the file and trying to read it later, I now get a message from AppleWorks that it is getting errors trying to read the file.

What would cause such problems and how can I recover the file?

William C. Piquette
La Junta, Colo.

Thanks to a letter from subscriber Pete Johnson, we can tell you exactly how to duplicate this error.

Using AppleWorks version 1.3 or earlier, set up a new spreadsheet. Put the value "1" in cell A1 then enter "+A1" as the formula in cell B1. Copy B1 from C1 to DW (the end of row 1). You'll get a message that "Some cells were lost from Row 1." Save the spreadsheet file, then try to reread it. "Getting errors trying to read <filename>" will result.

Another, more insidious error: Start as above, but copy the formula only to the range C1 to CS1. No error is mentioned, but after saving the file it can't be reread, either.

Now copy the formula from cell C1 to CR1, then enter a label in CS1. AppleWorks will stop you after eight characters. Entering "abcdefg" is legal, but entering "abcdefgh" results in a "cells lost" error. However, in this case there is no problem saving and reloading the file; the label is automatically truncated back to an acceptable length.

Now manually enter the formula "+CR1" into cell CS1 (over the label). AppleWorks gives you an error and ignores the entry. But if you blank the entry first, AppleWorks accepts the manual entry with no error, but won't allow you to reload the spreadsheet after saving it. Yet if you blank out cell CS1 after the error, you can save and recover the resultant file.

*These anomalies all stem from a maximum limit on the number of data bytes that can be contained in a single AppleWorks spreadsheet row. This limit isn't discussed in Apple's manuals, though it has been mentioned in **Open-Apple** (August 1985, page 58).*

The bigger problem, however, is that the maximum limit for the number of bytes you can enter from the keyboard and ***save*** *to disk is higher than the maximum limit you can* ***load.*** *Thus the incredible situation like yours can occur where a spreadsheet you have created can't be reloaded.*

Fortunately, there is now a fairly easy method available to recover the file. Find someone who has AppleWorks 2.0 and an Apple Memory Card or the equivalent. When used with expanded memory, AppleWorks 2.0 allows more data bytes per spreadsheet row than earlier versions. It will readily load your "damaged" spreadsheet. Blank out a cell or two in your longest row and resave the spreadsheet. It should now load into an older version of AppleWorks.

Realize, however, that this very ability creates a new incompatibility—it's possible to create AppleWorks 2.0 spreadsheets that can't be loaded into earlier versions of the program (or even into 2.0 running on a computer without expanded memory) because of too much data in a single row. If you share your templates with others this can become significant. Of course, it also goes without saying that spreadsheets that use new features of version 2.0, such as the ability to include @AND and @OR inside @IF statements, won't run on earlier versions.

AppleWorks 2.0 subtleties

I recently upgraded from AppleWorks version 1.1 to version 2.0. I was sure that all my old AppleWorks spreadsheet files would work with 2.0, however, that was not the case. The differences are subtle, but they do make a difference in the final spreadsheet. The two differences that I have run into so far are as follows:

	Version 1.1	Version 2.0
Cell A2 (Formula)	@IF(A1=4*5,1,0)	@IF(A1=4*5,1,0)
Cell A1	20	20
Cell A2 (Display)	1	0
Cell A1	@NA	@NA
Cell A2 (Display)	NA	ERROR

In the first example, the only way to make version 2.0 respond the same way as Version 1.1 is to put the "4*5" in parentheses, like this: @IF (A1=(4*5),1,0).

In the past I've used a technique based on the second example to change ERROR to NA where my spreadsheet has division by zero, but it no longer works.

I sure would like to know if anyone else discovers such inconsistencies in the AppleWorks spreadsheet.

James R. Leth
Palos Verdes Peninsula, Calif.

We checked and found that AppleWorks versions 1.2 and 1.3 act the same as 1.1. Interestingly, ***Advanced VisiCalc*** *acts like AppleWorks 2.0 with your first example (it requires parentheses around 4*5), but it acts like the earlier versions of AppleWorks with your second example (it passes through an NA that is referenced by an @IF comparison rather than converting it to an ERROR).*

To make a division-by-zero show up as NA rather than ERROR, try this, where "divisor" and "dividend" are cell references: @IF(divisor=0,@NA,dividend/divisor).

Nulls nullified in 2.0

I use AppleWorks to report both U.S. currency sums and Sterling sums. What I previously did was to configure two "printers" (both actually the same Imagewriter) with different interface card control sequences. The control sequences switched the Imagewriter between the U.S. character set and the U.K. character set, which includes the "£" (I'd like to see if you can print that in ***Open-Apple***).

When I tried to duplicate this configuration with AppleWorks 2.0, I found that "control-@" (the null code, hex $00), which is needed to change character sets, can no longer be entered. To get around this I copied the SEG.PR file from my old version of AppleWorks to the 2.0 disk. This seems to work but I am left confused by the whole deal. Why is Apple removing functionality from AppleWorks?

Tony Bond
Herts, England

Slow find, fast sort

We have one IIe running an expanded version of AppleWorks 24-hours-a-day. The computer is dedicated to one data base file. It is used by non-computer people; open-apple-F(ind) is their most important tool. If they have to stare at a blank screen for more than a few seconds, they will ignore the file entirely, and that would, sooner or later, be disastrous. Never mind that before implementing the file it took them three to ten minutes to dig the information out of a series of card files. They've forgotten all about that. And since this file will be around 500K before the end of 1987, the problem can only get worse.

Consequently, any speed differences between AppleWorks 1.3 and 2.0, as expanded with Applied Engineering's desktop expanders, are of great interest to us. After a series of tests I've concluded that there are significant speed differences. AppleWorks 2.0 is about twice as fast at loading and sorting files, however, the open-apple-F(ind) command is about three times slower on the newer version.

I've also concluded that *Super MacroWorks*, doesn't slow anything down in spite of the size of the files it adds to AppleWorks. There is, however, a minor bug in *Super MacroWorks* that prevents one from stopping the printer in mid-flight. Load the MAIN.MENU program and change the 208 in line 18 to 240 (fix courtesy of Beagle Bros' excellent technical support department).

And please, no more of this foolishness about some other program being better than *Super MacroWorks*. The control-S(ubdirectory) macro, which eliminates the need to type pathnames, is worth twice the cost of the program alone. This new version is a real barn-burner. Try it.

W.M. Patterson
Davis Wharf, Va.

Many unhappy returns

Since we have an Apple Laserwriter connected to a Mac and an Apple IIe with AppleWorks, it is most useful for us to be able to transfer AppleWorks word processor documents to the Mac for printing on the Laser printer. This can be done with the *Mac.Transfer* software from Southeastern Software (7743 Briarwood Drive, New Orleans, LA 70128, 504-246-8438). The program costs about $50, requires an Apple IIe to ImageWriter cable (usually), and uses a Super Serial card as a sending mechanism. AppleWorks files are changed to ASCII and sent to the Mac.

Your reference to the August 1985 article about formatted AppleWorks files in response to the February 1987 letter "AppleWorks vs chemistry" (page 3.4) suggests that you may not be aware that AppleWorks 2.0 places returns at the end of each line when you print a word processor document to "a text (ASCII) file on disk." Previous versions didn't do this—they put returns only where you had entered them manually.

Mervin Brubaker
Dillsburg, Penn.

You're right. I'm becoming disenchanted with the disenhancements that are turning up in AppleWorks 2.0. This one alone is reason enough to make sure you keep a copy of AppleWorks 1.3 around. As the previous letters show, AppleWorks 1.3 is also handy for creating SEG.PR (printer configuration) files with nulls in them and for doing faster finds.

Inside Applesoft

I would like to know if there is any way a user program can access the ROM routines in Applesoft directly, like a CALL or some other way.

Kenneth Noel
Madison, Ala.

It's both possible and common for programmers to access the Applesoft ROM routines from assembly language. However, Apple itself doesn't provide any information on how to do this, probably because Applesoft was written by and licensed from MicroSoft.

Roger Wagner Publishing, however, provides a program with its Merlin assembler called SOURCEROR.FP that will print a disassembly of Applesoft. Similarly, S-C Software sells a package that will allow you to generate a source listing of Applesoft compatible with its assembler.

Applesoft ROM calls are usually made from ampersand routines, although it is also possible to use a CALL. For some examples, see "Input absolutely anything" in our September 1985 issue, pages 65-68, and the follow-up information in the October 1985 issue, page 74.

Two other more complete sources of information are ***Assembly Language for the Applesoft Programmer****, by C.W. Finley, Jr. and Roy E. Myers (Addison-Wesley) and* ***Call -A.P.P.L.E. In Depth #1: All About Applesoft.***

Random numbers

The *Apple II Reference Manual* speaks of a location in memory to which 1 is added repeatedly while waiting for input from the keyboard. The resulting number is used as a random number seed for programs. Please tell me where this location is and how to use it.

Glenn Golter
Wheaton, Ill.

The Apple Monitor leaves random values in bytes 78 and 79 ($4E-$4F) every time you call on it for a keystroke. While waiting for the keystroke, the Monitor changes the values stored in these bytes thousands of times a second. The changes stop the instant the Monitor senses a keystroke. The randomness comes from the impossibility of controlling the amount of time between keystrokes to the degree that would be necessary to foil randomness. Try this:

```
10 GET A$ : REM rerandomize values
20 PRINT "BYTE 78 IS ";PEEK(78)
30 PRINT "BYTE 79 IS ";PEEK(79)
40 GOTO 10
```

There are three limitations to these random numbers. One is that somebody has to press a key for each new random number pair. Another is that the numbers you get will always be between 0 and 255 inclusive; if what you really want is a number between 1 and 10 you'll have to do some careful

mathematical manipulation to convert what you get into what you want. The third is that you can't repeat a series of random numbers for test purposes.

The good part is that the numbers you get will be absolutely unpredictable—which is more than can be said for even the best software-based random number generators (Applesoft's is one of the worst, incidentally). For much more on random numbers, see my column in ***Softalk****, April 1984, page 52, and the chapter "Random Number" in Don Lancaster's* ***Assembly Cookbook for the Apple II/IIe****, page 345.*

Telling Pascal time

Where does Pascal maintain the current date? I have attempted to control and update the time from within my programs and have never completely succeeded. I have learned that the date is retained in bytes 20 and 21 of block 2 of the disk directory in a compact bit format. But, because under certain conditions I keep receiving the old, non-updated date, I wonder whether it is not also duplicated somewhere within memory.

Roger R. Eddy
Concord, N.H.

I turned this one over to Dennis, who, with Bill Basham's help, came up with this:

Apple's UCSD Pascal operating system keeps the date in memory as well as on disk. The date format is the same two-byte format used by ProDOS (Pascal does not keep track of the time). This format uses 7 bits to designate the year, 4 bits to designate the month, and 5 bits to designate the date for a total of 16 bits. The bits are organized like this:

```
Hi byte  Low byte
--------  --------
YYYYYYYD  DDDDMMMM
```

As the bytes appear in memory, their order is reversed in typical 6502 fashion. To find the date in memory, I set the date with the Pascal FILER, then popped into the IIe's monitor and did a memory search. The date is usually found in two locations with this search: the lower memory location is apparently left over from the date's position within the FILER program's memory image, the second is the date's actual location in the resident p-system program. The location is different for various versions and memory configurations of Apple Pascal:

Version	64K	128K
1.1	$AA1B	<NA>
1.2	$ACFC	$BB5A
1.3	$ACFE	$BB5A

IIc printer problems

I have an Apple IIc system with an additional 512K memory from CheckMate and an ImageWriter printer. I use the large memory as a ProDOS RAM disk to store, among other things, AppleWorks. Everything works fine. I often have to reboot with DOS 3.3 to use some of my software, and then I reboot with ProDOS. A nice feature of the RAM disk software provided by CheckMate is that the RAM disk isn't lost during these reboots; AppleWorks is still in memory and works fine—except for the printing.

Somehow, rebooting with DOS 3.3 and then ProDOS confuses the serial ports and produces garbage (usually unprintable characters) in place of normal printouts. Hitting the RESET key doesn't help nor does turning off the printer. Running the Apple IIc Utility Disk, which resets the serial ports, usually fixes the problem. The most reliable fix, however, is to turn the IIc power off, but then I lose the RAM disk. What's happening and how can I fix it?

S. A. Klein
Madison, Wisc.

The Apple IIc keeps some information about its serial port settings in the auxiliary memory "screen holes." There's nothing about booting either DOS 3.3 or ProDOS that changes these values, so I suspect it's one of your DOS 3.3 programs that is causing the trouble. The simplest solution is to simply fix the values after rebooting with ProDOS and before running AppleWorks.

For a little program that shows you how to figure out what the values are supposed to be and how to reset them, see the November 1985 ***Open-Apple****, pages 86 and 87.*

Fontrix user group

I am the programmer of Apple *Fontrix* and president of the Apple Fontrix Club. The club is unaffiliated with Data Transforms, publisher of *Fontrix*. Membership is $30 ($15 renewal) and includes a monthly newsletter, a Graffile catalog, two free Graffiles from the catalog, and royalty payments for any Graffiles you submit and which are sold from the catalog.

Rod Nelsen
Apple Fontrix Club
PO Box 29857
Thornton, CO 80229-0857
(303) 451-7577

UniDisk manual eject

You *can* get a 3.5 disk out of the drive with the power turned off (see "IIgs questions and observations", January 1987, page 2.94). There is an access hole directly below the eject button. Take a paper clip, straighten it, and press it into this hole and the disk will be ejected.

Ronald W. Jones
West Chicago, Ill.

Well, yes, but it takes a good deal of pressure, about an inch of paper clip, and feels a bit like you are bending something you shouldn't. In Apple 3.5 drives the hole goes right through the eject button.

All chips not off same block

We have a very active user group here in Dhahran (about 300 members, mostly employees of ARAMCO) and are well supported by the local Apple dealer, more so than any dealer I dealt with in the states. I now have an Apple IIgs, (one of two I know of in the Kingdom) with an AE gsRAM card. Even here I have managed to locate a couple of pieces of IIgs software. I also have AppleWorks 2.0 and the new Applied Engineering Appleworks expander. I am waiting for *Pinpoint 2.0*, which I ordered a couple of weeks ago. A Pinpoint product I did receive, and that works very well, is *Point-to-Point*. This communications package works with the built-in slot-2 port on the IIgs with no problems.

I did have one problem in using the slot-2 port with my older Apple modem—finding a cable that would work. It is way too early for the Apple dealer over here to begin receiving this type of accessory, but he came through for me with an old-Mac to Imagewriter II cable that has a 9-pin connector on the Mac end and the small 8-pin connector on the other. I just turned it around and it works fine.

I've had one other problem I want to point out. I bought my gsRAM card with a meg on board, as I had two sets of 256K at home to bring it up to 1.5 meg. After installing those chips, the machine kept crashing. So I removed my own chips and everything was OK. Then I removed 512K worth of chips from a RAMWorks board (they were the same brand and number as the chips on the gsRAM board) and installed them on the gsRAM board. This worked. I then placed the aftermarket chips in the RAMWorks card. This worked, too.

What it boils down to is that you may be money ahead buying these boards, especially the complex ones, with the chips supplied by the manufacturer. I would have hated to have bought the gsRAM board with only 256K on it and then buy 1.25 meg of chips that wouldn't work.

Anyway, I like the IIgs. Apple's going to sell a zillion of them.

Claude A. Allen
Dhahran, Saudi Arabia

We have had other reports that not just any 256K chip will work on cards that reside in the IIgs memory expansion slot, but we haven't yet heard a good explanation of what the difference is. Until someone who knows explains it to all of us, I'd recommend that extra chips for IIgs cards be purchased cautiously.

Time to look in the toolbox

I have a couple of programs that run under ProDOS 1.1.1 but that won't run under ProDOS 1.2. In order to allow these programs to use the IIgs clock I wrote a special clock driver for ProDOS 1.1.1. I thought some of your readers might be interested in the heart of this driver, which demonstrates how to make a toolbox call on the Apple IIgs. It goes like this:

```
00/0300: 18           CLC
00/0301: FB           XCE          Put that sucker
00/0302: C2 30        REP  #$30      in 16-bit mode.

00/0304: F4 00 00     PEA $0000    PEA pushes the two
00/0307: F4 00 00     PEA $0000      operand bytes on
00/030A: F4 00 00     PEA $0000      the stack--make
00/030D: F4 00 00     PEA $0000      room for 8 bytes
                                     of output.
00/0310: A2 03 0D     LDX #$0D03   ReadTimeHex cmd
00/0313: 22 00 00 E1  JSL $E10000  Call tool locator

00/0317: 68           PLA
00/0318: 8D 30 03     STA $0330    seconds/minutes
00/031B: 68           PLA
00/031C: 8D 32 03     STA $0332    hour/year
00/031F: 68           PLA
00/0320: 8D 34 03     STA $0334    day-1/month-1
00/0323: 68           PLA
00/0324: 8D 36 03     STA $0336    filler/day of week

00/0327: 38           SEC          Back to 8-bit mode
00/0328: FB           XCE
00/0329: 60           RTS
```

William A. Coleman
5904 NW 49th St
Oklahoma City, OK 73122.

I'm including your address so anyone who wants your whole clock driver can contact you. It's been suggested that I include an address with every letter I print, but I don't want to discourage anyone from writing and I don't want to cut myself out of good "message threads"—would printing everyone's address have either of those effects?

It might be good to remind everyone that our August 1986 issue went into what many of you thought was excruciating detail on 65816 assembly

language. Coleman's program uses a number of commands that will be new to 6502 assembly language programmers—you may find that issue's information helpful.

For now, let's just say that the first three lines of Coleman's routine are the standard way to go from "6502 mode" to "65816 mode," and the last three lines are the standard way to go back.

The general format of a IIgs toolbox call is to push some bytes on the stack for any information the tool will return, then push some bytes on the stack with any data the tool needs, then load X with the tool number and, finally, JSL (jump to subroutine, long) to the IIgs tool locator, which is always at byte $0000 in bank $E1. After the tool call, any data passed to the tool will have been stripped off the stack and the calling program can just start pulling off results. All toolbox calls, incidentally, must be made from full 16-bit mode.

In this example, Coleman uses the PEA instruction to push eight bytes on the stack for information that will be returned by the tool. PEA pushes the two bytes that follow the command onto the stack. I tend to think of it as a "push immediate" command, but it's really one of three new commands for pushing absolute, indirect, and relative addresses onto the stack without disturbing the contents of any registers.

All the programs I've seen so far use PEA $0000 to reserve stack space for the data a tool will return. On the other hand, PHA in 16-bit mode also puts two bytes on the stack, but takes up only one byte in the program, compared to PEA's three. There doesn't seem to be any reason why the reserved space has to hold zeros. Can anyone enlighten us as to why people are using PEA rather than PHA?

The number of the tool that reads the clock and returns eight hex bytes on the stack is $0D03. Apple's manuals name this tool ReadTimeHex. If a program switched back to 8-bit mode before pulling the time off the stack, the data would come off the stack in this order—seconds, minutes, hour, year, day-of-month (0 to 30), month (0 to 11), one garbage byte, and day-of-week (1 to 7, 1=Sunday).

To enter Coleman's program from the IIgs Monitor, use a CALL -151 in the usual way from Applesoft, then enter "!" and press return to get into the IIgs Mini-Assembler. To enter the first line of the program, type "300: CLC". The 300 tells the Mini-Assembler where to store the program as it assembles it. After that, simply enter a blank space and the assembly language command—the program will be stored in successive bytes.

To get out of the Mini-Assembler, press return on a blank line. If you try to use 300L to list the program, it won't look right. This is because the program is being listed as if the program was running in 8-bit mode rather than 16-bit mode. To correct the listing, enter the following bit of magic, "0=e 0=m 0=x". It's important that you use lower-case letters. To go back to 8-bit listings use, "1=e 1=m 1=x".

To run the program, try this, "300G =T 330.337" (the spaces are important!). This will run the program, print the current time on the screen using a special IIgs command (=T), and show you the contents of bytes 330 to 337. These bytes are where Coleman's program stores the stack information returned by the tool call. Convert the hex bytes to decimal and you should be able to quickly grasp how they relate to the time shown on the screen.

I haven't yet had much experience with the IIgs toolbox, but Coleman's program prompted me to try a few things, including reading a page or two of the developer draft of Apple's toolbox reference manual (November 5, 1986 version). I hope the documentation on ReadTimeHex isn't an indication of the quality of the whole manual. There are two significant mistakes on the page about this command. The first is that it shows ten bytes being pushed on the stack rather than eight (the last two are called "buffersize"—what buffer?). The second is that it says the day-of-week is returned as a number between 0 and 6, with 0 being Sunday. It's really 1 to 7, with 1 being Sunday.

There's another IIgs time tool that returns the time in ASCII rather than in hex. There's also a mistake in the documentation about it—it indicates you should push a two-byte address on the stack pointing to where you want the tool to place the ASCII string. In fact, the program requires a four-byte address. Here's a little program that calls this tool:

```
00/0300: 18           CLC
00/0301: FB           XCE          16-bit mode
00/0302: C2 30        REP #$30

00/0304: F4 00 00     PEA $0000    Address where tool
00/0307: F4 30 03     PEA $0330      should put time.
00/030A: AD 03 0F     LDX #$0F03   ReadASCIITime
00/030D: 22 00 00 E1  JSL $E10000

00/0311: 38           SEC
00/0312: FB           XCE          8-bit mode
00/0313: 60           RTS
```

This tool doesn't return any data on the stack, so after the JSL there's no pulling to do. The pushing that's done in this program passes information that the tool will use (where to place the ASCII string). To run this program and see the results, type "300G =T 330.343" after entering the program.

A neat feature of the ASCII time tool is that the time is returned in whatever format has been selected by the user. Three date formats (mm/dd/yy, dd/mm/yy, yy/mm/dd) and two time formats (AM/PM, 24-hour) are available on the IIgs. The user selects his or her preferred format with the control panel. This is the tool to use if you want to display the time within a program.

Odd bank out

Programmers should know that if they are printing with a PR#1 or PR#2 their output can be garbled if the INVERSE flag is on. The INVERSE/NORMAL flag (location $32) not only affects screen output, it also scrambles the data going out the port. I work for a firm that uses a form of Braille desktop publishing (we send data disks to a Braille printing house that has an Apple IIc plugged into a $50,000 machine that produces metal plates for Braille embossing). Our software occasionally had the inverse flag on. The result was over $6,000 worth of Braille with all the key words garbled. Sometimes these little bugs can get quite expensive.

Our firm sells a DOS 3.3 word processing system for visually impaired Apple II users. I thought it would be interesting to modify the program to make use of the additional 128K found in the Apple IIgs. My task was to load and save blocks of data to memory banks $E0 and $E1 from an assembly-language memory-management program running in auxiliary memory (bank $01).

The first thing I found was that it was necessary to set the SHADOW register of the IIgs to $1C. This prevents the IIgs from automatically copying what's written in banks $00 and $01 into banks $E0 and $E1 in the range of memory that I was interested in using ($4000-$BF00). You do this by poking $1C into byte $C035.

Next I discovered that memory management was much more complicated than described in the Apple IIgs technical manual. With the 65816 in full native mode, the K register (program bank, i.e. where is the program running) and the B register (data bank, i.e. which bank is being read or written to) control what banks the IIgs is working with. But when the IIgs is emulating an Apple IIe or IIc, the soft switches RAMRD and RAMWRT are used to control banks. When a program is running in main memory, both switches are off, so the Apple is reading and writing to main memory. When a program is running in auxiliary memory, both switches are on and the Apple is reading and writing to auxiliary memory.

I discovered that if the soft switches are on when you use the Apple IIgs in native mode, it makes the B and K registers act as if they were odd, even though they are even. In particular, if a program is executing in auxiliary memory and then switches to native mode, the B and K registers both hold zeros. However, your program will read and write to banks $01 and $E1 because the soft switches make B and K "act" odd.

My program switches into native mode, sets the B and K registers to $01 (which matches what the IIgs is already making them appear to be), turns the auxiliary memory soft switches off, and then accesses any part of IIgs memory it wants to using the "long" instruction mode. After the read/write operation is over, everything is undone in reverse order: turn auxiliary memory soft switches on, set the B and K registers to $00, go back to emulation mode. Finally, the program returns control back to main memory using the XFER routine at $C314. Notice that these is no command to change the K register except a "long jump." I use it to jump to the next instruction using the bank number 00 or 01 as appropriate. Here is the code I used:

```
This routine accesses all IIgs memory from AUX mem.
   This routine MUST run in AUX mem. The calling
   program must set the SHADOW register at $C035 to
   $1C so that the IIgs won't overwrite the memory
   area from $4000 to $BFFF in banks $E0 and $E1.
CLC
XCE          Go to native mode.
JML *+4      Jump long to next instruction--use $01 as
               the bank number. This sets K to $01.
LDA #$01
PHA
PLB          Set B to $01.
STA $C002
STA $C004    Turn off AUX mem soft switches.
REP #30      Full 16-bit mode. Don't do this until
               you've changed the soft switches.

---(Load or save to bank $E0 or $E1 goes here.)

SEP #30      Back to 8-bit mode.
STA $C003
STA $C005    Turn AUX mem soft switches on again.
JML *+4      Jump long to next instruction--use $00 as
               the bank number. This sets K to $00.
LDA #$00
PHA
PLB          Set B back to $00.
SEC
XCE          Back to emulation mode.
```

Anyway, that's the first program I wrote using the new op codes in the Apple IIgs. I know that I am certainly not "playing by the rules". The purpose of this trick is to give a program some extra power while you re-tool your programming skills to make full use of the IIgs Toolbox.

David Holladay
Raised Dot Computing
Madison, Wisc.

Good. Now the next thing we need to figure out is how to use the IIgs Memory Manager to set aside the memory you are using in banks $E0 and $E1 so a desk accessory or something doesn't overwrite what you've stored there. I've spent a couple of hours trying to do this with the IIgs Mini-Assembler, but haven't yet figured out how to talk the Memory Manager into giving me a "UserID." That's supposed to happen when you use a tool called "MMStartUp," which every application program is supposed to call as one of its first actions. All the Memory Manager will give me is an "invalid UserID error." Have any of you IIgs wizards figured out how to do this yet? Rich Williams, are you listening? Don't send me any workshop-loader-macro baloney—first I want to know how to do it under ProDOS 8 (or even DOS 3.3) with the Mini-Assembler. That makes it more like an adventure game.

Making connections

Here's a source for the hard-to-find DB-19 plugs and jacks used on Apple II Smartport devices and 5.25 floppy disk drives. One small problem is that they require a minimum order of five and they don't actually stock the item, but at least they can get them for you. The company is Stout Cables Inc., P.O. Box 76, Sewickley, PA 15143 412-741-3900.

Jeffrey Young
Woodside, N.Y.

I called Stout and they can also provide the 15-pin connector needed to build a IIgs RGB cable and the male MINI DIN-8s needed for cables that plug into the IIgs serial ports.

Another source for 15- and 19-pin connectors is JDR Microdevices, 110 Knowles Dr, Los Gatos, CA 95030 800-538-5000.

No endorsement

On page 3.8 the February 1987 issue of **Open-Apple** you mention the program *Fingertips* as allowing rudimentary use of a modem (see "Communications software"). I was unable to get this feature of the program to work. I wrote to Northwest Synergistic on two separate occasions but did not receive a reply. Since the rest of the program isn't worth much I tossed it in the drawer and am contenting myself with spreading the word.

When you mention a product in your publication it can be interpreted as an endorsement. I thought you might like to know what kind of turkeys you are endorsing.

William G. Ingersoll
Ft. Walton Beach, Fla.

Oddly enough, if you go back and read that letter and our answer again you'll see that what we were really trying to do was avoid endorsing any specific product. Dennis and I want to make it clear that we are usually as confused by the plethora of Apple II products as anyone else. Unfortunately, we are also in a position where we have to answer questions regarding these products.

A good chunk of our mail is from people who want to know about the features of a specific product or about the products available to solve a specific problem. We can handle these one of two ways; simply tell you we don't have any direct experience with a product and make no offer of information or try to recommend some products to check.

We usually try to follow the latter path. However, we can do little more than what most readers could do for themselves—read advertising and reviews, consult catalogs, go to user group meetings, log on to bulletin boards, and call or write manufacturers. We have one additional source of information that most people don't have—letters from people like you who give us a feel for how much you like or dislike specific products (and we're feeling you don't like this one).

*We want to make it clear that you should never take the mere mention of a product here in **Open-Apple** as an endorsement; if one of us likes and uses a product, we'll say so. Even then, the fact that one of us likes and uses a product doesn't even mean that the other of us does, much less that you will. Conversely, I've disliked products that readers have thought very highly of, for example, see "**SuperCalc 3a** defended" in July 1986, page 2.45.*

The ghost in the machine

When an Apple II reads a text file, some horizontal lines, maybe a centimeter long, flicker in ghostly fashion along the left margin of the screen. Is this a contrivance to signal that something is happening or is it just one of those little idiosyncrasies of the machine?

William D. Vernon
Odessa, Texas

It's a long story. Consider that Applesoft doesn't know a disk drive from a putter. When you read a text file, all Applesoft sees is a PRINT statement that begins with a silly control-D and an INPUT statement. In response to INPUT, Applesoft calls a built-in routine in the Monitor. That routine, as always, puts a cursor on the screen and jumps to the current input routine. Uncle DOS always has himself hooked in there. Because of your control-D command he knows you want to read a file. So he puts a line from the file in the input buffer and passes control back to the Monitor. The Monitor removes the cursor from the screen and passes control back to Applesoft.

All this happens so fast that the cursor doesn't appear on the screen long enough for the Apple II video circuitry to scan it in its entirety. Usually only one line of the cursor makes it to the screen before the Monitor removes it. That's what you are seeing. I'd have to classify it as an idiosyncrasy rather than a contrivance.

IIgs speed control

I thought your readers might be interested in this: On the IIgs, location 49206 may be poked to change system speed. Poke in a 4 for slow or a 132 for fast.

Barry M. Fox
Hicksville, N.Y.

Whoa, Barry. That's the IIgs configuration register you're playing with. It contains a few other bits of information you don't want to change by accident. A safer way to change speeds is like this:

```
10 REM slow down (if I<=127 then already slow)
11 I=PEEK(49206) : IF I>127 THEN POKE 49206,I-128

10 REM speed up  (if I>=128 then already fast)
11 I=PEEK(49206) : IF I<128 THEN POKE 49206,I+128
```

If you do this from inside a program, don't forget to put the speed back where you found it when you quit.

FIDdle sticks

In your December 1986 issue, page 87, you gave a procedure for using an EXEC file to automate FID for DOS 3.3. It worked fine until I tried to load a large file (135 sectors, *ASCII Express*). Then the procedure got as far as the first "X" after the "N" for no prompts and had to be completed manually.

With the large file deleted the EXEC file would work. Is there any other way to do it?

I've had trouble sorting zip codes with the AppleWorks data base because my file includes a mix of 5-digit and 9-digit zip codes. I've been using the format 99999.-9999 for 9-digit zip codes. Including the decimal point gets the zip codes arranged in the correct order.

Tony Pizza
Camarillo, Calif.

You're right. The EXEC file seems to die an unnatural death when a large file is copied. Nobody here knows why or how to avoid it. Does anyone out there know?

*The mixed 5-digit/9-digit zip code problem is even easier than that to solve. Take out your decimal points. Just sort the zip codes **alphabetically** rather than **numerically** and they'll come out in the order you want. Alphabetical sorts are also **faster** than numerical sorts (within AppleWorks, anyhow).*

NO BUFFERS help I

A comment on John Sklar's question in the February 1987 **Open-Apple** (page 3.7). In my experience with *GPLE* running under ProDOS, the NO BUFFERS AVAILABLE message occurs after I have pressed control-C to escape from a long CAT listing. Without *GPLE*, control-C simply aborts the remainder of the listing except for the free space information. With *GPLE*, which (usually) intercepts the control-C, the listing is immediately and completely ended and control is returned to the keyboard. Unfortunately, this sidesteps Basic.system's normal cleanup at the end of the CAT command, which includes freeing the CATALOG buffer.

Simply issuing a CLOSE command with no parameters is sufficient to solve this problem and get rid of the NO BUFFERS AVAILABLE error. It is often quicker to do this than to wait for the end of a long CAT listing. If you are lucky, Basic.system will see the control-C before *GPLE* and the normal cleanup will occur. The giveaway is whether or not the free space information occurs.

The "Applesoft ONLINE call" letter in the same issue (page 3.6) has one small problem. The program doesn't reset the default slot and drive. This can be a problem if you are working without a prefix to force Basic.system to remember the last used drive, as under DOS 3.3. The cure is to PEEK the contents of $BE3C-BE3D (48700-48701), the default slot and drive in the Basic.system global page, and restore them after issuing any PREFIX commands that change the defaults.

A comment on the IIgs keyboard. My first impressions were a bit doubtful, but I am rapidly growing to like this keyboard a lot. However, this is my first letter since buying the IIgs, and I have just realised the consequences of moving the solid-apple (option) key to the left of the open-apple key—no more single handed large cursor movements in AppleWorks or *Apple Writer*! I had not realised how automatic skipping a word at a time using the solid-apple and arrow keys had become until I started placing numerous backslash characters into this letter.

Peter Watson
Box Hill North, VIC

As I understand it, you would add lines like these

to Kline's VOLUME.FINDER program:

```
15 DS=PEEK(48700) : DD=PEEK(48701)
90 PRINT D$;"PREFIX /"
95 POKE 48700,DS : POKE 48701,DD
```

Line 90 in Kline's original program is redundant because of an editing error by yours truly and can be overwritten by the line above. I think many people don't know they can get Basic.system to behave more like DOS 3.3 by voiding the current prefix with a "PREFIX /" command (the slash is important). After this command Basic.system looks in the last drive accessed for everything, just as DOS 3.3 does.

I agree with your comments on the IIgs keyboard. Things could have been a lot worse, however. The IIgs keyboard is a result of delicate compromises between the Apple II and Macintosh keyboards. The same keyboard comes with Apple's newly-announced Macintosh computers. While Macintosh users got their traditional position for the solid-apple (option) key, Apple II users got traditional placement for everything else of importance, including the arrow keys, which have a different arrangement on the Mac-Plus. This "single-keyboard" strategy will be very important to II users in the future, as any third-party keyboards or other "desktop bus devices" (bar code readers, joysticks, trackballs, and so on) developed for the corporate Macintosh world will be usable in the cottage-industrialist Apple II world as well.

Open-Apple

[redacted]

is written, edited, published, and

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

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

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 single back issues are currently available for $2 each; bound, indexed editions of Volume 1 and Volume 2 are $14.95 each. Volumes end with the January issue; an index for the prior volume is included 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 from Speech Enterprises, P.O. Box 7986, Houston, Texas 77270 (713-461-1666).

Unlike most commercial software, ***Open-Apple*** is sold in an unprotected format for your convenience. You are encouraged 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. Unsatisfied 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 unfulfilled portion of any paid subscription will be refunded on request. MY LIABILITY 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.

ISSN 0885-4017 **Source Mail: TCF238**
Printed in the U.S.A. **CompuServe: 70120,202**

NO BUFFERS help II

Here is another reason for the NO BUFFERS AVAILABLE error when using *GPLE* and *Double-Take*. I have had the same problem with *GPLE* and have traced it to the following.

When installed, *GPLE* marks pages $8A to $99 as busy in the ProDOS bit map. The problem occurs when another program, usually on exit, frees up pages it has obtained through JSR GETBUFR by calling JSR FREEBUFR. Among other things, calling FREEBUFR frees up all buffer pages by moving HIMEM back to $9600, a page that has been marked as busy by GPLE. FREEBUFR does not touch the bit map. Thus, ProDOS is locked out of its own buffer area. Because of this problem, I have removed the routines for marking the bit map from all of my homegrown utilities.

George W. Hernan
Reston, Va.

*Not marking the bit map doesn't solve the fundamental problem here. When your second program calls FREEBUFR to release the memory it has obtained, this call also frees the memory **GPLE** has obtained. At that point in time **GPLE** will still be in use and connected to the operating system. If the bit map is marked as free, then the next time ProDOS uses a file buffer it will overwrite **GPLE** and the system will come down in flames.*

The fundamental problem is FREEBUFR itself. It provides no way to selectively deallocate a buffer. It deallocates all buffers, or, more specifically, all buffers up to a single selectable point.

*Any program that calls FREEBUFR is a danger to all peace-loving people. These programs don't consider what the consequences will be for other programs that have called GETBUFR. The consequence will always be that other programs will be overwritten and will cease to function. **The only safe way to deallocate buffers is to ask the user to re-execute Basic.system.***

For the record, to allocate a buffer simply place the number of memory pages you desire in the accumulator and JSR GETBUFR ($BEF5). If an error occurs the carry flag will be set on return. If the carry flag is clear, you have been allocated a buffer and the accumulator will hold the high byte of its address (the low byte is always zero).

*You can defend yourself from the warmongers who call FREEBUFR by immediately storing the high byte of your buffer minus 4 at $BEFB. The FREEBUFR routine, if called, will then unwittingly destroy only buffers it has allocated after yours. Some people think that by correctly manipulating $BEFB they can call FREEBUFR with no bad consequences for others. $BEFB can only protect those who asked for a buffer **before** you did, however—it can never both release your buffer and protect those who asked for a buffer **after** you.*

*For more on how to live together with other programs in peace, see the June 1984 DOStalk, "A TYPE Command for ProDOS," in **Softalk**, page 157. Apple's ProDOS Technical Note #9 may also be helpful, but it suffers from the delusion that no other program will ask for a buffer before or after you do.*

Memory gotcha II

Last month you published my letter (page 3.16) saying that I prefer the standard-slot type of RAMdisk to the aux-slot type of RAMdisk because I can load the RAMdisk with programs, boot another disk (even from a different operating system), and then later reboot from the RAMdisk without any problems.

I have to take it back. I've been noticing that every once in awhile I would try to reboot the RAMdisk but get an "UNABLE TO START UP FROM MEMORY CARD" error. I'd catalog the RAM disk and find that all of my work files had vanished. I was perplexed by this and consulted a knowledgeable colleague. He pointed out that the Apple Memory Expansion Card (one of the standard-slot memory boards) uses screenholes in the Apple memory to keep track of the status of the RAMdisk. If a screenhole gets clobbered, the RAM disk loses its mind and decides to start all over again. I had been trying to run a variety of programs in my Apple, including copy-protected games, some of which probably care not one whit about preserving screenhole values.

I suspect that the other numbered-slot memory boards (AST Sprintdisk and Applied Engineering RAMFactor) also keep important RAMdisk status information in the precarious screenhole locations, though I don't know for sure. I still use my RAM disk, but now I'm very very careful about what other programs I run if I expect the RAMdisk contents to remain intact.

Phil Thompson
Portland, Ore.

*RAMdisks have always been somewhat fragile creatures. But at least you **can** reboot from the standard-slot type of card. I pointed out last month that you can reboot without destroying the files on aux-slot cards, but I missed the fact that rebooting from an aux-slot RAMdisk itself is impossible.*

Nice kingdom you have here

I'm a new "Apple" owner (I bought a Laser 128 rather than the real thing). I'm not new to computers. I learned programming on a Univac 1102 in 1961; the language was ALGOL. We have had up to four computers in the house at one time and usually I use the IBM-clone in my study. However, since I bought this Laser for my son, the IBM-clone sits with cold chips and my son complains rightly that I'm hogging *his* computer. I can't stay away from it!

For years I refused to buy an Apple because it looked like a ripoff—they were always priced higher than they were worth. However, because of the vast library of Apple educational programs, my son's reluctance to succeed in school, and the low price of the Laser 128, I could refuse no longer.

I've discovered that the graphics are less than great; even a $50 TI-99 could outdo an Apple. The sound is embarrassing to anyone who owned a TI, an Atari, or Commode Door 64. Also, it's sort of ugly inside, with screen memory stuck in the middle of things and strange bugs in one of the oldest DOSs around. But,... I love it!

It has class. That class isn't inherent in the architecture of the machine; it has something to do with the users and the programmers and the companies that support it. The quality of the programming exceeds that of any of the other small PC's. The range and breadth of the available documentation astounds me. The software, like the *Multiscribe* word processor I'm using, is FUN.

I also like ***Open-Apple's*** approach and your writing style. You have class. In fact, as my First Sergeant used to say, "You guys shine like a diamond in a goat's (anatomical expletive deleted)."

Dave Netzer
Lucasville, Ohio