Open-Apple™

Releasing the power to everyone.

September 1987
Vol. 3, No. 8

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

IIgs chip fix announced

Apple sent out postcards to registered owners of IIgs computers in August offering a free chip upgrade. Two chips are involved—the Video Graphics Controller (VGC) and the ROM. The upgrade to the graphics chip will get rid of the pink fringe seen in double-high-resolution monochrome mode (e.g., on the "Desktop") and some other exotic graphic bugs. The new ROM expands features and fixes bugs in the old ROM.

According to an alert I found on Apple's AppleLink message system August 13, units that have serial numbers (found on the bottom of the case) that start with 704 or less need both chips. Units that start with 725 or higher already have both new chips installed. The alert didn't say anything about serial numbers starting with 705 through 724. It did say that the serial numbers given in Apple's August Service and Support Notice for dealers were incorrect.

I haven't actually seen the chips at work—Apple's postcards seem to have arrived in users' mailboxes before the chips arrived at dealers' doorsteps—but those who have seen the chips indicate they should be approached with open arms. There may be some new IIgs software somewhere that isn't compatible with the new ROM, but there shouldn't be. I haven't yet heard any concern about trading old chips in for new ones, as there was with the IIe enhancement.

Though Apple's postcard indicates you can "simply take your Apple IIgs to any authorized Apple dealer" for the free upgrade, I suggest you call ahead to make sure your dealer has the needed chips on hand before unplugging all those wires.

***Open-Apple* Washington correspondent Tom Vier** made an amazing discovery while perusing the *Apple II SCSI Card Technical Reference Manual* (available from APDA—the Apple Programmers and Developers Association). The Smartport assembly language command that causes a 3.5 drive to eject its disk causes a SCSI drive to reformat itself! There are not many programs around that eject 3.5 disks, but if you have one that does and a SCSI drive too, proceed with extreme caution. We have heard of SCSI drives that have been erased by this coincidence.

Programmers—if you have ever written anything that ejects 3.5 disks, go back and look at "More Smartport Commands" in our January 1987 issue, page 2.92. It demonstrates a method for determining whether a unit you are about to send the eject-disk command to is a 3.5 drive or not. If it's not, DON'T DO IT. The method involves issuing a Status Call to the unit and examining a byte the call returns that's known as Device Type. Since that article was written, Apple has issued an updated list of Device Type values (SmartPort Technical Note #4). Here they are:

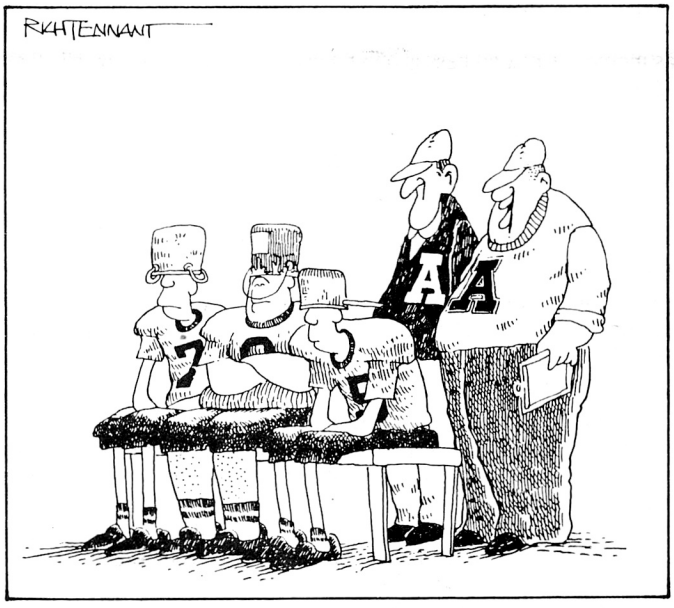
```
$00 memory expansion card
$01 3.5 disk
$02 hard disk
$03 generic SCSI
$04 ROMdisk
$05 SCSI CD-ROM
$06 SCSI tape
$07 SCSI hard disk
$08 SCSI sequential device
$09 SCSI printer
$0A 5.25 disk
$0B reserved
```

Look for scads of new Apple II products to be announced at AppleFest in San Francisco September 18-20. Applied Engineering will be showing off its new PC Transporter card. This card lets you add what just about amounts to an IBM PC-AT to your Apple II. Rumors of this card (known as "Little Blue") have been around for months; my feelings toward it had always been apathetic—why not just buy a PC clone, for not much more money, instead? My attitude changed after reading about the card in the September issue of *A+*, however ("PC Transporter," pages 30-46). The card was designed by a company called The Engineering Dept. This is a group of former Apple engineers who have previously worked at designing things such as the Apple IIe and IIc, ProDOS, and the IWM chip, which is the heart of Apple's 3.5 disk controller.

For starters, look at what the card does when you're running Apple II software. Why, it's a 768K memory expansion card and 3.5 disk controller. You can connect one or two platinum Apple 3.5 drives to the card (the white UniDisk 3.5 won't work). The card allows the Apple 3.5 drives to read and write in your choice of Apple or MS-DOS formats. If you also need to be able to read and write MS-DOS 5.25 disks, you can daisy chain two IBM-type 5.25 drives together with the Apple drives. (Any IBM drive will work in theory, but the card uses Apple-standard 19-pin drive connectors; standard IBM drives have 34-pin connectors. AE plans to sell IBM-style drives with Apple-style cables—and it won't take long for cable adapters to become available elsewhere.)

The card uses a 7.2 MHz NEC V30 microprocessor, which gives you about three and half times more power than a standard 4.77 MHz 8088-based PC. The PC has a full boat of PC RAM (640K) and "Color Graphics Adapter" or CGA graphics hardware. It has a socket for an 8087 math coprocessor. It can use other peripherals connected to your machine, such as printers, modems, mice, and disk drives. (You can store your IBM data on any ProDOS-compatible device, but an MS-DOS computer won't be able to read those disks—for transportability you must use 3.5 inch drives or IBM-style 5.25 drives connected to the PC Transporter.)

The card is expected to sell for about $500. While AE will be showing it off at AppleFest, the only bad news is that shipments aren't expected to start before late October.



"IT WAS BETWEEN THAT AND NEW CLASSROOM COMPUTERS."

While the PC Transporter seems destined to be the Most Amazing New Engineering Feat at AppleFest, there will be much more. For example, if you own a IIgs, you might want to check out the Checkmate Technologies booth. They'll be announcing a new gizmo for the IIgs called Memory Saver. You unplug your IIgs memory card from Apple or Applied Engineering, plug it into Memory Saver, then plug Memory Saver into the IIgs memory expansion slot. It provides battery backup for your memory card and allows you to partition part of your RAM card as a IIgs "ROM" disk. Since what the IIgs thinks is a ROMdisk is really RAM-based, you can easily change what's on it at any time. The price for this device is expected to be about $130. It's not clear as I write this whether it will work with any IIgs RAM card or only those from Apple and Applied Engineering. If it works with all IIgs RAM cards, it should destroy the market for ROM- and EPROM-based piggy-back boards.

On the software side, look for lots of new IIgs software, about half-a-dozen new Apple II desktop publishing packages, lots of AppleWorks add-ons (including Beagle Bros new Time Out series, mentioned here last month), and bunches of new games.

Even ***Open-Apple*** will be announcing four new top-secret products at AppleFest. We've decided to go for the least-expensive new product award—all four will sell for less than $3.01 each. Come by and see us at booth #222. ***Open-Apple*** Technical Guru Dennis Doms will be there to answer hard questions and I'll be there to answer easy ones. My wife and Sally Tally have been insisting on going, too.

The Applefest conference schedule is interesting. One thing that hasn't received as much publicity as it deserves is that Apple will have daily presentations of its "Apple Classroom of Tomorrow" research project. This project has been underway for a couple of years, but Apple has never before opened the project up to the public as it will at AppleFest.

The project involves classrooms in seven school districts around the U.S. In each Apple has set up "ideal" classrooms in which every student and teacher has a personal computer both at home and at school. Kids and teachers from three of the sites will be at the show, each afternoon from 12 to 4, demonstrating what they do at school. From 11 to 12 each day, Martin Engle, who runs the program for Apple, and administrators from the three schools will make presentations on the program.

Another meeting you ***Open-Apple***-types might be interested in will be an APDA-sponsored Apple II Standards conference to be held Sunday from 12 to 2. The meeting will focus on exchanging ideas about what standards are lacking in the Apple II world and what they should be (e.g. Super Hi-Res graphics file formats, interface-card command codes, management of extended RAM).

We have a bunch of AppleFest discount coupons good for $5 off the $15 exhibits-only 3-day ticket or $10 off the $50 exhibits-and-conferences 3-day ticket. If you're going, send us a stamped, self-addressed envelope and indicate how many coupons you want. We must get your envelope on or before September 10. Hotel rooms near the Civic Auditorium are mostly gone already, but doesn't everyone have friends in San Francisco?

Sensible Software, publishers of the spelling checker we've been happily using around here for the last year or so (and lots of other good stuff), has moved. It's new address is 335 E Big Beaver, Suite 207, Troy, MI 48083 313-528-1950.

Not only can you hook a bar code reader to the Apple IIgs "desktop bus" (keyboard connector), you can also get a device that reads the magnetic stripe on credit cards. The bar code reader is $795, the magnetic stripe reader $695, a combination reader $995. Prices decline rapidly if you purchase more than one. Contact TPS Electronics, 4047 Transport, Palo Alto, CA 94303 415-856-6833.

We've discovered one more computerized index to *Open-Apple* in addition to the two mentioned here in July (page 3.43). This one uses the Fastfind program. For more details contact Jim Hammond, 28503 Coveridge Dr, Rancho Palos Verdes, CA 90274.

Educational software at the high-school level and above tends to leave me pretty cold. Most of the stuff I've seen seems to be done by people who haven't quite caught the idea that the computer is a medium of instruction that is different from books, video tape, and blackboards. Good educational software should use the computer to do things that simply can't be done with paper-based media. That's why I'm really excited about a package I recently received in the mail called *The Power of Nation States*. I like it so much that I want to talk about it here as an example of high-quality educational software.

The Power of Nation States is designed for high school classes in Global Studies, World Geography, American History, or World History. It is totally AppleWorks-based. However, it doesn't teach anything about AppleWorks. It assumes that the teacher and students already know how to turn on a computer, startup AppleWorks, and load files. After using *The Power of Nation States*, most teachers and students will know a great deal more about AppleWorks than they did before, true, but that's not a direct goal of the course of study.

The Power of Nation States comes with a teacher guide and a text book. Unlike most teacher guides and text books, however, these are in AppleWorks word processing files on the *The Power of Nation States* disk (or disks—four 5.25 disk sides). The reason the text book is in a file is so that any teacher using *The Power of Nation States* can shorten, lengthen, or otherwise modify the lessons. Teachers are *encouraged* to modify the lessons to fit the needs and style of each class.

At the core of *The Power of Nation States* are two massive AppleWorks data bases. The first, called Trade Culture, contains 24 categories of information about the world's 167 nations, as well as two blank categories for students to use. Categories in this data base include such things as gross national product, percentage of population under 15 years old, population growth rate, religions, exports, export trading partners, imports, import trading partners, ethnic groups, languages, resources, percentage of population unemployed, and life expectancy.

The second data base, called Power, has 28 categories of information about the 167 nations. Some of the information duplicates what's in the Trade Culture data base. Other categories include population, per capita income, political system, alliances, percentage of population living in rural areas, percentage of population with access to safe water, percentage of population able to read and write, land area, total military expenditures, total value of exports, total value of imports, total number of people in a state's military forces, and the number of nuclear warheads a country possesses.

In the first lesson (assuming an unmodified textbook), students are asked (either individually, in small groups, or as a class—depending on how many computers are available) to scroll through a category called "Recognize?" in the Trade Culture data base and enter either YES, NO, or NEVER depending on whether they are familiar with a country, have heard of the country but know nothing specific about it, or have never heard of the country.

After that part is completed, the students are asked to open-apple-A(rrange) that column so that all the YES countries appear together. Then, in a category called "Friendly?," the students are to note what they believe about each YES country in terms of whether it is friendly, hostile, or neutral toward the United States.

Next the students are asked to arrange the data base on a category that specifies whether each country is in the "1st World," "2nd World," or "3rd World." Immediately thereafter they are to arrange it on the "Recognize?" category. The students are then asked to ponder such questions as "Of the states you recognized, how many were 1st world? 2nd world? 3rd world? Of the states you marked NEVER, how many were 1st world? 2nd world? 3rd world? With which world are you most familiar? least familiar?"

Next the students are asked to arrange the "Friendly?" column. "In which world do you perceive the U.S. has the most 'enemies'? In which the most 'friends'? Do the states you marked as 'friends' tend to have one major ethnic group or many ethnic groups?"

The students are asked to use open-apple-L(ayout) to move two categories called Export Trade Partners and Import Trade Partners so that they appear on the multiple-record screen and ponder, "Do we tend to do more trade with states you marked as 'friends'? as 'enemies'? as 'neutral'?"

Finally, the students are to sort the data base on the gross national product category, from 9 to 0, and investigate whether states marked as "friends" and states marked as "enemies" tend to be "rich" or "poor" on a global basis.

At the end of the lesson, each student should have a good idea of the general characteristics shared by states he or she perceives as friends or enemies, and states he or she recognizes or does not recognize.

In lesson two, students compare the U.S. and Egypt, using both the Trade Culture and Power data base files, to determine the seven to ten most significant differences between 1st world and 3rd world countries, assuming the U.S. and Egypt and good examples of such countries.

In lesson three, students compare Japan, the USSR, and Bangladesh to determine the differences between 1st, 2nd, and 3rd world countries, again assuming these are good examples of each type of state.

In lesson four, the students are introduced to the concept of power. The text discusses how power has been defined by a number of authorities and

presents an historical look at the power of states. The computer part of lesson four asks the students to arrange the power data base "from 9 to 0" by population, and open-apple-H(ardcopy) the top 15 countries. Next they are asked to do the same for nine other categories. Using the printouts, students are to determine which 15 countries seem to be the most powerful.

In lesson five, the students are directed to a spreadsheet called the Power Grid. This spreadsheet allows each student to devise his or her own formula for "power." The spreadsheet allows the students to assign points for population, per capita income, total military expenditures, and nine other categories. Then the spreadsheet can be recalculated and the countries rearranged from 1 to 167 based on the results. An important part of this lesson is that "power is an elusive concept, and reasonable people may interpret numerical data quite differently....There is little agreement about the nature of power. While that may be true, it is educationally beneficial to measure power and quantify power among states."

In lesson six, which is the final lesson, students use the Power data base to print a report giving global totals for a variety of categories, with subtotals for NATO and Warsaw Pact countries. They also use a spreadsheet called the Alliance Grid to determine the relative power of NATO, the Warsaw Pact alliance, and of non-aligned nations using the formula they developed in lesson five.

In an addendum, the authors explain how to use another data base file, called Percentages, included on the *The Power of Nation States* student disk. This file shows each country's data as a percentage of the world's total, rather than as an absolute number. This file lets you see not only which ten countries have the highest gross national products, it also lets you see what proportion of the world GNP they contribute. This file also includes a report format that shows the total percentage of the 1st world, 2nd world, and 3rd world for a number of categories.

In the addendum, the authors say, "The Percentages file has more significance than just adding one more way to look at a given state's statistics. It is a model for manipulating data into formats that were heretofore impractical for the high school student. The ability to look at information in ways that are meaningful to you, rather than ways that are meaningful to a book publisher, is quite significant. It is possible for an advanced AppleWorks user to take the basic information in the data bases on the disk, move the data to a spreadsheet, manipulate it mathematically, and move the new data back into an entirely new data base."

And that, in essence, is what excites me about this package. It not only teaches lessons in history and geography, it teaches lessons in thinking and analysis. It not only trains students how to use quantitative analysis, it also trains them to recognize how ambiguous a concept such as "power" can be. One thing missing from the package is a listing of the sources of the numbers in the data bases. I'd think an interesting lesson seven would be to have students go to a library and find out how many different sources of discrepant numbers are available to plug into the data bases and the power formula. Talk about ambiguity.

Two other features I like are having the text book on disk and the price structure of the package. Having the text on disk not only makes it easy for teachers to modify, it also allows the text (and the data bases, for that matter) to include the latest information. Parts of lesson six, for example, were obviously updated within the last couple of months.

The price of *The Power of Nation States* is $95 per site. Purchasers are granted a limited "site license" that allows a school, individual, or organization to make as many copies of the disk as are necessary to make it useful at a given "site." For these purposes, a "site" is considered to be a single school or campus, not a school district or a university system. The disks are not copy protected. The teacher disk includes lesson plans, test questions, an AppleWorks self-assessment questionnaire that indicates whether students know AppleWorks well enough to do the computer exercises, and a data base file for making labels for student disks.

The Power of Nation States is published by Data Disc International, 1430 Willamette, Suite 577, Eugene, OR 97401. The company has offered to send a free demonstration disk, which includes a subset of the Power data base, a major portion of the Power Grid spreadsheet, and some of the text of lessons 4 and 5, to any ***Open-Apple*** subscriber who requests one. Take a look and make copies for all the high school social studies teachers you know. This is an incredible package. It's the first thing I've seen in years that makes me wish I was still in high school. (Wait a minute, why limit this package to high school students—adult-oriented organizations should be interested, too...)

Data Disc is also at work on two other packages that I haven't seen, *The Power of the Presidency* and *Nuclear Decisions*.

Reviewer's Corner

Here at ***Open-Apple***, we spend about $300 a month on typesetting. Each month we have about 70,000 characters of information typeset. I prepare the material with a word processor, using macros to embed typesetting codes into the files. These codes tell the typesetter to switch fonts, to change margins, to use *italics*, and so on. The files are sent to a typesetting company by modem.

There, the files are processed by some very expensive typesetting equipment. The machines figure out my codes, hyphenate and justify my lines, and produce pretty sheets of paper that I can cut up and paste together to make the "camera-ready" "artwork" that becomes ***Open-Apple***.

At least that's the way it works most of the time. This review wasn't sent to our typesetting company. Instead, I wrote this review using AppleWorks, embedded a new set of typesetting codes into the file with a new set of macros, and printed the file on an ImageWriter II with the help of a new program from Data Transforms called *Printrix*. Data Transforms calls *Printrix* "Personal Typesetting Software." I think that's a pretty fair description of what it is and what it does.

Instead of requiring a very expensive machine (or $300 a month), *Printrix* produces near-typeset-quality output on a variety of dot-matrix and laser printers. Like a great athlete, *Printrix* does what it does so well, and makes it look so easy, it's hard to appreciate. For example, take a very close look at the distance between the lines in the first three paragraphs of this review. If you're good at this kind of stuff, you'll notice that the line spacing increases in each of those paragraphs. *Printrix* allows you to adjust line spacing in increments of one dot. The exact size of a "dot" depends on what printer you are using, but on an ImageWriter II, it's nearly imperceptible.

Likewise, *Printrix* allows you to adjust the minimum amount of space between letters and between words. When paragraphs are fully justified, as the first three printed here were, *Printrix* will automatically "microjustify," or add space between words and letters, to make a line look even. On paragraphs that are ragged right, ragged left, or centered, the letter- and word-spacing is always exactly what you set it at. Here, for example, is *Printrix's* default letter- and word-spacing for the font I'm using. This is a little too much for my taste.

I guess it's obvious by now that, in addition to allowing you to adjust letter-, word-, and line-spacing in very small increments, *Printrix* also supports all the justification features we've come to expect typesetting machines to have.

The *Printrix* manual is well written and well organized, and was completely typeset using the IBM version of *Printrix* and an unidentified laser printer. It's a very good example of what a near-typeset-quality book looks like--which is quite good. The only thing the manual is missing has nothing to do with *Printrix*--it should have a chapter on what kinds of tools and equipment are available for pasting up typeset copy. Most art supply stores carry special materials that make

this step easy, but novice "personal typesetters" won't know this unless it's in the manual.

Since *Printrix* uses the "graphics mode" of whatever printer it is printing on, printing is a lot slower than using the same printer's "text mode". The limit here is not only the speed of the printer, but also the speed at which *Printrix* can make the necessary calculations. Like *MacWrite*, *Printrix* on an unaccelerated Apple can't push the ImageWriter as fast as the ImageWriter can print. It took about 15 minutes to print this review using a IIgs in "normal" mode, although much of that time was devoted to the font samples printed below (only one font is held in memory at a time, so the samples involve a lot of disk access as *Printrix* switches back and forth between fonts for each set of eight vertical dots it prints). From the top to the first sample took a little over 8 minutes (but only about 6-1/4 with the IIgs in "fast" mode).

The only other review I've seen of *Printrix* so far was in *Scarlett*, the newsletter of the Big Red Apple Club in Norfolk, Nebraska. That reviewer didn't like *Printrix*, but his dislike was based on the expectation that *Printrix* would be a **page layout** program. It's not and it doesn't claim to be (although it does have a feature for printing in two columns). What you see on your computer screen is your favorite word processor with lots of funny codes embedded in the file--the same thing professional typesetters see. What comes out of your printer has to be cut up and pasted together, just like the professionals do it.

One thing that *Printrix* can do that my professional typesetter can't is incorporate Apple II graphics into text. (He can't incorporate *any* graphics into text, much less *Apple II* stuff.) *Printrix* can use standard single-resolution graphics, *Fontrix* graphics, and *Print Shop* ART.graphics.

Printrix comes with a number of one-size fonts of limited usefulness. Its best fonts come in seven sizes each. Three of these are serif faces, for those of you who know what that means, and one is sans serif. Here are some samples of these four faces in their largest and smallest manifestations (and notice the nice tabbing):

Ask (or tell) Uncle DOS

It seems my elves have been running short of 8's and have been replacing them with whatever is handy. If I could borrow a couple of them from you, please, put one in the June 1987 issue, on page 3.37, in the sixth line of Ask Uncle DOS. The second character in that line should be an 8, not a 3. In other words, the correct address of the Basic.system input hook is 48690-91 ($BE32-33) and may it remain there forever. You'll find that this correction is recursive and requires you to correct the December 1986 issue as well.

Put the other 8 you're lending me in last month's issue, page 3.56. In the AppleWorks "Desktop sizer for Slinky-type cards" program, line 210, the second digit of FIX should be an 8, not a 4. In other words, the correct value for FIX in line 210 is 9829+4096. If you've had trouble getting the program to work with AppleWorks 2.0, this is why.

Avoiding control-I gobbling

The letter "Cards still gobbling ctrl-I" in your August issue (page 3.54) prompts me to tell you my solution for getting control codes past the interface card and into the printer. In a word, instead of PRINTing them, POKE them.

With my Grappler-Plus, a byte written to $C080 + $n0, where n is the slot containing the card, goes directly to the printer. For slot 1, this address is 49296. Thus, POKE 49296,C will send the ASCII value C directly to the printer, without even needing a preceding PR#1. The benefit of sending characters directly is that this method bypasses the card's firmware, thus avoiding the "gobbling" of control characters, as well as unneeded linefeeds that mess up the location of top-of-form.

Keith J Bernstein
Bronx, N.Y.

There are two other benefits to your technique—you avoid having Applesoft automatically set the high bit of the character you want to send, which always happens if you use PRINT. And you avoid having DOS interpret a control-D as the beginning of a command.

However, you have also created two significant problems. The major one is that you aren't testing to see if the Grappler is ready for another character. If you run your program on an accelerated Apple, you'll probably find it won't work anymore. This is because you will be sending characters to the Grappler faster than the Grappler can send them to the printer, and some characters will get skipped. And, obviously, if you run your program on a computer that has any other type of interface card, it won't work anymore because the hardware location you're using isn't standardized.

OK, I admit it, when I write programs for myself, I sometimes go straight to the hardware, just as you did. But I do it knowing full well that the program is locked to one computer. I do it knowing that if I change the interface card or the computer, I'll probably have to redo the program.

In most situations, having to redo the program every time you move to a different computer or change an interface card is crazy, unacceptable, and a waste of time. The more widely distributed the program is, the more time it wastes. While I admit the command codes for interface cards aren't as standardized as they should be, my answer to last month's "gobbling" letter for getting control-I through the interface card will work on all cards I've ever heard of. To solve the more complicated problem of sending ***any*** *control code without gobbling (not just control-I), see "Control-D(efeated)" in December 1986, pages 2.84-86. Take particular note of the program on page 2.86, which is a more generalized solution that works on any Apple and with most (but not all) interface cards.*

The issue of interface-card command-code incompatibilities is one I hope to discuss at APDA's standards conference at AppleFest (Sunday, 12-2). If any of you have a favorite incompatibility you'd like to see discussed, let me know.

RAMdisk detective revisited

In your August issue, a letter asked how to tell if a volume is a RAMdisk ("RAMdisk Detective, page 3.52). You mentioned the difficulty of doing so if the device isn't chained to a Smartport. However, I have used the following method, which is based on Apple's ProDOS Technical Note #16.

First, issue a GET_FILE_INFO call (command $C4) to the MLI, using the name of the volume you want to check. This will return, among other things, the size of the volume in blocks, which may be useful later. More importantly, this will place the unit number (slot and drive) of the device into the DEVNUM byte on the ProDOS global page—$BF30.

The unit number is in the form DSSS0000, where D is the drive and SSS is the slot. Convert it to the form 000DSSS0 and use it as an index into the DEVADR table at $BF10-$BF2F. If the driver address is $FFxx (where xx is anything), then it is a safe assumption that the device is a ProDOS RAMdisk. Usually xx will be zero, but I believe it is a $05 for CheckMate's /MRAM.

This method works only for non-Smartport RAMdisks, so it will still be necessary to check for Smartport RAMdisks.

Alan Silver
Yorba Linda, Calif.

Good idea! Here's what the assembly language code would look like:

```
first, check for a Smartport (see Jan 87, page 2.90)

if not a Smartport, do GET_FILE_INFO, then

LDA $BF30       DEVNUM
CLC
ROR             convert DSSS0000
ROR              to 000DSSS0
ROR
TAX             use result as an index
LDA $BF11,X     get high byte from DEVADR table
CMP #$FF        compare to $FF
BEQ it's.a.RAMdisk
```

No fancy tricks, please

Using PEEK(-1) to identify a IIgs, as suggested by Bill Basham (July 1987, page 3.48) is a bad idea. Apple has provided a reasonable way to identify a IIgs: SEC, JSR $FE1F, BCC it's a IIgs. This is only six bytes of machine code.

HELVET

TIMES

CASLON

PRESS

Helvet

Times

Caslon

Press

You can also create your own fonts using Data Transform's program *Fontrix*. There's no such thing as too many fonts, however, so I hope Data Transforms comes up with some utility disks that have more standard text typefaces in a wide variety of sizes. Up till now, font disks have put too much emphasis on providing fairly useless fonts such as Old English and not nearly enough emphasis on providing a variety of sizes in a single typeface of the kind you'd actually use to typeset a book or newsletter.

Back in January 1985 (page 3), I ended a review of *Fontrix* with, "The day will come when software will be available that will turn an Apple II and upcoming laser printers into typesetting machines. However, we aren't there yet." Well, it looks like we're there now--and with dot-matrix printers yet. I think *Printrix* makes a good compromise between quality and price for any small newsletter or book publisher.

Printrix **by Data Transforms, Inc. (616 Washington St., Denver, CO 80203 303-832-2502). ProDOS. Unprotected. "As is" warranty. $65.**

While I agree that the value in PEEK(-1) isn't likely to change on a IIgs, future IIe or IIc models could easily have the same value there. This is because the ROM that seems to be at $C074 on a IIgs isn't really there at all, but *appears* there because of some custom chips Apple built the IIgs around. Future IIe or IIc models could use the same custom chips.

David A. Lyons
North Liberty, Iowa

You're right. I probably shouldn't have published that particular tidbit from Basham. But he's always full of such good stuff. For example...

IIgs programming subtleties

Here are a few observations to add to your discussion of the IIgs memory manager ("IIgs memory manager revealed," July 1987, pages 3.47-3.48).

If you ask for a block of memory that is bigger than the largest available block, the IIgs memory manager will try to get it for you by moving or purging blocks that belong to another program.

For example, my program *Diversi-Cache* uses a cache buffer adjustable in size from 0 to 800K, which can be purged. Unfortunately, few new IIgs programs will purge it properly. First of all, the automatic purging part of the memory manager isn't even present in the first release of the IIgs ROM. You have to boot ProDOS 16, version 1.2, to get purging to work. Versions 1.0 and 1.1 have bug-filled purging routines, so be sure to get version 1.2. The new ROM Apple is giving away in its IIgs upgrade program has the new purging routines included.

Another problem with purging is that there's no way to tell in advance how much memory is purgable. The FREEMEM and MAXBLOCK functions of the memory manager do not include purgable memory. The only way to trigger purging is to simply try to allocate a large block with NEWHANDLE. Don't assume you don't have enough memory by looking at FREEMEM. Instead, try to allocate the memory and check the error status after each allocation call.

I've also had the problem you described where the 3.5 drive goes off-line until a power-off reboot ("IIgs slots, drives missing, August 1987, page 3.54). I think I'm close to tracking it down. The problem only occurs when you daisy-chain a 5.25 drive onto the back of an Apple 3.5 drive. The problem is caused by poor current sinking capacity in a new 40-pin custom chip in the Apple 3.5 drive. This chip can barely drive the circuits in the 5.25 drive. When it overloads, a bit flips and makes the daisy-chained drive look like a 3.5 instead of 5.25. This causes both drives to come on at the same time, and the 5.25 drive overpowers the 3.5 drive. Pressing reset turns off both drives, but the 3.5 drive won't work again until you power down.

If you've got the nerve, you can unplug the 5.25 drive with the power on, and the 3.5 drive will work again. I've gotten away with it three times, but I don't recommend doing it, since chances are high you'll toast a chip somewhere.

The problem usually occurs with non-Apple 5.25 drives that have 1K pull-up resistors. New Apple 5.25 drives use bigger resistors. If you have a non-Apple drive, the best solution is to use an interface card in slot 6 instead of daisy chaining. Another solution is to put a UniDisk 3.5 between your Apple 3.5 and your 5.25 drive.

Finally, let's talk about reading the keyboard from machine language on the IIgs. It's important to remember that the IIgs has a type-ahead buffer. You should only clear the keyboard strobe ($C010) *after* reading the keyboard and the status of the open-apple and option (solid-apple) keys. In other words, *never touch $C010 except after reading a key.* This is critical if you want your programs to work with macro programs like *Diversi-Key.* If a program throws keystrokes away, the macros don't work. Here is an example of a correct way to read the keyboard:

```
KEYIN   LDA $C000
        BPL KEYIN
        LDY $C025  ;bit 7 is OA, 6 is option
        STA $C010  ;be sure this is LAST
        RTS
```

Location $C025 on the IIgs contains the status of the open-apple and option keys. If your program must also work on a IIe or IIc, you can read locations $C061 and $C062 instead (but *before* the STA $C010).

Bill Basham
Farmington, Mich.

I was ready to confirm much of your "missing 3.5" theory. When the problem happened to me I had one Apple 3.5 and one 5.25 daisy-chained together. Several months ago I added a UniDisk 3.5 between those two and hadn't experienced the problem again,... until today.

This time it was the UniDisk 3.5 that conked out. Yes, the 5.25's motor came on for no reason just before the UniDisk went to sleep. In addition, the 5.25 drive on my IIgs has always been a new "platinum" Apple model. Let's keep working on this one.

In retrospect, my July memory manager discussion didn't make it clear that I was talking about using the memory manager from a ProDOS 8 cold boot. While this is the easiest place for beginners to start when working with the memory manager (and aren't we ***all*** *beginners), it doesn't get us very far because of the problem with tool bugs that Basham mentions in his letter. ProDOS 16 is sophisticated enough to load "tool.setup" files that fix bugs in the ROM-based toolbox. But unless you first boot ProDOS 16, then move to ProDOS 8, you don't get these fixes. So, for today's lesson, let's look at what happens when you use the memory manager after first booting ProDOS 16 and then executing ProDOS 8.*

Now an MMStartUp call works just fine. As explained in July, a successful MMStartUp call tells you what your UserID is. In this case, you'll usually find that your UserID is $3001. Scan the chart of possible IDs from July and you'll notice that this is a "ProDOS" ID.

Under ProDOS 16, a program called the System Loader is responsible for starting up the programs you choose to execute. The System Loader figures out how much memory the program you have chosen needs, gets an ID for it from the ID manager, and gets memory for it from the memory manager. Then it has ProDOS load the needed files and it starts them running.

If the program you choose is a ProDOS 8 application, the System Loader requests a "ProDOS" ID from the ID manager. Using this ID, it then asks the memory manager for all of the memory in banks 0 and 1. By the time a ProDOS 8 application (that is started up after a ProDOS 16 boot) gets control, it already has an ID and banks 0 and 1 have already been allocated to it.

If the program you choose is a ProDOS 16 application, on the other hand, the System Loader gets an "application" ID. It then requests two blocks of memory. One will be for the program itself and could be in any bank. The second will be a $400-byte block in bank 0 that is to be used for that application's

private stack ($300 bytes) and direct page ($100 bytes).

At any rate, the first thing your program is supposed to do is start up the tool locator with ***TLStartUp*** *(LDX #$0210). This tool call doesn't use the stack and has no errors. That's because it doesn't really do anything, but it might someday, so call it. Next start up the memory manager. As mentioned, this is how you find out what UserID you've been assigned.*

Next you're supposed to use something called ***LoadTools*** *(LDX #$0E01). You have to specify what tools your program will use and the minimum acceptable version numbers you'll take. Old LoadTools hauls all the tools that aren't in ROM into memory from disk and makes sure the version numbers are ok. (He's the one primarily responsible for the fact that it seems to take 40 days and 40 nights to get a ProDOS 16 program up and running. One millennium, please.) Next you start up all these tools.*

Now, suppose that the program we are writing is a word processor. We need to acquire some more memory for data files. When we do this (as described in July), we should set the "aux ID field" to something other than zero. Here's how:

```
assumes 16-bit mode

    LDA UserID
    ORA #0100
    STA DataID   an ID for data blocks
    AND #F0FF
    ORA #0200
    STA HelpID   an ID for help blocks
```

Now, in addition to our main UserID, we have used the aux ID field to create an ID for data memory and an ID for memory that will be used to store help files. Again using the techniques described in July, we next ask the memory manager for three blocks of data memory, because our "user" has just told us to load three word processing files onto our desktop. We tell the memory manager to assign these files a purge level of 1 (0 is not purgable, 1 is least purgable; I assume we will always "lock" any data file that is changed but not yet saved back to disk to keep it from being purged). Next, the user tells us to display a help screen. We use our HelpID to request a block of memory for our help file. We give this block a purge level of 2.

As the session progresses, our user might want to move one of the word processing files to our clipboard. If so, we use the DataID to request memory for our clipboard. Now suppose the memory manager discovers it is out of memory and looks around for something to purge—it might find the help file and reclaim that memory.

Later still in the session, our user once again asks to look at the help file. At this point we don't know whether it's been purged or not. So we make a ***RestoreHandle*** *call (LDX #$0B02). This tool requires that we push the "handle" (see July for explanation) of our help screen block on the stack before making the call. It returns nothing but error codes. If we get error $0201, there's not enough memory left to load the help file (maybe the clipboard is still full of stuff). If we get error $0203, this block was never purged, so our help file is still in memory and we can immediately use it. If we get no error, it means our help file was purged, but its memory has now been restored and we can load another copy of it in from disk (the clipboard was used, but is now empty).*

Finally, our user decides to quit. After saving the data files, our program must relinquish the memory it got from the memory manager. To do this we use the memory manager's ***DisposeAll*** *(LDX #$1102) tool, as described in July. However, unless you're working under bare ProDOS 8, you should never use DisposeAll with your main UserID. This results in deallocation of the memory in which you reside and can have drastic consequences. Instead, make separate DisposeAll calls using each auxiliary ID.*

Finally, call ***MMShutDown*** *(LDX #$0302, push UserID on stack, nothing returned, no errors) and* ***TLShutDown*** *(LDX #$0301, no stack usage, no errors). Neither one does anything now, but who knows what the future will hold? Then issue a ProDOS QUIT call. This eventually will take you back to ProDOS 16, whose responsibility it is to do a final memory manager DisposeAll call using your main ID (****after*** *the MMShutDown call with your ID, oddly enough), and to see that the miscellaneous tool DeleteID gets a copy of your obituary.*

(Thanks to Apple's IIgs Technical Note #17 and a sneak peek at Roger Wagner's upcoming book ***Apple IIgs Machine Language for Beginners****, which looks terrific, for much of this information.)*

Avoid special memory

Speaking of the memory manager... if an application doesn't specifically prohibit NewHandle calls from allocating space in "special" memory (the managed parts of banks 0 and 1 and the display pages in banks $E0 and $E1), elements of the program can get put all over the double hi-res pages, including code, heart beat queue elements (816/Paint), and event queue elements (DeluxePaint II). What a mess when you are trying to use the double hi-res screen from a desk accessory.

If you use the system disk's "Launcher," it loads the RAM-based tools. These never leave memory again unless the system loader needs the memory for loading another program file. The files are marked loader-purgable (purge level 3), but can't be purged by the memory manager for a ProDOS 8 application because the loader isn't active. Another Catch 22.

Ken Kashmarek
Eldridge, Iowa

SoftSwitch and copy protection

Regarding your review of *SoftSwitch* from Roger Wagner Publishing in the July issue (pages 41-42), I must agree completely about the protection scheme. One thing you don't mention is that the scheme is even more annoying than it appears. When you first configure a disk with this program it writes a signature byte in the battery backed-up RAM at byte $FB on your IIgs (in RAM that Apple has designated as reserved). If you attempt to boot a disk containing *SoftSwitch*, the program checks for this signature byte in the RAM and won't install unless the low bits of the byte there are "10", or else it requires the original (protected) disk to be online somewhere.

This means that if you replace or disconnect your battery or have your motherboard replaced during repair or upgrading, you lose the signature byte and *SoftSwitch* henceforth works only with the original disk online. This is a terrible feature as your original disk won't work to install the signature byte after the first use, either.

It is possible to make a minor change to the file SYSTEM/SYSTEM.SETUP/TOOL.SETUP.2 to fix this. By BLOADing this file at $2000 and typing \48 A2 03 0C 22 00 00 E1 68\<2000.8800P you can locate the code that reads the signature byte from the battery RAM. I found it at $2BA9. If you replace this code with 48 A2 03 0C 68 68 A9 FE 00 and BSAVE the file back to the disk, then *SoftSwitch* no longer requires the signature byte in the battery-backed RAM. You can even copy the file to other disks and have them install *SoftSwitch* in memory without going through the install procedure.

Peter Stubbs
North Lambton, NSW, Australia

As I said last month, in general my attitude toward copy protection is to simply ignore programs that use it. We can live without them. Unfortunately, I didn't realize the extent to which ***SoftSwitch*** *was protected when I wrote about it, or I wouldn't have. Since I could make copies of the* ***SoftSwitch*** *system disk that worked just fine, I thought that having an uncopyable install disk was merely an annoyance. I did suspect that the program had some sort of one-program/one-computer protection scheme, and asked Wagner about it, but put the question wrong. (I said that it appeared* ***SoftSwitch*** *was checking some sort of serial number embedded in the IIgs and asked whether Apple's engineers had put one in there somewhere. Wagner assured me they had not.)*

In general, as part of my policy of ignoring copy-protected software, I'm not interested in publishing information on how to defeat copy protection schemes. I decided to publish the last paragraph of this letter primarily because I feel an obligation to those of you who may have purchased ***SoftSwitch*** *on the basis of my July article. It also demonstrates the ultimate futility of copy protection—this program has been on the market for less than three months and its protection scheme has already been defeated. Based on carbon copies of Wagner's mail that people have been sending me, I'm pretty sure the copy protection on* ***SoftSwitch*** *will end up costing a lot more in management and customer service time (and in lost sales) than it will ever earn by selling more products.*

That said, please send all further letters with tips about defeating copy protection schemes directly to my friends at ***Computist****, P.O. Box 110846-T, Tacoma, WA 98411. I'm not interested.*

TK!tinkerbell

I called Universal Technical Systems about *TK!Solver* after seeing your mention of it in the July issue (page 3.43). Their update is for IBM, not Apple.

Is there any type of programming known as "subliminal" that uses the brain through the eyes and the power of suggestion to help students learn?

C. K. Bevan
Atlanta, GA

You're right about ***TK!Solver****. As soon as that issue hit the streets someone from Universal who actually knew something about the Apple version called me and said that the only difference between today's product and the original is the price. It's still DOS 3.3-based and still copy-protected. The person at Universal I talked to before writing July's article was full of gas.*

Back when I was a graduate student in journalism school, the subject of subliminal persuasion came up in an advertising seminar. Subliminal means "below the threshold of conscious perception." (The idea, for example, is to get Charmin's Mr. Whipple to give a sales pitch over a supermarket's loud speakers at a sound level no one can quite hear—which would be the only condition under which I would enter the building). The class took a look at the research that had been done and determined that this subliminal stuff was invented by the mother of the first guy I talked to at Universal. Subliminal

learning, advertising, and mind control are cousins of the tooth fairy. (Which is not to say that someone somewhere hasn't written a program that "uses" it.)

Changing the blink speed

Your reply concerning the AppleWorks cursor ("Blink and its gone," May 1987, page 3.32) was good and helpful, but only so far as it went. I have spent more time than I like trying to find the hiding little insert cursor when I am editing text. Hence, I was both happy to learn I am not alone and to see your suggestions for improving the situation. I have made the patch you described and now have a blinking checkerboard as my insert cursor.

But the real culprit in this situation is not the shape of the cursor. It is the speed of blinking. The replace cursor blinks with a longer time on and shorter time off. The insert cursor blinks with shorter time on and longer time off. I think the reason it's so hard to find is that it's so rarely on. Where in AppleWorks can we go to control the blinking speed?

Robert R. Eddy
Concord, N.H.

For AppleWorks 2.0:

```
BLOAD APLWORKS.SYSTEM,A$2000,TSYS
CALL-151

2DA9:   LDY #$1C        insert cursor on
        JSR $1F25

2DB3:   LDY #$6C        insert cursor off
        JSR $1F25

2DCB:   LDY #$6C        replace cursor on
        JSR $1F25

2DD5:   LDY #$1C        replace cursor off
        JSR $1F25
```

Modify these to your heart's (and eyes') content.

Are we fans of fans?

Do I need a system fan on my IIe? When I increased my RAMWorks II memory to one meg and added a clock card along with Central Point Software's Universal Disk Controller, I also installed a heavy duty power supply from Applied Engineering. I will probably add a TransWarp card in the future. In your response to the letter, "The AppleWorks machine" (July 1987, page 3.44), you mentioned increased memory, accelerators, and power supplies, but you say nothing of cooling fans. Are system fans simply a $60.00 gimmick?

James H. Weber
Middlebury, Conn.

Apple II hardware guru Jim Sather takes time out to explain the importance of cooling in his book ***Understanding the Apple IIe*** *(pages 10-3 and 10-4). He includes a sample chart of what happens to the temperature inside an Apple II with and without a fan. According to his tests, the temperature inside an Apple can rise as much as 25 degrees F. above ambient temperature, and a fan can reduce the temperature by about 15 degrees. There's no question that many electronic components age faster at higher temperatures, so a fan may improve the longevity of your computer and its peripheral cards. At extreme temperatures some components simply stop working.*

Don Justesen, my local guru on Apple power supply questions, tells me that recent tests he's done indicate that heat can have a significant effect on the quality of the voltages provided by power supplies. Thus it's possible that the power supply problems discussed in "Releasing the power" (July 1987, page 3.45) are caused as much by the heat that power-hungry cards produce as by the power that they consume. A fan, of course, would help take heat out of the equation.

The downside of fans is that they are a major source of noise pollution. One of the likable aspects of Steve Jobs was that he hated fans. That's why neither the Apple II nor the original Macintosh had one built in. Unfortunately, with Jobs gone, Apple has accepted fans with a vengeance.

Apple now sells an add-on fan for the IIgs that it recommends to users who have more than three cards installed in that machine. This sucker sells for $49, which is about 20 times what it costs Apple to make. The fat margins apparently went into research and development—some marketing wizard wanted to make sure you knew it was in there after spending that much money. The funds were well spent—even after burying it deep inside the case, the engineers were able to make the IIgs fan sound like a small vacuum cleaner. I recently took a couple of cards out of my IIgs for use elsewhere and took the opportunity to disconnect the fan at the same time.

Kensington, the System Saver people, have just announced a new unit designed specifically for the IIgs. It sits on top of the IIgs case and includes a fan, four power outlets, two power switches, surge suppression, and noise filtering. It retails for $100. Its value will be directly proportional to the amount of noise it makes, which is unknown, by us at any rate, at this time. It's hard to imagine it being louder than Apple's fan, however.

Thinking about databases

In regard to the speed of the AppleWorks database Find command ("Slow find, fast sort," April 1987, page 3.20), I used a 1,500 record database (110K on disk) to compare versions 1.3 and 2.0. The former took 13 seconds to Find a single record; the latter, 22 seconds. Both times are slow, but Find is an unusually broad search. Version 2.0 has been optimized for Select, which can almost always accomplish the same task. Version 2.0 took a respectable four seconds; version 1.3, seven seconds. For continuous online searching of a database, I use 2.0 and write a macro for Select.

Given the limits of the processor, optimizing Select was a reasonable design choice, but that choice should have been documented because people tend to use the simpler Find. Most relational systems optimize indexed searching, even at the expense of an unindexed search. Few offer the equivalent of the AppleWorks Find (unless, of course, you specify the same condition field by field).

In regard to "Making AppleWorks relational" (June 1987, pages 3.33-3.35), I agree that the important question is how far AppleWorks can go. Its database is fast enough because it keeps everything in RAM and requires no indexing, but serious relational applications need a fast hard disk and carefully designed indexing.

The practical advantages of using related files are several:

1. It saves data entry time.
2. It minimizes the problem of inconsistent entry of repeated data.
3. It saves the cost of RAM and disk memory.
4. It almost always allows programs to execute faster.

Inconsistent data is the most significant problem, so if AppleWorks can be only partly relational, this is where to focus attention.

The database example you used is more complex than what is needed to explain basic concepts. The simplest relation needs three files, in this case patients, medications, and transactions. However, cases with six or more files are common, and that is where operations bog down.

Transaction files generally have the most records and they must be designed to conserve memory. Such files typically contain only unique identifiers. You then design readable query screens and reports by opening multiple files and retrieving data from each. Most applications require relations in reports, memory operations, and screen displays. You can already buy AppleWorks add-ons for reporting from two files.

When it is cumbersome to relate files in screen displays, you need to make the unique transaction identifiers understandable to the searcher. For example, you can use mnemonic three-letter codes for the drug name. Sometimes the choices are difficult; you might reluctantly decide to use patient last names rather than social security numbers, adding numbers to repeated names, such as Smith1 and Smith2. (Even social security numbers are not always unique because some are borrowed or fabricated. This is a problem in health and social service database applications, particularly when the fabricated duplicate enters the system first.)

The most practical (and intelligible) relational application for AppleWorks might well be using two files to double the number of fields per record. Alan Simpson describes the Big File technique in chapter 9 of Simpson's *dBASE III Library*.

VP-Info is my choice for the MS-DOS relational system most likely to be useful on an Apple II and I read somewhere that Paperback Software is developing a II version. *VP-Info* has a built-in compiler, supports a math coprocessor, and optimizes the operations that eat time (such as Index and Append). It currently lists for $99 and can run on anything down to a 256K PC with two floppy-disk drives. Although *VP-Info* has no menu-driven front end for non-programmers, one could write AppleWorks-like applications (I have written *FrameWork*-like applications).

I like 10 MHz AT-compatible systems for large analytical database applications and have assembled a dozen in recent months. You have to order parts from nine different sources and do a lot of testing to get a completely dependable machine, but the cost is low (less than $2200 with a coprocessor and 44MB hard disk) These systems can run *VP-Info* applications with Indexed Find times of 0.1 second for a 30,000 record, 16MB file.

On other fronts, my limited testing indicates that AppleWorks spreadsheet recalculation takes about 5 per cent longer in version 2.0, as one might expect with the added functions. For reasons that I cannot understand, the Word Processor Find and Replace commands take about 10 per cent longer.

Notice that the new *Using AppleWorks* manual carefully explains the Silentype method for printing formatted ASCII files (page 249), even though the built-in ASCII option in version 2.0 now does that (and only that). Maybe someone can patch it back to what it used to be. Apple informed me in March that version 2.0 was then the only upgrade officially available. Maybe Apple should start a "downgrade"

policy, because version 1.3 remains a better program for some applications. Roger Coats must know that better than anyone, because he still advertises version 1.3 (*see September 1986, page 2.58*).

Robert Ericson
North Providence, R.I.

IIgs vs the clones

As I am on the verge of needing a new computer I was very interested in the exchange of views under "Love's labor's lost?" in your June issue (page 3.40).

I bought my Apple II in 1980, turned it into a II-Plus sometime later. With cards and a full ASCII keyboard I have enhanced it about as far as seems economically feasible.

Unlike Mr. Boering, I do not feel the choice is between the IIgs and the Mac II, but between the IIgs and one of the many IBM PC-AT clones. The Mac is a specialty machine in another price range.

Favoring the MS-DOS machines is the quality of the available software. For example, this letter is being written using WordStar. When I bought it several years ago, it was the standard in word processors. With the addition of a CP/M card, it could be run on the Apple. It was a great product, but now as I peruse *Infoworld* and *Byte*, I see updated versions of Wordstar, much improved spelling checkers, thesauruses, and other varied products that are available only for the MS-DOS machines. It appears that most current software is being written primarily for these machines.

Additionally, there is Apple Inc. itself. I still feel that the company would like to kill the II line, or if they can't gracefully do that, relegate it to a back room somewhere with no support or upgrading. From the reviews of the IIgs I have read, the machine itself is indicative of this and not what it should have been. It appears to be slow. According to the review in the April *Byte*, my accelerator card will do basic calculations faster. The graphics resolution is low by current standards. The 65C816 can address 16 megabytes of memory but for some reason the IIgs is expandable only to eight. Only 256K on the motherboard is close to inadequate.

In the normal course of events these things would be upgraded and improved. Unfortunately, with what I perceive to be Apple's attitude toward the II line, I believe we will have all gone to our great reward before that happens. Sculley and crew don't want to be bothered with the II line and they certainly don't want it competing with their beloved Mac.

The IIgs with two drives, RGB monitor, fan, and Applied Engineering's RAM Plus board comes to about $2950. This puts it in competition with some pretty fair MS-DOS machines.

Favoring the IIgs is that everything I have is basically compatible with the new machine. If I went to another system I would have to replace everything I need immediately or keep the old machine around until it is gradually phased out. Preserving and transferring required data files would be a major problem. Of course, if I have to do this eventually anyway, it would certainly be better to do it now than make any further investment of time and money in the Apple line.

I know of nothing to do but wait as long as I reasonably can and see what transpires. Until I do I will try to avoid any major investment in computer related products.

William G. Ingersoll, Jr.
Shalimar, Fla.

We've thought for some time around here that cheap MS-DOS clones are stiff competition for the II family. As always, the key is to find the software you want and buy a machine it will run on. MS-DOS software tends to be stronger than what's available for the Apple II in institutional applications. The Apple II is stronger in elementary and secondary education and in home applications. Small businesses have the most difficult choice—fast, friendly, integrated AppleWorks on the one hand, or more powerful, less friendly, non-integrated packages on the other.

I think your description of Apple was more accurate under the reign of Steve Jobs than it is now. Apple appears to me to be committed to two lines of computers united by a single line of peripherals. One of those lines has to be downward compatible with the original Apple II, which puts some limits on it that the other, more expensive line, can disregard. Within those limits the IIgs is a spectacular machine. There is much more that is new about it than any previous update to the original Apple II. If the Apple of Sculley wasn't committed to the II family, the IIc would have been the end of the line. (On the other hand, if I thought your concerns were totally unwarranted, I wouldn't have printed your letter. There are days when it seems everyone at Apple uses a Macintosh.)

At any rate, Applied Engineering's new PC Transporter card will work in your II-Plus. It requires an "IBM-style keyboard," but you indicate you've already upgraded the keyboard on your II-Plus. So now you have three choices, not two—a IIgs, a clone, or add a clone to the computer you already have. That old II of yours may never die.

Go, Logo, Go

I do a lot of work in Logo and was looking forward to using it with the greatly expanded memory on my IIgs. However, I still have the same amount of nodes to work with as before. Is there a way to access more of the IIgs memory? Help, I want more nodes!

Douglas Lane
Big Bear City, Calif.

Can you tell me how to print out procedures and graphics using *Logo II*, an Apple IIe, and an Epson MX80 with a Grappler? The manual only talks about the ImageWriter.

Barbara M. Kirch
Linwood, N.J.

Based on the number of letters we've gotten asking questions similar to the first one here, there seems to be a great deal of misunderstanding in the kingdom about what's possible when you upgrade to a IIgs. Software that you bring to the IIgs from an earlier Apple will do one of two things—exactly what it did before or less. Never more. To use the expanded memory capacity of the IIgs, you'll have to get a new version of Logo that has been specifically written or modified for the IIgs. We don't know of any, but this isn't an area of great expertise around here.

We discussed how to print out ***Logo II*** *procedures in September 1986, page 2.63. Use DRIBBLE 1 POALL NODRIBBLE.*

In that issue I mentioned that some Logo users have had trouble with printer line-lengths when listing programs, and that these troubles could usually be conquered by changing the interface-card line-length setting with a control-I command. I also said, "you can't send commands to some interface cards, such as the Grappler, because ***Logo II*** *prints CHAR 9 (control-I) with the high bit clear and the Grappler expects the high bit to be set."*

While all that is true, Dennis recently figured out that it's not really the reason so many Logo users have trouble with the Grappler. ***Logo II*** *talks to printer cards using what is known as the "Pascal 1.1 interface." Unlike Apple's printer cards, the Grappler expects the command character to be control-Y, not control-I, when the Pascal interface is used. Those of you who can't get Grappler control-codes to work with one of your programs should try changing the command character you're sending to control-Y. Much of today's assembly language software uses the Pascal interface rather than the older Basic interface, so this could explain why control-I codes sometimes don't seem to work on the Grappler.*

With that problem solved, the easy way to print graphics with your equipment is to tell the Grappler to do it. Wake it up with a control-Y (CHAR 25 in Logo), send a "G" to tell it to print a graphic, and add your choice of D(ouble size), E(mphasized), I(nverted), M(ixed mode), or R(otated) (see the Grappler manual for more on these codes). For example, to print a copy of the screen, including the four lines of text at the bottom, double size, emphasized, and inverted, use TYPE CHAR 25 PRINT "GMDEI. Try a procedure like this one:

```
TO PRPIC
MAKE "SLOT 1
OPEN :SLOT SETWRITE :SLOT
TYPE CHAR 25 PRINT "GMDEI
CLOSE :SLOT SETWRITE []
END
```

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.

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

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