Open-Apple™

December 1985
Vol. 1, No. 11

ISSN 0885-4017
newstand price: $2.00
per page photocopy charge: $0.25

Releasing the power to everyone.

:FLASH:

The UniDisk 3.5 upgrade enhances the Apple IIc. The upgrade, which is free to purchasers of the UniDisk, involves a motherboard change. The new board has 32K of ROM where before there was only 16. In addition to the machine language code needed by the UniDisk, the ROM includes more commands for the IIc serial ports (including the XON/XOFF protocol that was previously missing, as discussed here last month on page 84). After an upgrade, the ports are closely compatible with Apple's Super Serial Card. The new ROM also includes a miniassembler, like the enhanced IIe.

Apparently it was technically feasible to add the ROM without changing motherboards, but Apple decided to combine the UniDisk 3.5 upgrade with the ongoing motherboard upgrade for people having serial-port timing problems. The timing problem occurs on early IIcs (see our June issue, page 47). New IIcs have had the timing problem fixed for some time. But new IIcs won't include the UniDisk upgrade until after the first of the year, according to Apple spokesperson Linda Merrill.

The unused cassette switch at $C028 toggles the two 16K banks of ROM on and off (see May, page 39). No switch is available to read the current bank status, however. The traditional ID bytes for the new ROM (see May, page 40) are the same as for the original IIc. You can tell the difference by looking at $FBB3. An $FF there indicates an original IIc, a zero indicates a 32K ROM.

There's lots of magic in the UniDisk connection, too. It's really more than that. Apple's documenation calls it CBUS—it's a non-standard high-speed serial connection for "intelligent" peripherals. Peripherals become intelligent when they have a microprocessor and software in them. Subscriber Tom Vier has discovered that inside each UniDisk 3.5 there's a 6502 microprocessor, ROM, RAM, and one of Apple's IWM chips. The CBUS interface appears to allow up to 127 devices to be connected to an Apple II in a daisy-chain—including character-oriented devices such as printers and modems. Isn't that interesting?

Apple lowered II-family prices about 20 per cent on October 1. The suggested retail price of the IIe is now $945. The price includes a 64K extended 80-column card. The IIc is available in a package with a monochrome monitor for $995 or with a color monitor for $1249. Look for Christmas-time shortages of both machines. Apple says its retail sales have been increasing since July and orders from dealers for the holiday selling season have been brisk. According to a survey conducted by Infocorp, the Apple IIe was the best selling computer in U.S. retail computer stores in September. (IBM's hottest machine came in third.)

The new Franklin computers are pretty exciting, too. For the price of a IIe you can get a 128K Franklin with two disk drives, a numeric keypad, a parallel card, mouse characters, and a fan. Franklin claims its new operating system is "functionally compatible" with DOS 3.3 and ProDOS and "also has several additional features not found in those two." Apple agrees that Franklin DOS 2 doesn't violate any Apple copyrights.

The machine even looks nice. The bad news? Only two slots. However, "a bus connector mounted on the side of each ACE 2000 series computer... allows for the addition of a four-slot expansion chassis or other peripherals that may be developed by outside manufacturers." Franklin is also quite proud of its keyboard, which is totally Apple-incompatible. I would be inclined to buy one of these things if the keyboard was like my others. The keyboard *is* detachable—maybe someone will quick come out with a replacement that has the cursor keys where they finally seem like they're supposed to be.

Applied Engineering continues to push back the limits of AppleWorks. The AppleWorks expansion software being shipped with Applied Engineering's new RamWorks II card allows 36 files on the desktop, 16,300 records in a data base file and 16,300 lines in a word processor file. A printer buffer (which allows you to continue to use AppleWorks while large documents are being printed) and the ability to convert 3 megabytes of RAM into a 2,205K desktop have been built in as well.

RamWorks II has room for 1 megabyte's worth of 256K chips. It can be further expanded by means of either a 512K or a 2 megabyte piggy-back card (these cards also work with the original RamWorks). Owners of the original can upgrade to RamWorks II for $120 (call first to get a return authorization; send in your card and a check; the RAM chips on your card will be removed and inserted on your new RamWorks II for you—Applied Engineering, P.O. Box 798, Carrollton, Texas 75006 214-241-6060.).

Apple sells its 64K 80-column card for $139; the equivalent RamWorks II (which you can expand to 1 megabyte with your own chips) is $179. A 256K RamWorks is $249; 1 meg $519; 3 meg $1699. An optional add-on RGB card is $129. Bob Ryan of ***inCider*** did a nice comparison of the RamWorks II, Checkmate Technology's MultiRam IIe, and Legend Industries E' card in their December issue, page 18.

Meanwhile, Apple has priced its own Memory Expansion Card at $299 for the 256K version and is shipping it now. Apple will sell you additional memory for the card for $69 per 256K, installed. This makes the 1

"NOT ONLY DID WE GET YOU AN APPLE WITH A MOUSE LIKE YOU ASKED, WE ALSO GOT YOU A BANANA WITH A LIZARD."

megabyte version $506 (or cheaper if you buy and install your own chips)—an unusually competitive price for Apple.

As mentioned here in October (page 73), Apple's card uses an addressing scheme that is completely different from any other card on the market. Applied Engineering's RamWorks and similar cards are organized as a series of additional 64K auxiliary memory banks. Apple's card, on the other hand, is organized as a one-byte peephole.

You read from or write to Apple's card by peeking or poking at byte 49283 + (SLOT*16). (In hex that's C083 + S0, where S is the slot number). You can transfer a sequential group of bytes between the card and memory faster than you can transfer the same group from one part of memory to another. This is because hardware on the card automatically increments to the next byte. Your program just keeps peeking or poking at the same peephole.

A disadvantage with this scheme, however, is that it's impossible to run a program while it's on the card. Programs must be moved into main memory first. This isn't a major disadvantage, however, since it's *almost* impossible to run a program stored in multiple auxiliary memory banks, too. Very little software actually uses the additional memory on cards like RamWorks to execute code. Most such software simply uses the additional banks as memory storage areas.

As simply a storage device, Apple's card is very clever. You tell the card what byte you want to appear in the peephole by placing that byte's address in 49280-49282 (+ SLOT*16), low-byte first. It uses any standard slot except slot 3, which means it works with a II-Plus as well as a IIe. It won't interfere with interrupts as auxiliary-memory RAM cards sometimes do. Nice card. Too bad the auxiliary-memory cards have already set a standard in this area.

Bus School

Mix and match your 5¼" drives

by Tom Vier

Contrary to what Apple would have you believe, all of its 5¼ inch disk drives—the older Disk II, the 5¼ inch UniDisk, the two-drive DuoDisk, and the IIc external drive—are essentially the same. Within certain guidelines you can mix and match Disk IIs, the IIc external drive, and the 5¼ inch UniDisk (but not the newer 3½ inch UniDisk) all you want, though some combinations require modifying the connector plug.

The controller card Apple is selling with the 5¼ inch UniDisk and the DuoDisk is like an old VW Beetle with a Continental kit. It's nothing more than a classic Disk II interface card with a new connector hanging off the back. The major difference is that the Disk II interface has two 20-pin header connectors, one for each drive, while the UniDisk/DuoDisk/IIc interface has a single, female DB-19 connector.

The drives themselves are also similar, except for the wrappers. The UniDisk/DuoDisk/IIc drives are standard ¾-height drives in Apple's latest designerware cases. The UniDisk has a permanent cable that terminates in a male DB-19 connector. On the back of each UniDisk there is also a female DB-19 connector. If you have two UniDisks, you plug the first one into the interface card and the second one into the back of the first. The IIc external drive is similar to a UniDisk, but since it will always be the second drive, it doesn't have the female connector on the back. The DuoDisk has a single female connector on the back and comes with a separate cable that has male DB-19 connectors on each end. One end of the cable plugs into the interface card, the other into the DuoDisk.

The DB-19 connectors on the UniDisk, DuoDisk, and IIc external drives all share functionally the same electronic signals, or pinout. And these are equivalent to what's found on the Disk II's 20-pin header connector. The following chart shows what's where:

Apple II disk drive pinouts

signal	use	Uni-DuoDisk	IIc external	Disk II
GND	ground reference	1-4	1-4	1,3,5,7
-12	-12 volts DC	5	5	9
+5	+5 volts DC	6,16	6	11,12
+12	+12 volts DC	7-8	7-8	13,15,17,19
WRPROT	write protect	10	10	20
PH 0-3	stepper motor phases	11-14	11-14	2,4,6,8
WRREQ	write request	15	15	10
ENBL	drive enable (low)	17,(9)	17	14
RDDATA	read data	18	18	16
WRDATA	write data	19	19	18
EXTINT	external interrupt	NA	9	NA
	not connected	NA	16	NA

The two 20-pin connectors for drive 1 and drive 2 on the old Disk II interface card are identical. All pin-pairs, in fact, always carry the same signals except pin 14. This pin activates the drive—a Disk II drive ignores all signals until pin 14 tells it to pay attention.

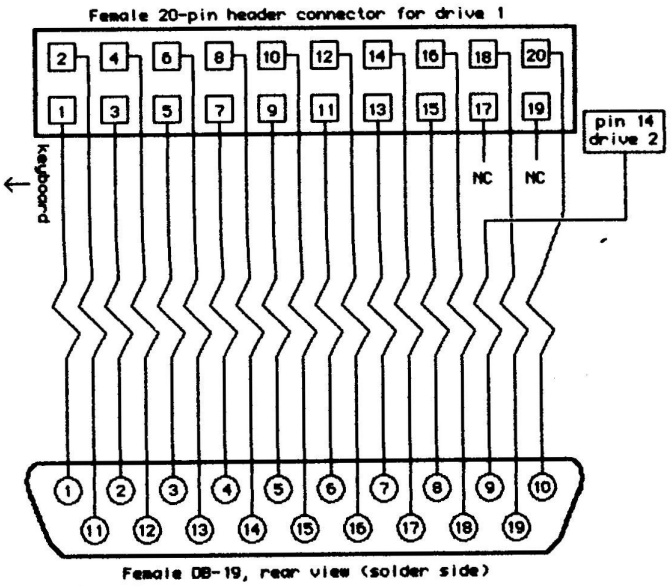
Unlike the Disk II interface, the UniDisk/DuoDisk/IIc interface has only one connector. If more than one drive is used, the drives are connected in a daisy-chain. Pin 17 carries the signal that enables the first drive in the chain. The interface card sends the signal that enables the second drive on pin 9. Inside a UniDisk, the signal from pin 9 is routed to the daisy-chain connector's pin 17. Consequently, all of the newer drives are electronically the same—each looks for its own activation signal on pin 17.

The IIc, of course, supports only one external drive. Pin 9 on the IIc is used for external interrupts. Pin 16, which otherwise carries a 5 volt supply of power, is unused on the IIc. This makes no difference because there is a second source of 5 volts on pin 6—and inside all Disk II drives these two lines are tied together.

All this means that any Disk II compatible add-on drive, with a IIc adapter, can be daisy-chained off of a UniDisk or plugged directly into a UniDisk/DuoDisk controller or a IIc. IIc adapters are readily available from several mail order houses (see the August issue, page 61, for the address of one). In addition, a IIc external drive plugs right into a UniDisk or a UniDisk/DuoDisk controller, and a UniDisk plugs into a IIc.

You can use the newer drives with an old-style interface card. This set-up would be most useful to people with special, non-Apple disk controller cards, such as those with diagnostic routines on them.

While the pinout table given earlier makes it appear that building an adapter to go between the DB-19 and 20-pin header connectors would be a complicated soldering job, everything literally falls into place. The accompanying diagram shows how to make an adapter that converts a Disk II controller into a UniDisk/DuoDisk controller. Be forewarned, however, that a wrong connection could be very unhealthy for your system. Success would also be limited by your tolerance for the potential Radio Frequency Interference that could find its way around your house if you use unshielded cable, and by your ability to buy the relatively rare DB-19 connector.



My Two Bits

by
Tom Weishaar

The computer revolution has brought many marvelous wonders to the people of earth. Least among these is the California lawyer. Among other things, California lawyers write the warranties that software publishers use. Here, for example, is the warranty you get with AppleWorks:

Even though Apple has tested the software described in this manual and reviewed its contents, neither Apple nor its software suppliers make any warranty or representation, either express or implied, with respect to this manual or to the software described in this manual, their quality, performance, merchantability, or fitness for any particular purpose. As a result, this software and manual are sold "as is," and you the purchaser are assuming the entire risk as to their quality and performance.

Compare Apple's warranty to this next one, which I found on a box of Kix breakfast cereal this morning:

GUARANTEE — If you are not satisfied with the quality and/or performance of the KIX in this box, send name, address, and reason for dissatisfaction — along with the entire box top and price paid — to:
General Mills, Inc.
Box 200-A
Minneapolis, MN 55440
Your purchase price will be returned.

General Mills projects a consumer-oriented image. It appears confident about the quality of its product and is proud to offer that product to the public.

Apple, on the other hand, projects an anti-consumer, wimpish image. It appears totally unsure about the quality of its product. The only thing it appears sure of is that it wants to push responsibility for its own mistakes off onto consumers.

(It gets worse. The really big phonies in the software business warrant the disks themselves — which they bought from somebody else — to avoid *looking like* wimps. Instead they look like deceivers.)

The Apple warranty, and the hundreds of others like it from other software publishers, shows no confidence, no pride, and no goodwill. It is a California legal masterpiece. It puts Apple in an unassailable *legal* position. However, it also shows a total disregard for customers — not the kind of policy on which Fortune 500 companies are typically based.

I contend that such guarantees are actually damaging to software publishers in the long run because they put the consumer in a no-win situation. The magnificent legal citadels of the software publishers are impenetrable. However, psychological studies have repeatedly demonstrated that people placed in no-win situations stop playing the game. If a software publisher is unwilling to promise that its product has value, what is the point of buying it rather than borrowing a copy from a friend?

Thus, the California lawyer's concentrated, narrow-minded focus on the legal issue of product responsibility backfires. The total disregard for consumer goodwill that is entombed in the typical software warranty ends up to be quite costly for software publishers. I think the negligible cost of servicing a real warranty on a qood-quality product would be much less.

What the lawyers are apparently afraid of is that customers will blame a faulty software product for damage to data and other programs. If a software product did, in fact, erase a few hundred hard drives, the potential liability could wipe a software company out. The need for software publishers to *limit* their liabilities, however, isn't a good excuse to *totally eliminate* them.

I don't think anyone buying a program that costs less than $500 should expect to be able to collect damages in excess of the program's purchase price, even if the program is clearly at fault. People who want to be able to collect indirect and consequential damages shouldn't buy mass-market software.

On the other hand, a software product should be warranted to perform as described in its advertising and documentation. If the software doesn't perform, the buyer should be able to return it for a full refund.

Here in Kansas we don't have any California lawyers (though we *are* just 20 minutes away from the Harry S Truman Sports Complex, home of the world champion Kansas City Royals) to talk me out of putting a guarantee where my mouth is. However, I did have to place the elf who has graced the back page of ***Open-Apple*** low these many months in job retraining to do it. In his place from now on you'll find, mixed in with a bunch of other fine print, a real warranty. Unlike Apple, Inc., I am proud of my product, confident of its quality, and willing to guarantee your satisfaction.

AppleWorks Pie

Back in the August issue I wrote a tips-and-tricks overview of AppleWorks and promised "more next month." Next month is finally here. This time we're going to take an in-depth look at the AppleWorks clipboard.

What the clipboard can't do. There seems to be a bit of confusion among AppleWorks users about the clipboard. The clipboard is where the things you cut out of your documents go to wait until you paste them back in somewhere else. Only one item can be on the clipboard at a time.

You can put things onto the clipboard with either the open-apple-C(opy) command or the open-apple-M(ove) command. C(opy) leaves your original material intact; M(ove) deletes what you select from the original material. You paste the contents of the clipboard back into documents with these same two commands. Just as you would expect, C(opy) will leave the contents of the clipboard intact; M(ove) will leave it empty.

What's puzzling about the clipboard is that you can't cut a section out of a spreadsheet, for example, and paste it into a word processor document. The clipboard can *hold* a piece of any kind of document, but *pasting* requires that what's on the clipboard be the same kind of stuff as the document you are pasting into.

Thus, you can cut or copy a section out of a word processor document and paste it back into the same or any other word processor document. You can cut or copy records out of a data base and paste them back into the same or any other data base. And you can cut or copy rows out of a spreadsheet and paste them back into the same or any other spreadsheet. But you *can't* go from the word processor to the data base or spreadsheet, or do any other kind of cross-type move, with one simple exception.

The exception is that data base and spreadsheet *reports* can be open-apple-P(rint)ed to the clipboard — such reports become word processor documents. This effectively gives you an easy way to move spreadsheet and data base information into the word processor. Notice, however, that you must use the P(rint) command, rather than the C(opy) or M(ove) commands, to get the information onto the clipboard and that data can flow in only one direction — into the word processor.

Clipboard size. The clipboard is pretty big, but not massive. It can hold a maximum of 250 word processor lines, 255 data base records, or 255 spreadsheet rows. If you try to cut, move, or print something larger than this to the clipboard, you will get a message saying it can't be done.

The clipboard uses up space on the AppleWorks desktop, though you can never tell exactly how much. Neither can you tell what is on the clipboard without actually pasting what's there into a document of the appropriate type. There is no direct command for emptying the clipboard (something you might want to do when desktop space gets tight). However, you can empty it by moving (not copying) what's on it into a document. Alternatively,

you can almost empty it by copying or moving a single record/row/character to it—this erases what was there previously, since the clipboard can only hold one thing at a time.

The clipboard and the AppleWorks word processor. Inside the word processor, the copy and move commands are quite flexible in terms of the size of blocks that you can work with. Blocks can be as small as a single character or as large as the 250-line limit mentioned earlier. To select a block, move the cursor to either end of the block and press open-apple-C(opy) or open-apple-M(ove). You'll be asked whether you want to copy or move text within the document or to or from the clipboard.

"Within-document" copies and moves involve a combined cut-and-paste operation that doesn't use or disturb the contents of the clipboard. After you tell AppleWorks whether you're doing a within-document or clipboard transfer, it will tell you, "Use cursor moves to highlight block, then press Return."

You can move the cursor with the four arrow keys, of course. Don't forget, however, that you can also move it with the open-apple-1 through open-apple-9 keys. This makes it easy to select blocks that extend from the cursor position to the beginning or the end of the document. And whether you move forward or backward the character the cursor was on when you initiated the command will be included in the selected block.

Any formatting commands embedded in the selected block will be transferred, just as you would expect. In fact, if you use the open-apple-Z(oom) command to display formatting instructions, C(opy) or M(ove) can capture the instructions without any of the surrounding text. This feature can be very handy when you create a document that switches back and forth between complicated formats. Or you can create a file full of special formats, use open-apple-Q(uick change) to bring up that file, copy a specific format into the clipboard, then go back to your original document and copy the format into your text. This trick is much faster than creating special formats by hand.

The clipboard's 250-line size limitation in the word processor is based on the number of lines that appear on the screen. At the bottom of the word processor's display is a counter that tells you the line and column position of the cursor—this is the same line count that the clipboard uses. You can increase the amount of word processor text that will fit on the clipboard by increasing the amount of text in a line—either by widening the margins or choosing a type face with more characters per inch. However, the maximum number of characters you can get in a line will still be the same as the maximum you can display on the screen—77 by my calculations. The 250-line limitation also applies to reports you P(rint) to the clipboard from the spreadsheet or the database.

The clipboard and the AppleWorks spreadsheet. In the word processor, the only difference between the C(opy) and M(ove) commands is whether the original block is erased or not; the only difference between within-document transfers and clipboard transfers is what happens to the contents of the clipboard. Users are often quite confused when they move to the spreadsheet or database and discover that the differences between C(opy) and M(ove) and between within-file and clipboard transfers are much more profound.

In the spreadsheet, *within-file* copies and moves mimick traditional spreadsheet commands. For example, if you choose C(opy) and "within-worksheet," you get what most spreadsheets call "replication"—a very powerful feature for duplicating formulas. If you choose M(ove) and "within-worksheet," you get a traditional spreadsheet choice of moving rows or columns. Unlike many spreadsheets, however, AppleWorks allows you to select a block of rows or columns to be moved; you are not limited to moving one row or one column at a time.

Clipboard moves and copies are similar to the standard within-worksheet moves, but with a couple of important differences. First of all, clipboard moves and copies can only be done with blocks of *rows*. Columns cannot be moved to the clipboard. Secondly, clipboard moves and copies remember the *relative* position of cells outside the block being transferred. Within-worksheet moves, on the other hand, remember the *absolute* position of cells outside this block.

For example, if row 11 includes a formula to sum the contents of cells A1 through A10, and you M(ove) that row to row 31 the *within-worksheet* way, the formula will continue to point to cells A1 through A10 after the move. But if you M(ove) it *via the clipboard*, the new formula will point to the 10 cells just above the formula's new position (A21 through A30). This gives you a 1-2 punch that many spreadsheets lack.

And don't forget, of course, that you can use the clipboard to M(ove) or C(opy) a block of rows from one spreadsheet to another. What gets transferred to the new spreadsheet is the *formulas* (not the *results of formulas*) from the original spreadsheet. (If you insist on moving results, P(rint) them to a DIF file, use the DIF file to make a new spreadsheet, and use the clipboard to move them from the new spreadsheet to the spreadsheet you want them in.)

The clipboard and the AppleWorks data base. Just as the spreadsheet allows you to put only whole rows on the clipboard, the data base allows only whole records. Word processor users who are comfortable using C(opy) and M(ove) on single words and parts of sentences often find this hard to believe, but unfortunately, it is true.

In addition to remembering the differences between M(ove) and C(opy) and between within-file and clipboard transfers, in the database there are also slight differences in these commands depending on whether you are in the multiple-record display or Z(oom)ed into the single-record display.

In the *single-record display*, M(ove) doesn't do anything. C(opy) will make up to 99 duplicates of the current record—it asks you how many duplicates you want. There is no way to use the clipboard when you are in the single-record display.

In the *multiple-record display*, C(opy) once again has the power to make up to 99 duplicates of the record the cursor is on. You can also use it, as well as M(ove), to transfer a single record or a block of records to or from the clipboard.

M(ove) can be used to actually rearrange the order of the records in a data base file. The open-apple-A(rrange), or sort, command is the more typical method for rearranging the order of records, however.

The real beauty of M(ove) and C(opy) in the data base is the way they allow you to split and merge files. For example, imagine you have a file that tracks magazine and newsletter articles. A friend who subscribes to ***Open-Apple*** would like a copy of that portion of your file. How do you separate these records for her?

First, load *two* copies of your main file onto the desktop. Enter the second one and use the open-apple-N(ame) command to change the name of the file, and the open-apple-D(elete) command to empty it. This file is now a skeleton that holds all your category names and display and report formats, but no data (actually, there will still be one record left in the file—there always must be at least one—but you can nip it out later). Now Q(uick change) to the first copy of the file and use open-apple-R(ecord select) to display only those records related to **Open-Apple**. C(opy) them to the clipboard. Q(uick change) back to the empty file and C(opy) them from the clipboard. This kind of flexiblity and speed is what makes AppleWorks great.

Problems can develop when you move data from one data base file to another, however. The *number* of categories, at least, poses no problems. If the file you are copying into has more categories than the one you copied from, the extra categories are left blank. If the copy-from file had more categories, the extra ones are lopped off.

The problem is with the *order* in which the categories appear. Records are transferred between the clipboard and a file with the categories in the order in which they were originally defined. To see what that order is, press open-apple-N(ame). *You cannot change this order*, so if moving data between various files is important to you, you must have this in mind when you set up the files to begin with. (Consider making all files copies of a single, mother file.) Using the open-apple-L(ayout) command to change the order in which categories appear on the screen *does not* change the order in which they are stored on the clipboard.

Although neither C(opy) or M(ove) can be used with information in a single category, there are a few tricks you can use for copying the contents of a category in one record to the same category in another record. The most powerful is what the AppleWorks documentation calls the open-apple-" or ditto command. Many people have trouble getting this command to work—the problem is that it should really be called open-apple-'. If you press the shift key to get the quote mark, this command doesn't work.

Say you have a file on insects. You originally coded butterflies as BFL in this file, but now you think BTFLY would be a better code. Is there any quick way to make a mass change from BFL to BTFLY? Sure. If this category doesn't already appear on the screen when you're using the multiple-record display, use open-apple-L(ayout) to move it over. Important—as you exit from L(ayout), specify that you want the cursor to move *down* the screen when you press return. Now use open-apple-R(ecord select) to remove everything but the BFL records from the screen. Change the first one to BTFLY. Put the cursor on the second one, press and hold down on open-apple and the apostrophe, and don't blink. (Don't panic, either. If some of your records seem to disappear it's because the record selection rules are looking for BFL—once you change the records to BTFLY they no longer fit the selection rules.)

Ask (or tell) Uncle DOS

Some 2nd looks at 800K

In spite of your comments about the cost and capacity of Apple's new 800K 3½ inch drives, I find them very attractive. I don't use Pascal or CP/M and have converted almost totally to ProDOS. I have a computer at the office and one at home and find that I must carry a lot of work back and forth. The capacity, portability, and sturdiness of the new disks make them very attractive to me.

Larry Miller
Lafayette, Ind.

As a subscriber since your first issue, I'm aware of your bias toward the combination of the Apple II and DOS 3.3. But your comments on the cost-effectiveness of disk storage systems in the October issue (page 73) seemed a bit, uh, emotional?

I've often seen cost-per-byte comparisons between the Disk II and hard disk subsystems. The hard disk always wins, of course, with its much lower cost relative to its storage capacity. But each comparision assumes that fixed magnetic media is superior to removable disks, and then laments the absence of backup failities. Your solution, to use two chained Siders, one to backup the other, does solve this problem.

But how many users need ten megabytes of on-line storage? If you're running a bulletin board, maybe. If your database management system is searching through several hefty files, definitely. But even when I'm working with a pretty full AppleWorks desktop, about 700K, I don't need access to every other file that I've got. Those files that I do have on the desktop are related, and are usually stored on the same disk, so backing up what I've just changed isn't all that difficult. (Aside: I use a one megabyte RamWorks.)

My usage may not be typical, but for what I do, I don't need a hard disk. I have many small program files, and not just large data files waiting to be manipulated; I'm not running a business where speed is essential; and I don't mind needing to swap disks just to find a file in a different category than my previous task. I would like to store those monster AppleWorks files on one disk, though. And the cheapest solution, in my opinion, is to buy one UniDisk and a dozen double-sided disks.

As for the lack of UniDisk support for DOS 3.3, I noticed a mention in the November *Nibble* of a fully compatible (but slightly limited) version of DOS 3.3, called *UniDOS 3.3*.

E.C. Floden
Hanover Park, Ill

In addition to Nibble's ***UniDOS 3.3****, two other versions of DOS 3.3 for the UniDisk have appeared already. One is called* ***AmDOS****; the author, Gary Little, who's written several very good Apple II-related books, will let you have this one for $15 (Gary Little, #210 131 Water St, Vancouver BC Canada V6B 4M3). The second is called* ***Profix 2.1*** *and is published by Nordic Software, Inc. (4910 Dudley St, Lincoln, NE 68504 800-228-0417/402-466-6502).*

I haven't seen any of these products, but the one from Nordic Software may have a big head start on the other two. Nordic has been very quietly selling a version of DOS 3.3 for Apple's ProFile hard drive for some time. (Get it, ***Profix****?) The new version allows you to store DOS 3.3, ProDOS, and Pascal files on the same drive; to boot from a UniDisk 3.5 or ProFile directly into DOS 3.3; to mix or match various-sized volumes on a drive; and to use up to 14 drives at once. It comes with backup and restore utilities and is compatible with* ***ProntoDOS****. (They know how to get a plug in this newsletter, don't they?) The price for all this is $69.95.*

My initial dislike of Apple's new drives is slowly weakening for all the reasons you mention. I still think Apple made a major marketing mistake by not building DOS 3.3 support into the drive, but that is now water under their bridge and they'll have to swim in it.

In addition, I recently got a second Sider drive, which is quickly bringing my rapture with these things to a slow simmer. Remember the letter in the April issue (page 29) about the fellow who returned his Sider because of all the noise it made? I think I got his reject. This one sounds like the disk has spokes with playing cards permanently attached as noise makers. It also occasionally emits a sound like pebbles falling into a metal barrel.

When I tried to contact the Sider technical support team for help, I got either busy signals or a tape saying to call between 8 and 5 pacific time, Monday through Friday. I got the tape at noon on Tuesday and at one on Thursday.

So I called Darrell Echols, Marketing Manager at First Class Peripherals, to find out what was up. He said they've been running a sale — good till December 31 — with the 10 megabyte Sider selling for $595 and their new 20 megabyte version for $895. That's a savings of $100 on each. The sale increased sales, just as it was supposed to. Consequently, the technical line was jammed with impatient new purchasers either wanting to know when their drive would be shipped or wanting installation help. He also mentioned First Class is interested in hiring qualified Apple II technical support people; if you're interested in moving to Carson City, Nevada give him a call (800-538-1307).

Echols told me that the two primary complaints First Class gets about its drives are the documentation and the noise. As I've experienced here, the noise problem doesn't occur in all units, or even in affected units all of the time. They've addressed the documentation problem by producing a new user's guide and utility software — now being shipped with new drives and available to present owners for $20. They've also figured out what's causing the noise problem, Echols said, and all drives shipped after November 1 should be relatively quiet. Older noise makers will be fixed under the Sider's standard one-year warranty.

Echols told me that during its first year in business, First Class sold more than 15,000 hard disks to Apple II users. He also said that First Class will soon announce a tape backup unit.

The incompatiblity problem

I concur with your October editoral concerning the new Apple UniDisk 3.5. While I applaud Apple's good intentions in providing users with more disk storage, they have, in my mind, disregarded a very important problem: media incompatibility. It will be bad enough for current Apple owners trying to decide whether to upgrade to one (two?) of these new drives, with the attendant problems — which to use as the default drive, transferring data back and forth, and so on. It's going to be far worse on *new* byters. I see the average new Apple II owner being forced to buy three drives, two of one kind (either 5¼ or 3½) for normal work and on of the other so as to guarantee the ability to run all the software out there. The only advantage I see to the UniDisk 3.5 is that software publishers will have the ability to pack larger and more complex programs onto one disk.

A far better solution would have been the following: a new 5¼ inch drive, double-sided, double-density with around 570K (143 times four) of storage, along with a controller card that would read either new 570K or old 143K disks *on the new drive*. Ideally the new 570K drive would accommodate all the popular operating systems, but even if it accepted only ProDOS and Pascal in the 570K mode, it would run these and DOS 3.3 and CP/M in the 143K mode. Old-timers would buy the new drives to stay up to date, and new buyers could buy the new drives confident of the ability to run all exising software. IBM (shudder!) saw this soon after its PC was introduced, and upgraded from 160K to double-sided 360K drives without user resistance.

Such a drive with an attractive price, say $350 retail, and the Apple seal of approval would sell like hotcakes.

My work involves medical research in airway pharmacology. We have used a dedicated Apple IIe to run a high-pressure liquid chromatograph for over a year. It runs the system perfectly and at one-third the cost of a custom-designed controller. We're about to buy a second Apple to run another one. We are using a II-Plus to perform real-time acquisition and processing of data from a multi-channel polygraph recorder through an analog/digital board. That's three Apples dedicated to controlling state-of-the-art scientific instrumentation. Why Apple, Inc. doesn't pursue this market aggressively is beyond me; there isn't a better system for the money.

Steven R. White, M.D.
Chicago, Ill.

Fuzzy-wuzzy was a monitor

In your October article (page 73) on the Apple's new UniDisk 3.5, you ask how Apple expects to sell these things. It's simple. They have 3,000 stores and, unfortunately, of the 3 million Apple II users, 10 per cent at best are familiar with alternative products not made by Apple computer.

The quality of Apple's new color monitors does not measure up to that of RGB. In fact, Apple's own literature says "this product is for the consumer who primarily uses color software, but occasionally uses 80-column productivity software." When you compare the 80-column text of the Apple monitors with that of a good RGB monitor like the Taxan 420, you'll agree that the 80-column text on the Apple monitor is poor at best.

Also in the October issue you said, "RAM prices are dropping rapidly." I see no reason to count on continuing RAM price declines — these prices tend to oscillate. In the last 30 days we have seen our cost of 64K RAM chips double. Many people expect 256K RAM prices to increase around the first of the year.

Naturally, I enjoyed your criticisms of the Apple memory card. The primary disadvantage we see is

that it must be installed in slot 4 or 5. The type of customer who buys a large memory card is already using or saving those slots for something else.

Dan Pote, President
Applied Engineering
Carrollton, Texas

The 80-column text on Apple's new monitors is definitely not as sharp as what you see on monochrome or RGB monitors. However, those 80-column characters are no fuzzier than 40-column text on a television—the only other alternative if you want cheap color. You wouldn't want to use AppleWorks 40 hours a week on one of these monitors, but I continue to think they will be a big hit in homes and schools. They do both color and 80-columns at the lowest possible price. I've been trying to buy a monitor like this for years; I'm glad somebody is finally manufacturing them.

Buzz, clatter, pop

I own an Apple IIe with DuoDisk drives. The drives have always made a horrible buzzing noise whenever I format a disk or when an I/O error occurs. They have otherwise functioned properly for over a year—my only complaint is the noise. I asked an Apple dealer about this and he could only tell me that this was "normal." Is this true? If so what is going on inside the drives that causes this noise? Can it be eliminated (the noise keeps my wife awake when I'm working late into the night)?

Alan Richter
Drexel Hill, Penn.

When I bought my first disk drive, I had it back at the dealer's within 24 hours because of all the noise it made. The technicians said there was nothing wrong with it. I didn't believe them.

Five and half years later that drive still makes lots of noise. However, it hasn't crossed the dealer's threshold since—it's never required repair or adjustment of any kind. If my lawnmower ran like that I wouldn't have buffalos grazing in my front yard.

The noise at disk initialization occurs because DOS 3.3 tells the drive it should move the drive head outward 40 tracks. Since the head is rarely as far in as track 40, however, this causes the head to clatter against an internal stopper 20 or 30 times. The function of this is just to make sure the head gets moved to track 0.

DOS 3.3 also pulls this trick when you boot a disk and when a disk error occurs. ProDOS, on other hand, is much quieter about disk errors—perhaps your wife would enjoy getting a copy for Christmas.

80-column clocks, too

Until your November issue, I had been trying unsuccessfully for over six months to figure out a way to read the time from my clock card and display it on an 80-column menu screen. The problem was that reading the clock called for a command sequence like this one:

```
20 PRINT D$;"PR#4" : PRINT D$;"IN#4"
30 INPUT "%";CL$
40 PRINT D$;"PR#0" : PRINT D$;"IN#0"
```

But PR#0 is incompatible with 80 columns. If I changed it to the 80-column-compatible PR#3, however, my screen would be erased. I asked several different knowledgeable people who said what I wanted to do couldn't be done—not even Applied Engineering's technical support people could help me. Everyone said I had to come out of the clock reading with a PR#0. Consequently, my 80-column program ended up with 40-column menu screens.

But I just tried the tips you gave in the November issue (page 82) for turning printers off with my clock card and, lo and behold, IT WORKED. Now I have my menus in 80-columns, just like the rest of my program. Please point out to your readers that the warm PR#3 will work with a clock card as well as a printer.

Frank Adler
Lakewood, NJ

Legal papers served

Your November issue sure solved a crisis here in Parker County. We bought a IIe with the AppleWorks/RAMWorks combo for our D.A.'s office, but could not get AppleWorks to support legal length forms. Most of the documents from that office simply had to be on "legal," so boy was our faces red! Anyway, after consulting with just about everyone, we were about to buy a second printer. That little tip about AppleWorks' "accepts top-of-page commands" foible has really saved the day here in Weatherford and throughout the county of Parker. You sure made the D.A.'s bunch happy. Lord help the poor soul that ever gives you a hot check in Parker County.

Gary Maddox
Weatherford, Texas

I discovered that foible one day when I was trying to print on mailing labels with the AppleWorks word processor. It would leave eleven blank labels between every one it printed on. I thought this was a Texas-size waste of mailing labels.

It seems that if you tell AppleWorks your printer can handle top-of-page commands, it will send a form feed at every page break rather than a carefully measured series of carriage returns. This works fine as long as you use the standard 11 inch sheet of paper or figure out some way to tell your printer you are using a different size. Unfortunately, the paper length command (open-apple O, PL), doesn't tell your printer anything.

AppleWorks does a marvelous job of counting carriage returns correctly, however, so things work best if you simply set "accepts top-of-page commands" to "no."

SpeedDemon vs Titan Accelerator

We recently compared the SpeedDemon card with Titan Technologies' Accelerator IIe in preparation for buying a product that would speed up our work with *VisiCalc Advanced Version*. The Accelerator IIe seems to be anywhere from a little to a lot faster than the SpeedDemon, depending on the application.

According to Titan, their power consumption is less, too. On the other hand, their price is higher; but I've seen Titan's card advertised by the mail order houses for as little as $219.

Are there other speed-up cards than these on the market? Would I be able to plug a 65802 into the 65C02 socket on my Accelerator IIe and get even more speed?

Ken Sarkozy
Kalamazoo, Mich.

These are the only two speed-up cards I've heard of. The 65C02 chip in these cards is a special version that is capable of executing instructions 3.5 times faster than the standard 65C02. The cards don't actually speed up your Apple 3.5 times, however, because the Apple's memory chips can't respond that quickly. The cards include a small amount of expensive, high-speed memory. Machine language program segments are moved into this high-speed memory for execution. As I understand it, the cards show different amounts of speed increase because they have different algorithms for deciding what to move into high-speed memory.

If you could get a special high-speed 65802, it would probably plug into your Accelerator and work. My understanding is that high-speed versions of this chip are not yet available, however. In addition, you wouldn't get "even more speed" from the 65802 unless you were using software written specifically for that chip. The speed boost possible with 65802 and 65816 chips comes only with programs written to use their advanced capabilities. These chips can also run older 6502 programs—but not any faster than a 6502 can.

Copy deprotection hints

I would like some information on deprotecting popular software packages so that they might be put onto a hard disk.

Roger W. Todd
Murfreesboro, Tenn.

Software protection and deprotection are two subjects that have never interested me very much, so you'll see precious little in this area in ***Open-Apple.*** *There are, however, two Apple-II-oriented publications that are almost completely devoted to this subject. They are* ***Hardcore Computist*** *($20/6 issues: P.O. Box 110846-T, Tacoma, WA 98411 206-474-5750) and the brand new* ***Apple Software Protection Digest*** *($24/yr: 2068 79th St, Brooklyn, NY 11214 718-232-8429).*

Fortunately for those of us with a dislike for copy protection, many software developers seem to be coming to the same realization that Beagle Bros, Penguin, and a few other companies came to long ago—copy protection is a bad ***marketing strategy.*** *It kills sales without stopping illegitimate copying. Lots of the best software hitting the market today isn't copy protected to begin with—including the best-selling program in the world, AppleWorks.*

Three easy questions

I have three questions:

1. I wanted to change the volume number of a DOS 3.3 disk. I tried zapping byte 6 of the Volume Table of Contents (track $11, sector 0), which several sources indicate is where DOS stores the volume number of a disk, but it didn't work. The DOS Manual says that the volume number of a disk can't be changed without reinitializing the disk. Is this true? What is the function of the byte I zapped? If DOS doesn't use that byte, where does it get the volume number it displays with the CATALOG command and that it uses to check for VOLUME MISMATCH errors?

2. In your January issue (page 2), you described a technique for trapping an END OF DATA error when reading a text file with an Applesoft program. I tried this technique in a program to read a random-access file that had some missing fields in some of the records. If the program encountered an END OF DATA, it was supposed to move to the next record. But instead, it behaved as if the record length had been changed to 1, and went back and read stuff from low-numbered records. I got it to work right by reOPENing the file (with the correct value of L) in the error-handling routine, but I sure wish I understood why.

3. I have read that you should always complete a

FOR-NEXT loop before leaving it; otherwise you may eventually overflow the stack. There are times, however, when it is awkward to observe this rule meticulously. The Applesoft POP command is used when you want to exit a subroutine without a RETURN. Can POP or something similar be used to offset the evil effects of a premature exit from a FOR-NEXT loop?

Paul Nix
Summit, N.J.

I have three answers:

1. The volume number used by CATALOG and VOLUME MISMATCH is embedded within the sector headers that also tell DOS which track and sector is currently passing over the read-write head inside your disk drive. The only easy way to change these is to initialize a new disk with the volume number you want and then use FID to copy your files over to the new disk. Alternatively, Quality Software's ***Bag of Tricks****, by Don Worth and Pieter Lechner, includes a program that can initialize single tracks and preserve the data in them. Byte 6 of the Volume Table of Contents is set to the current volume number by DOS when a disk is initialized but is never used thereafter.*

2. Under DOS 3.3, the END OF DATA error causes the file in use to be closed. Your error handling routine no doubt looped back to the READ statement, which has (under DOS 3.3 only) the undocumented power to OPEN a pre-existing file. It did so, setting the record length to the default of 1. This creates an incredible mess, which you handled correctly by reopening the file with the correct record length in the error-handling routine.

3. The CALL -3288 trick from the January issue will also clear the stack of unfinished FOR-NEXT loops.

EXEC inside programs

Why does the "EXEC" command work the way it does? If used half way through a program, the computer will run the whole program before EXECing things.

Allen Hawthorne
Vernon, Alab.

Applesoft operates in two modes—immediate-execution mode (you type commands in on the keyboard) and deferred-execution mode (you RUN a program). The EXEC command can be used to type a series of commands that have been saved in a text file into Applesoft's immediate-execution mode.

When a program is running, on the other hand, you can't EXEC immediate-executions commands, just as you can't type them in on the keyboard. However, EXEC ***can*** *be used when a program is running to provide response to INPUT and GET statements. This can allow you to automate the execution of certain types of programs.*

You can automate the execution of FID, for example, by creating a text file that BRUNs FID and gives it commands. The following EXEC file, named COPY ME, will execute FID and copy itself onto another disk:

```
BRUN FID,D1
1
6
1
6
2
COPY ME
Q
9
```

If you insist on EXECing a series of immediate execution commands in the middle of a program, you'll have to put an END just after the statement that prints your EXEC command. By using the CONT(inue) command at the end of the EXEC file, you can restart the Applesoft program where it left off, assuming you haven't changed it in any way. Or you can restart your program at the end of your EXEC file with a GOTO linenumber, or a RUN linenumber, or even an ON X GOTO linenumber. The point to remember is that you must stop execution of the Applesoft program—and thus enter immediate-execution mode—before you can EXEC a file holding immediate-execution commands.

EXEC and AppleWorks

I am having trouble controlling AppleWorks with an exec file. Is it possible that AppleWorks lacks the capability of being automated? It seems like the AppleWorks startup disk clears my exec file and takes control. If this is the case, is there any way of avoiding this handicap?

Richard Hare
Los Osos, Calif.

Under ProDOS, exec files are creatures of and are controlled by Basic.system. The ProDOS kernel doesn't know an exec file from a Wozniak. When you start up AppleWorks, it knocks Basic.system out of memory and takes over. Your exec file dies under suspicious circumstances.

(Let's hope the AppleWorks system program tells the ProDOS kernel to close all open files as one of its first actions—else ProDOS will think the exec file is still open, though it will have no idea what to do with it. All system programs should attempt to close all files at startup; there's no reason to assume the previous system program closed its files, although it should if possible.)

In my mind you are exposing the biggest weakness of AppleWorks—the lack of a way to automate it. It needs the ability to execute command files and it badly needs a "macro" or "glossary" function such as those in ***GPLE*** *or* ***Apple Writer****. Rumor has it that the author of AppleWorks, Rupert Lissner, is hard at work giving the program a mouse interface. I hope this rumor is wrong, as AppleWorks already has one of the best human-computer interfaces of any program going. What it needs is macros and command files. Are you listening, Rupert?*

AppleWorks hates reset

Did you know that if you press control-reset while using AppleWorks it will hang? You have to reboot; you lose all your data. Why would you want to do such a thing? Maybe, like me, you have a Grappler-Plus with a Bufferboard, or some other *internal print buffer.* The only way of clearing the internal buffer, if a document is not printing out the way you expected, is to push and hold control-reset for a few seconds.

But AppleWorks hangs if you push control-reset, so you wait it out, wasting a lot of paper and causing needless wear on the printer. Is there any way to add reset protection to the working copy of your AppleWorks disk? If you know of any, I (and my printer) will be eternally grateful.

Marc S. Renner
St. Paul, Minn.

You're right, it does hang. Isn't that a pain the chips? Why doesn't it warmstart like other commercial-quality programs? I don't know of a patch, but I suspect there's a ***subscriber out there somewhere*** *who can come up with one.*

HEADLINE TYPE

Is there any way the Imagewriter's headline type can be accessed from AppleWorks?

Raymond J. Schuerger
Sewickly, Pa.

The manuals don't make it very clear, but you can get the Imagewriter's headline type by specifying a small characters-per-inch number within AppleWorks. The Imagewriter-AppleWorks combination provides 10 different character widths—4, 5, 6, 7, 8, 9, 10, 12, 15, and 17 characters-per-inch (the Imagewriter's 13.4 characters-per-inch mode isn't used by AppleWorks).

The different character widths are combinations of the different sizes built into the Imagewriter and the Imagewriter's elongated or double-wide character mode. If you specify a characters-per-inch setting the Imagewriter doesn't have, you get the next wider size. For example, if you specify 13 or 14 characters-per-inch, AppleWorks will put a few more characters on a line, but will still use the ImageWriter's 12-characters-per-inch characters.

Apple Plotter and AppleWorks

My department purchased an Apple Plotter. The plotter recognizes only control-C as the end-of-text-to-be-plotted command. It uses other control characters as well. The manual describes the various ways to send commands to the plotter, including a section on how to use *Apple Writer's* control-V feature. However, it neglects to say anything about AppleWorks.

Since the plotter isn't going to recognize any of the normal AppleWorks word processing printer codes, I decided to rename everything and make a chart of the changes. The characters-per-inch option for custom printers allows one to define 20 different command sequences, more than enough for the plotter. For example, I have specified CI 4 as control-C. From the AppleWorks word processor I use open-apple-O to select CI 4 whenever I need a control-C.

Duncan Orr
Heber City, Utah

Others who try this trick should remember that AppleWorks sends the current characters-per-inch code as well as a margin code at the beginning of each ***line.***

Spilling *Mousepaint*

How do I get *Mousepaint* to print? I have a IIc, a PBI interface, and a Gemini 10X.

Jim Willis
West Monroe, La.

My understanding is that the only printers supported by ***Mousepaint*** *are Apple's Dot Matrix Printer and Imagewriter. Other printers aren't supported because they all use different protocols for printing graphics. Generally the protocol starts with a command sequence that says "the next x-number-of-bytes are graphics", followed by a string of characters in which the bits represent which pins on the dot matrix print head should be fired.*

Not only does the command sequence vary from printer to printer, but some use just six bits, some seven, some eight; for some the most significant bit is the top pin, for others the bottom pin; and so on.

With the IIc, the only way to proceed is to save your ***Mousepaint*** *graphic in a file and then use a*

graphic-dump program to print it. There are several such programs available. According to the ads, there is one for Epson printers built into your PBI interface, but your Gemini isn't compatible with Epson graphics.

*Take a look at Beagle Bros' **Triple Dump**, which comes with both DOS 3.3 and ProDOS versions (you'll need the ProDOS version for **Mousepaint** graphics). **Triple Dump** supports the Gemini 10X and the IIc serial interface (I'm not sure what effect your PBI serial-to-parallel converter may have).*

Large laser print

We make an Apple II word processor that works with voice, regular size print, large print, and braille output. Our software uses embedded commands to control format. We would like to provide a LaserWriter driver for large print—14 or 16 point Courier—in the LaserWriter's Diablo 630 emulation mode. Do you have any idea how to get the LaserWriter to come up in this mode? Are there any contract programmers who know PostScript?

Jesse Kaysen
Raised Dot Computing
408 South Baldwin
Madison, WI 53703
608-257-9595

*I sure get tired of tooting Don Lancaster's horn, but once again, he's the person you need to talk to (602-428-4073). His column in **Computer Shopper** has carried tons of information about the LaserWriter and its programming language, PostScript. Lancaster insists that the debate is already over—PostScript is the language of choice for printing and typesetting. He predicts all other advanced printing products coming down the pipeline will use it.*

*He also insists that the best computer to use with a LaserWriter is an Apple IIe running **Apple Writer**. See his letter in the November 4 issue of **InfoWorld** (page 54) for his opinion of using the LaserWriter with the Macintosh.*

I've included your complete address so that any PostScript programmers within earshot can contact you.

Open-Apple

[redacted]

is written, edited, published, and

ISSN 0885-4017. Published monthly since January 1985. Worldwide prices (in U.S. dollars; airmail delivery included at no additional charge): $24 for 1 year; $44 for 2 years; $60 for 3 years. All back issues are currently available for $2 each; seven or more from any single volume $14 (postpaid). Index mailed with the February issue. ***Open-Apple*** is available on disk for speech synthesizer users from Speech Enterprises, P.O. Box 7986, Houston, Texas 77270 (713-461-1666).

Source Mail: TCF238 CompuServe: 70120,202 Tele.: off hook

Still another CONVERT

Lots of people seem to have missed one easy but undocumented way of converting programs between ProDOS and DOS 3.3. The IIc System Utilities disk's "copy files" option will convert individual files between the two systems. Apparently, this option checks the DOS on the source and target disks during a copy and matches things automatically. The option works well with one drive. Why Apple didn't mention this in their manual, I don't know.

Mark Evans
Minneapolis, Minn.

In the October ***Open-Apple*** (page 77) a reader says the IIc System Utilities program is unsatisfactory for coverting files between ProDOS and DOS 3.3 on a single-drive system. He should try the program's "copy files" option.

By the way, converting the whole disk from one operating system to the other can wreak havoc with the high order bits in ASCII literals. The result is that certain Applesoft programs will not RUN or LIST properly. It's not so good to have your ASCII literals converted into Applesoft tokens.

Robert C. Moore
Laurel, Md.

Error message in error

When using ProDOS, sometimes I get a NO BUFFERS AVAILABLE message. I try to CLOSE any open files, and still get the same message. There are only one or two buffers being used, so it's not that there really aren't any available. What's going on?

Allan Hoffman
Whitestone, NY

There are two ways to get the NO BUFFERS AVAILABLE message from Basic.system. One is to try to open a new file when fewer than 1,024 bytes of free memory are available. This can occur when you have a very large program or a program that uses a very large array or two. The only commands that can detonate this type of error are OPEN, APPEND, EXEC, and Basic.system's GETBUFR call.

*The second type of NO BUFFERS AVAILABLE message occurs when you attempt to BLOAD a file into a memory area that is marked as protected in the ProDOS system bit map. **Open-Apple's** basic Basic copy program in the July issue (page 50-52), for example, caused this problem to occur with the binary files it copied. The copy program mistakenly set the loading address for all copies of binary files to zero (a fix for this problem was given in the October issue, pages 74-76). Thus, when you try to bload one of these copies, Basic.system tries to put it at $0000, but ProDOS can't do it because page zero is protected in the system bit map.*

In this case you really should get an error message something like BAD LOADING ADDRESS, but you get the buffers message instead. This problem can occur with BLOAD, BRUN, and the dash command.

*If you look at the table "Errors by ProDOS command" on page 213 of Apple's **Basic Programming with ProDOS** manual, you'll see that it neglects to mention that BLOAD and BRUN can precipitate this error. It also says the NO BUFFERS AVAILABLE message can occur with both versions of the catalog command, but because both these commands use the temporary command buffer, I doubt this is true.*

A keyboard input-anything

In the September ***Open-Apple*** you said, "There is no way you can write an Applesoft program that has data-entry and editing facilities as good as those in, say, AppleWorks." I would like to assure every Applesoft programmer this is not true. The entire realm of the blinking-underline cursor and the magic menu interface is available in an easy-to-use machine language package. And every Apple IIc owner already has a copy.

Shocked? Catalog the IIc Systems Utilities disk (sorry, DOS 3.3 lovers) and you'll discover that main program thereon was written by some nameless Applesoft programmer. All the PRINTs and INPUTs you have been responding to for shuffling your files are tied lovingly to Applesoft with the & command.

The anonymous file SU2.OBJ contains a machine language routine that installs itself in Basic.system buffer space and includes the ampersand commands &INPUT, &GET, &PRINT, and &EXIT. Serious hacking needs to be done to document this file and the Applesoft program that uses it, which is called SU.

I have gone over &INPUT and found that it includes:

```
* the blinking-underline insert cursor with the
    DELETE key enabled.

* setting ES=1 before calling &INPUT enables the
    escape key and returns ES=1 if the escape key
    is pressed or ES=0 if it isn't.

* the variables OA and SA enable the the open- and
    solid-apple keys. The variables OP$ and SO$
    hold the keys that, when pressed in conjunction
    with open-apple or solid-apple, cause an exit
    from the &INPUT routine.

* the maximum length of an input line is held in the
    variable FL. Any character you wish to fill the
    input area with can be held in the variable FL$.
```

The &INPUT command doesn't allow any prompts and can only input string variables. The syntax is, for example, &INPUT AN$—only the variable name may change. The routine works on both the Apple IIe and IIc and also seems to work properly in both 40 and 80 columns.

The &GET command displays the blinking underline cursor and, unlike Applesoft's 80-column GET, can return escapes and presses of the apple keys.

I hope you can spread the word to Applesoft programmers about this file, because it finally lets homemade programs have "data-entry and editing facilities as good as those in, say, AppleWorks."

Christopher W. Hogue
Tecumseh, Ont.

That's quite a little discovery you've made there, Christopher. The file SU2.OBJ, like all such programs from Apple, insists on being the first one alloted space by Basic.system. I played with &INPUT and found that it also supports control-Y (erase to end of line), control-X (erase whole line), and control-R (replace original text). You can use the routine to edit an existing string like this: EM$="Edit me" : &INPUT EM$. There are more adventures here for the intrepid—any volunteers?