Open-Apple™

May 1986
Vol. 2, No. 4

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

Releasing the power to everyone.

May I have your attention, please?

It's all Don Lancaster's fault. I suppose I have some responsibility as well, but it was his idea. To quote from his seminal work, *The Incredible Secret Money Machine*:

Employees are a hassle, a waste of time and money and a psychic energy sink. You should avoid them at all costs.

Your incredible secret money machine should have 0.834 employees—that is 83.4 per cent of you, nothing more, no less. The remaining 16.6 per cent of you should go for fun and rewind time.

Unfortunately, ***Open-Apple*** has been taking up about 834 per cent of my time. So I went back and reread what Lancaster suggests people like me do. "Find others whose money machines are up and working and use them to do the things you can't handle yourself," he says.

Lancaster's book was my primary instruction manual for starting ***Open-Apple***. (A very good manual it is, too—highly recommended for those of you who are in business for yourself or who are thinking about starting up something. It's a mere $7.50 from Synergetics, Box 809, Thatcher, AZ 85552 602-428-4073.) My only serious problems have come in areas where I didn't follow his advice. So a couple of months ago I decided it was time to start working less.

Another address. The sad, sad consequence of all this is that ***Open-Apple***'s subscription records are no longer kept on an Apple II. Your names, addresses, and expiration dates were transferred to 9-track mag tape this month and are now sorted and batch processed on a McDonnell-Douglas minicomputer running the Pick Operating System.

I want to make it perfectly clear that the limiting factor around here was me. The Apple II I've been keeping your records on had plenty of capacity left (until this month your names and addresses were stored in a huge AppleWorks file on a IIe with 1 megabyte of memory, a 10 megabyte hard disk, and an accelerator card). This system and one full-time person could handle a subscription list three or four times larger than what ***Open-Apple*** now has (that would be 12 to 16 thousand names). But I'm the only full-time person around, and I have newsletters to write.

The folks I chose to handle ***Open-Apple's*** subscription fulfillment specialize in fast, accurate work for small publications. They come to us with high recommendations. Sally Tally, our new circulation manager, promises improvements over the rather uneven level of customer service I've been able to provide you myself (some weeks I was real good; other weeks I was terrible), even though she has to use a minicomputer rather than an Apple II.

Because of this change, ***Open-Apple*** has a new post office box for all subscription-related correspondence, such as new orders, payments, and changes of address. It is:

Open-Apple Our main subscription address
P.O. Box 6331
Syracuse, NY 13217

We also have new invoice and renewal forms—for the most part they look like the ones real magazines use. Whenever you get one of these you'll also get an envelope, with our new address printed on it, to send it back in.

Abby, the little lady who used to urge you buy a multi-year subscription from the inside flap of our remittance envelope, reached ***Open-Apple's*** mandatory retirement age of 95 and is now working behind the counter at McDonald's. She may make some purely promotional appearances for us in the future, however.

Of course, none of this applies to those of you in Australia and New Zealand, but just to keep things consistent, we have a change of address for you, too. Our circulation manager in your earth-quadrant is:

Kerry Branson Our southern subscription address
c/o Cybernetic Research Ltd
576 Malvern Road
Prahran, Victoria 3181 AUSTRALIA

Uncle DOS and I still live here in Kansas City, or, more precisely:

P.O. Box 7651 Our editorial address, world-wide
Overland Park, KS 66207

Another Uncle DOS. The role of Uncle DOS is now often played by Dennis Doms, whose name has been appearing more and more regularly in the letters section of this newsletter. Dennis is a writer and certified Apple hacker who has been answering Apple questions from local user group members, electronically and in person, for years. He has now started his own little sideline answering most of the mail we get around here. He thinks it's wonderful to get paid to answer Apple questions, and I think it's wonderful to get these stacks of paper off my desk, and you, I hope, think it's wonderful to have someone to write to about your problems who will actually write you back. Keep those cards and letters rolling in folks. Uncle DOS loves them.

A bound volume of ***Open-Apple's*** first 13 issues—January 1985 through January 1986, inclusive—is now available. Our index appears on the covers. It sells for $14.95. I still have single copies of all back issues for $2 each. I have also decided to start selling the *DOStalk Scrapbook*, by me and Bert Kersey, for $14.95. There's a program disk that goes with that book for $10. We have a third book that will be officially announced in a moment, if I don't forget again this month.

Book prices include surface delivery anywhere in the world; for airmail outside North America add $1 per order plus $4 per book. Book and back issue orders will be handled fastest if you send them to our Syracuse address, though we'll eventually fill your order no matter how you get it to us.

"I ALWAYS BACKUP EVERYTHING."

```
Column:  (  A  )(  B  )(  C  )(  D  )(  E  )(  F  )
Row:
  1 |    Concrete requirement estimator.                    F O R M U L A S
  2 |    For new shop building.                             ================================================
  3 |
  4 |    Concrete required:    32.37 cubic yards            ((50*30*(6/12))+(2*49*0.8*(12/12))+(2*0.8*28.5*(12/12)))/27
```

Figure 1

```
Column:  (  A  )(  B  )(  C  )(  D  )(  E  )(  F  )
Row:
  1 |    Concrete requirement estimator.                    F O R M U L A S
  2 |    For new shop building.                             ================================================
  3 |
  4 |                          Main   East &  North &
  5 |                         Floor    West    South
  6 |                           Pad Footings Footings
  7 |    Length (feet):          50       49      .8 ft.
  8 |    Width (feet):           30       .8    28.5 ft.
  9 |    Depth (inches):          6       12      12 in.
 10 |    Concrete required:    32.37 cubic yards            ((C7*C8*(C9/12))+(2*D7*D8*(D9/12))+(2*E7*E8*(E9/12)))/27
 11 |
 12 |
```

Figure 2

Another book. According to the March 1986 issue of *Apple Assembly Line*, Tom Weishaar (egads, that's me!) has written a new book called **Your Best Interest: A Money Book for the Computer Age**. Now just because I've forgotten to mention this book here in ***Open-Apple*** for two months running doesn't mean that I'm not proud of it. I'm just easily distractable, that's all.

I wrote *Your Best Interest* during the summer and early fall of 1984, in between the death of *Softalk* and the birth of ***Open-Apple***. However, it's just hitting the bookstores now. It's the only computer-related piece I've done that isn't specifically related to the Apple II. The book is about interest rates (how to calculate the "time value of money" using a spreadsheet program) and related stuff (death, taxes, inflation, and deception mostly).

Since my own opinion of the book is highly biased (I think it's the best piece of work I've ever done), let me provide you with a second opinion. Here are the most complimentary parts of Bob Sander-Cederlof's review in *Apple Assembly Line*:

Your Best Interest *isn't about Apple assembly language, but I cannot resist telling you about it anyway.*

The book is about interest rates—how to understand them, how to calculate them, how they affect you. It was written for people who know how to use a spreadsheet program. All the hard math and books of tables are replaced by your favorite calc-alike....

Seven fascinating chapters lead you to an understanding of how financial transactions really work....

Have you thought about buying a house recently? Tom shows you how to figure the true cost of an adjustable-rate mortgage, how to compare different financing schemes, and how to protect your money. You'll learn about the tricks the money lenders sometimes use to take advantage of unwary investors and borrowers. And all is tied to spreadsheet models you can put into your Apple. I wish I had only known how to do these things when I bought a pickup truck last summer. Or leased a copying machine three years ago. And when we bought some land in the country....

The book is published by InfoBooks of Santa Monica, Calif. You can get it at your local bookstore (some have it with the computer books, some with the business books) or you can order it from ***Open-Apple*** for $9.95, postpaid.

I haven't talked much about spreadsheets in ***Open-Apple***, but it's not because I don't use them. *Your Best Interest* revolves around spreadsheet programs.

One of the greatest limitations of today's spreadsheet software is that users think of it only in the context of financial calculations. In fact, a spreadsheet is ideal for number crunching problems in the areas of spelunking, nursing diagnosis, and space exploration as well. One of the first magazines to recognize this and to consistently emphasize it is called *AgriComp*. It covers the area where computers and agriculture overlap ($24/year, 103 Outdoors Bldg, Columbia, MO 65201).

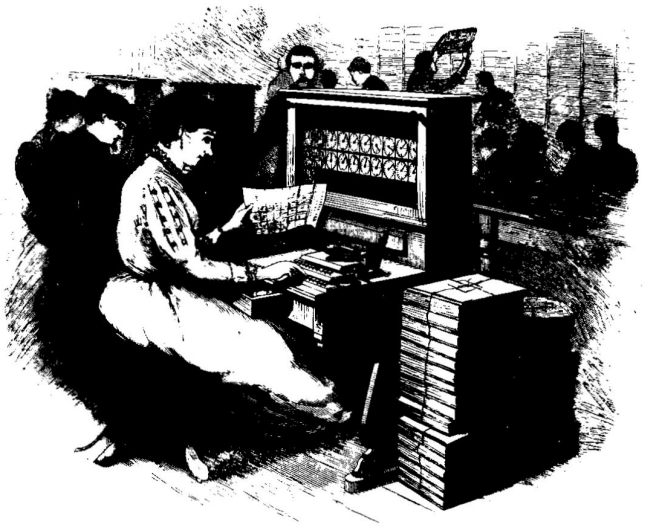
Since I've been busy this month hand-cranking 9-track tape through my UniDisk, I got *AgriComp*'s permission to show you the following article, which is one of the best I've ever seen on *practical* spreadsheet uses.

The Ad Hoc Spreadsheet

by Mark Wilsdorf

Ad hoc is a Latin phrase meaning "to this." In today's language, "ad hoc" is normally used to denote something created or established for a particular purpose (*to this* purpose). We often think of the thing established as being temporary. An ad hoc committee is a committee established for a particular, special purpose; often with the idea that the committee will be disbanded when its job is completed.

An ad hoc spreadsheet is one you set up to do some one-time job, not a job that you plan to do again and again. Most of us piece together an ad hoc spreadsheet as if we were a 10-year-old making a tree house—plenty of ideas and imagination but not a lot of planning. You don't want to spend too much time planning and building the spreadsheet, because you might only use it one time, to find one answer. But with just a little planning—or by developing a few good habits for building spreadsheets—the job of building



***Open-Apple*'s new circulation manager, Sally Tally, will track your subscriptic on a McDonnell Douglas Microdata Reality 4750 minicomputer.**

the spreadsheet will be faster and easier. And if you do store a copy of the spreadsheet on disk for possible later use, a bit of planning and documentation will make it much easier to reuse at a later date.

Consider the example of estimating how many cubic yards of concrete are required for the foundation of a grain bin or other building you are constructing. Every situation is different enough (circular versus rectangular area, level or unlevel surface, different footing depths, etc.) that you almost need a completely different spreadsheet each time you do concrete work.

However, with little additional effort, you can create a spreadsheet that will calculate the answers you want without requiring much of your time for spreadsheet development, that will produce a well-labeled printout (one that will still make sense if your work gets put off for a couple of weeks), and that will potentially be useful in other similar situations—in case you need to estimate some concrete requirements again next year.

```
Column: <  A  >< B  >< C  >< D  >< E  >< F  >
Row:
  1 |  Concrete requirement estimator.                          FORMULAS
  2 |  For new shop building.                                   ========
  3 |
  4 |  DATA:
  5 |                        Main   East &  North &
  6 |                       Floor    West    South
  7 |                        Pad Footings Footings
  8 |    Length (feet):       50       49      .8 ft.
  9 |    Width (feet):        30       .8    28.5 ft.
 10 |    Depth (inches):       6       12      12 in.
 11 |
 12 |  CALCULATIONS:
 13 |    Pad:                          27.78 cubic yards        (C8*C9*(C10/12))/27
 14 |    East or west footing:          1.45 cubic yards        (D8*D9*(D10/12))/27
 15 |    North or south footing:         .84 cubic yards        (E8*E9*(E10/12))/27
 16 |
 17 |  RESULTS:
 18 |    Concrete required:            32.37 cubic yards        +D13+(2*D14)+(2*D15)
```

Figure 3

```
Column: <  A  >< B  >< C  >< D  >< E  >< F  >
Row:
  1 |  /// Concrete Requirement Estimator              \\\   FORMULAS
  2 |  /// For new shop building.                      \\\   ========
  3 |  /// Prepared 09-Aug-85                          \\\
  4 |  ------------------------------------------------------
  5 |  DATA:
  6 |                        Main    East &  North &
  7 |                       Floor     West     South
  8 |                        Pad    Footings Footings
  9 |    Length (feet):       50       49      .8 feet
 10 |    Width (feet):        30       .8    28.5 feet
 11 |    Depth (inches):       6       12      12 inches
 12 |
 13 |  ------------------------------------------------------
 14 |  CALCULATIONS:
 15 |    Pad:                          27.78 cubic yards        (C9*C10*(C11/12))/27
 16 |    East or west footing:          1.45 cubic yards        (D9*D10*(D11/12))/27
 17 |    North or south footing:         .84 cubic yards        (E9*E10*(E11/12))/27
 18 |
 19 |  ------------------------------------------------------
 20 |  RESULTS:
 21 |    Concrete required:            32.37 cubic yards        +D15+(2*D16)+(2*D17)
```

Figure 4

```
Column: <  A  >< B  >< C  >< D  >< E  >< F  >
Row:
  1 |  /// Concrete Requirement Estimator              \\\   FORMULAS
  2 |  /// For: 24' Grain Bin                          \\\   ========
  3 |  /// Prepared 09-Aug-85                          \\\
  4 |  ///                                             \\\
  5 |  /// REMARKS: Volume of concrete in footing is   \\\
  6 |  /// calculated by subtracting volume of the     \\\
  7 |  /// cylinder represented by the inner wall of   \\\
  8 |  /// footing, from the volume of the cylinder     \\\
  9 |  /// represented by the outer wall of footing.   \\\
 10 |  ------------------------------------------------------
 11 |  DATA:                             Inner    Outer
 12 |                        Main        Wall     Wall
 13 |                       Floor         of       of
 14 |                        Pad       Footing  Footing
 15 |    Diameter (feet):   25.5        24.33      25 feet
 16 |    Depth (inches):       6          18      18 inches
 17 |
 18 |  ------------------------------------------------------
 19 |  CALCULATIONS:
 20 |    Pad:                           9.46 cubic yards        (@PI*(C15/2)^2*(C16/12))/27
 21 |    Inner wall cyl. volume:       25.83 cubic yards        (@PI*(D15/2)^2*(D16/12))/27
 22 |    Outer wall cyl. volume:       27.27 cubic yards        (@PI*(E15/2)^2*(E16/12))/27
 23 |
 24 |  ------------------------------------------------------
 25 |  RESULTS:
 26 |    Concrete required:            10.90 cubic yards        +D20+(D22-D21)
```

Figure 5

Here's a list of good habits to cultivate that will make your ad hoc spreadsheet development time more productive.

Put a descriptive label at the top of the spreadsheet. Your label doesn't need to be either long or fancy, just something that will immediately identify the spreadsheet's purpose when you load it into your computer or see it printed out.

Whenever possible, put each piece of data in a separate cell that is labeled with names and units of measurement. When you begin to assemble the data you need for a simple spreadsheet, it may be unclear how all of the data will be used. Labeling the data and placing it in separate cells will allow easier access as you build formulas—and will assure that your printouts show what data was used in calculating the results. Compare Figure 1, where the data is all lumped together in a formula, with Figure 2, which has all data separately entered and labeled. Which spreadsheet printout would you have more confidence in if you looked at them again one week from today?

If possible, reserve one area of the spreadsheet for nothing but data. Use a separate area for calculations and possibly even a third area for final results. Have you ever looked at a spreadsheet printout and wondered whether a number had been entered as data or was calculated by a formula? Don't allow "creeping confusion" to take over your spreadsheet as it grows larger. Divide your template into areas with different functio avoid potential problems—including accidental erasure of formula confusion over what data was used, ease of conversion to similar proble and so on.

You may also want to consider dividing your formulas into smaller chunk as done in Figure 3. Shorter formulas are easier to decipher than a single long formula, such as the one used in Figure 2.

Don't spend much time sprucing up your spreadsheet until you are mostly done with it, and unless you are likely to use it again later. One common waste of time during spreadsheet development is in trying to keep a template looking "perfect" in early stages of development. All the work you put into centering titles and drawing borders and fancy labels will be wasted if you later find that your spreadsheet requires wider columns or a change in structure.

For example, you might realize after the spreadsheet is finished that it could be printed on a single sheet of paper with a few modifications. Only when you are fairly sure of your spreadsheet's final form should you invest much time in "prettying up" its appearance—and then only if you will use the spreadsheet frequently enough to justify the investment of your time.

Save a *permanent* copy of the spreadsheet on a disk when you're finished with the job. You may want to go back later and check your calculations for errors. Or you may decide to do the job a bit differently—such as increasing the depth of the concrete footing—and you'll need to go back and change some data. Without a disk copy of your spreadsheet, you'll have to go through the whole development process again.

Other times, the need to do a similar job may pop up sooner than you expected. I deleted a disk copy of a spreadsheet for figuring a grain bin pad one fall, only to put up another bin of the same dimensions the following fall. It's a good idea to keep a disk or two of such miscellaneous spreadsheets. Often just a few changes are all that is needed to rework an old spreadsheet for a new situation. Figure 5 shows a spreadsheet for figuring concrete needs for a grain bin pad, which was developed by reworking the spreadsheet in Figure 4.

No AppleWorks @PI

If you try to type *AgriComp's* Figure 5 into an AppleWorks spreadsheet, you'll make the amazing discovery that AppleWorks has no @PI function. It's also missing @LOG10, which is used a couple of times in *Your Best Interest.*

This discovery led me to do a quick comparison of the functions available in the five spreadsheet programs I have at my fingertips. They include two DOS 3.3 programs—the original DOS 3.3 version of *VisiCalc* and *IACcalc* (which also goes by the names *The Spreadsheet* and *MagiCalc*). ProDOS entries were *MouseCalc, SuperCalc,* and AppleWorks.

All five include the following functions: ABS(olute value), AVERAGE, COUNT, ERROR, IF, INT(eger), LOOKUP, MAX(imum), MIN(imum), NA(not available), SQRT(square root), and SUM.

All but *SuperCalc* have a CHOOSE function; all but *MouseCalc* have NPV(net present value).

All but AppleWorks have EXP(e to a power), LN(natural log), LOG10, PI, AND, OR, NOT, TRUE, and FALSE.

AppleWorks and *MouseCalc* are both missing ISERROR and ISNA. AppleWorks and *VisiCalc* are missing ROUND. AppleWorks, *VisiCalc*, and *SuperCalc* are all missing ROW and COLUMN.

Only *VisiCalc* and *SuperCalc* have SIN(e), COS(ine), TAN(gent), ASIN, ACOS, and ATAN.

Only *SuperCalc* has MOD, RANDOM, four additional financial functions, eight date functions, ISDATE, ISNUM, and ISTEXT. *SuperCalc* also allows text "values."

Only *SuperCalc* and *MouseCalc* will draw graphs of your spreadsheet.

Only *SuperCalc* and AppleWorks are unprotected and can be moved to a hard disk.

As you can see, AppleWorks is missing lots of functions available with the other four packages. On the other hand, only AppleWorks has a spreadsheet larger than 60 columns by 256 rows. AppleWorks allows 127 columns by 999 rows. This isn't quite as wonderful as it seems, however, because only 60 cells on any single row can hold a formula, but it still beats the pants off the other four.

As a test, I recently made two very large spreadsheets using AppleWorks and Applied Engineering's desktop expansion software. In each cell I put a simple formula that added 1 to the value in the cell just above or to the left. One BIG.SHEET was 60 columns by 256 rows — the total capacity of the other spreadsheets — it used 387K and took four minutes to recalculate. My second BIG.SHEET was 60 columns by 999 rows — the AppleWorks maximum for formula-holding cells — it used 673K and took 6 minutes to recalculate. Of course, if you used more complex formulas your spreadsheet would use even more memory and take even longer to recalculate — but it can be done if you want to do it. Incidentally, those recalculation times are unaccelerated. A speed-up card would probably cut them in half.

I'm not satisfied with any of these five spreadsheets. Because I'm often running AppleWorks anyhow, I usually use its spreadsheet when I need to crunch some numbers, but I'm dismayed at the paltry selection of math functions it offers. My second choice is *IACalc*. It is a definite improvement over the older *VisiCalc* disk I have, with features such as eighty-column display, recognition of most memory cards (it gives me a 280K workarea on my 1 megabyte Apple), and automatic splitting of wide spreadsheets into pages when printing.

SuperCalc has possiblities. It is clearly the powerhouse here, but comes with a CP/M-style user-interface that's foreign to anything else I use. It reminds me of the shock of arriving in Portuguese-speaking Brazil after traveling through Spanish-speaking areas of South America. Armed with my high-school Spanish, I can eventually figure out the system in places such as Peru and Bolivia. In Brazil and in *SuperCalc*, however, I am totally lost. I can't even change money. Those of you who really need *SuperCalc*'s additional powers may find it worthwhile to spend the time it will take to learn how to use it. I haven't been able to justify the investment.

MouseCalc, as I have mentioned here before, scrolls so slowly it has no practical value. That's too bad, because otherwise it has possiblities.

Bus School

The Orange Squeezer Mod

by Tom Vier

Letters in the December (page 95) and January (page 101) issues of ***Open-Apple*** document a serious problem with Orange Micro's internal printer buffers. These devices capture what you send to your printer at machine-language speed. Then they spoon feed the data to your printer at a rate it can keep up with. Your Apple itself can also spoon feed a printer, of course, but without a printer buffer it can't attend to both you and the printer at the same time. With a buffer, you and your Apple can go back to word or data processing while your printer and the buffer clunk away.

Orange Micro sells two versions of its printer buffers. The Buffered Grappler-Plus is both a printer interface card and a buffer. The Bufferboard is a slot-resident device that requires a separate printer interface card. These boards are less expensive and less cluttering than external printer buffers, which demand a power outlet and desk space somewhere between your computer and your printer.

Orange Micro's slot-resident buffers, however, have one major drawback. They lack a switch to flush the buffer. If you tell your Apple to begin printing a long document and then realize you forgot to specify that you wanted the document double-spaced, for example, you need a way to tell the buffer to stop printing and throw its contents away. The flush-buffer signal that the Grappler boards use is an extended press of the reset key.

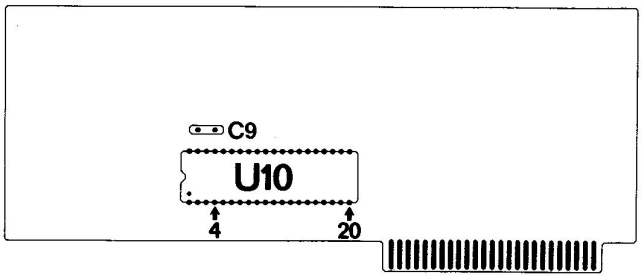
Unfortunately, a lot of software, including AppleWorks, won't tolerate this. If you press reset from within AppleWorks, you'll send the data on your desktop, as well as the data in the printer buffer, right on down the old porcelain convenience. Thus a simple desire to terminate printing can become a disaster.

Fortunately, there's a cure for this. We can install our own flush-buffer switch on Orange Micro's boards. It's very simple and very easy. All that is required is a little push button and a couple of lengths of small-gauge solid wire. The push button needs to be a normally-open, momentary type. The only tools you'll need are a wire cutter and a soldering pencil. The soldering pencil doesn't even need to touch the board unless you want to make the switch permanent.

Orange Micro uses the 8048 microcomputer chip to control its buffer boards. This chip has its own internal RAM, ROM and microprocessor. The 8048 monitors the Apple's reset signal through one of its I/O lines. It would be difficult to get the 8048 to flush the buffer by altering the signal on this line, however, because it would require extra isolation to prevent resetting the Apple's 6502 as well.

A much easier solution is to use the 8048's own reset line, which is available at pin 4. Pulling this line to a low voltage simulates a power-up and will clear the buffer immediately. To pull the line low all we have to do is to connect it to ground — conveniently available at pin 20 of the 8048. Our flush-buffer switch simply connects these pins together when the button is pushed.

First, attach the wires you'll be using to your push-button switch. Twist the wires together and cut them long enough so that you can place the switch in a convenient location. Now remove your card (either version, they are very similar) and place it before you, component side up, oriented as in the accompanying illustration. Locate the chip shown. It's hard to miss, since it's the biggest guy on the block. Remove the insulation from the tips of the wires and carefully insert one into the socket at pin 4 and the other into the socket



at pin 20. Do this with the chip in place — you don't even need to remove it. Make sure there is no bare wire exposed and your project is finished.

Optionally, if you want to make this installation permanent, solder the wires to the leads of capacitor C9 on the back side of the card. It's just above the 8048 (U10), as shown in the diagram. It also wouldn't hurt to put a .01 uf bypass capacitor across the leads of C9 to get rid of any noise the wires pick up. Put the card back in your Apple and run the switch out to a handy location.

Once installed, this switch makes it easy to stop a runaway document. First, get your program to stop printing, if it hasn't already. With many programs, including AppleWorks and *Apple Writer*, you do this by pressing the escape key. Next, press the flush-buffer switch to clear the buffer. Printing might not halt immediately because some printers also have small internal buffers. Turning off the printer will stop that cold.

Ask (or tell) Uncle DOS

April Fool is past

Was "Soft write-protect tabs" (April, page 2.23) an April Fool's joke, or is there something very different about the IIc disk hardware? These pokes will certainly blind DOS to the state of the write-protect switch, but they won't fool any self-respecting drive that I know of.

Michael Gruenthal
Durham, Calif.

The joke was on me. As many, many of you have noticed, those pokes only SEEM to work—the drive makes all the right noises and stuff—but nothing actually gets written onto the write-protected disk. I wondered why I had never read about those pokes anywhere else before. The genie in the machine really got me good this year. But I'll get even next April (look, your mouse is loose, etc.)

A poke from Australia

After saying that writers must "understand words and...spell them correctly" (January 1986, page 97), you mentioned that Bremner was Australian born. Is that anywhere near where I live?

I checked my IIc as you described on page 99 and I don't have the new miniassembler. Is there an old one?

Adrian Gallagher
Kanahooka, Australia

A little over 10 years ago I spent an afternoon sitting in front of five cranky University of Kansas professors. The occasion was an oral examination for a masters degree in journalism. Did they ask me about Walter Lippmann's influence on American journalism? No, they asked me how to ***spell*** *Walter Lippmann. ("Two 'p's," I said. "How many 'n's," they asked. "One?" I said. "Shame, Mr. Weishaar," they said.)*

Then Prof. Bremner, in his booming voice, said, "Mr. Weishaar, in your master's thesis you misspelled the word 'vegetable' twenty-one times. Could you please spell 'vegetable' for us?" I couldn't.

Should I go on? They did. It was like third grade. (I just wanted you to know that this isn't the first time I've had to take abuse about my spelling from an Australian.)

To get to the old miniassembler, you need a DOS 3.3 System Master disk. Boot it. When you see the Applesoft prompt, enter the command INT. This will put you into Integer Basic (the prompt will change to a greater-than sign). Then do a CALL -151, and a F666G. This gets you to the ! prompt of the old miniassembler. From there, enter a starting address, a colon, a space, an assembly language mnemonic, and, if required by the instruction, an address. The miniassembler will translate the mnemonic and address into machine code and place the correct numbers at the address you have specified.

Begin the next line with a space rather than with the address and colon. This will put that line's machine code right after the first line's. You can enter Monitor commands (old miniassembler only) by beginning the line with a dollar sign. Exit the old miniassembler with a $3D0G (for Basic) or $FF69G (for the Monitor). (You can exit the new miniassembler by pressing return on an empty line.)

Power supplies and track 0

I recently added a Sider hard disk to an enhanced Apple IIe system that had been having the ProDOS track 0 crash problems several users have written about in ***Open-Apple*** (January 1986, page 103; February, page 2.6; March, page 2.12).

With the Sider, the computer would reboot when I attempted to save a file to the hard disk. The file being saved would be destroyed or, in some cases, changed into a monster file. In one instance, a new, two-item AppleWorks database file was transformed into something too large to load into my 275K desktop. If I happened to be saving to a floppy, the old eat track 0 syndrome surfaced again.

I discussed the problem with First Class Peripherals. A technician there told me that most of the problems customers had experienced with the Sider were solved by replacing the Apple's power supply with one sold by Jameco Electronics for $39.95 (1355 Shoreway Rd, Belmont, CA 94002 415-592-8097). He said that Apple procured its power supplies from several sources in Asia and at least one of these suppliers seemed to have a quality control problem. (That would explain why only some enhanced IIes are having track 0 problems.)

I purchased the new power supply, installed it in about ten minutes, and have not experienced any problems since then. I have left the computer on for prolonged periods, repeatedly saved the same (expendable) file to hard disk and floppy and have had no crashes. I will leave it up to the technicians to explain why the power fluctuation resulted in the problems I experienced, but the bottom line is that $39.95 solved all my problems.

One final note: I wrote Apple on January 7, 1986 to describe the problem I was having. Thus far I have not received any response, not even an acknowledgement or a word of sympathy, much less any suggestion as to how to solve the problem.

I hope this will be of help to some of your readers.

John W. Patterson
Richmond, Va.

This is a follow up to my two letters concerning the track 0 crash problem (January 1986, page 103; March, page 2.12). It *may* be solved. Our school has been informed that, although Apple has not been able to find any problems with the two computers of ours that were sent to California, they would replace all 20 of our power supplies.

I hope this takes care of it, but the new power supplies haven't been installed yet so I'm not sure. I'll let you know. Thanks for your help.

J. Ernest Cooper
Lathrup Village, Mich.

The pieces of this puzzle are beginning to fall into place. If the track 0 crash problem is related to bad power supplies, that would explain why it is seen only in the newer enhanced IIe. It isn't the enhanced status of the machine that's causing the problem, but last summer's change in Apple IIe manufacturing facilities, which was probably accompanied by a change in component suppliers (July issue, page 49).

The power supply is the large metal box inside your II-Plus or IIe. It is held in place by four screws accessible from the bottom of the computer. The computer's on-off switch and AC-power connector are also built into the metal box. The switch and connector are accessible from outside the computer because of a strategic hole in the case that allows them to poke through. Inside the computer, a set of six wires comes out of the power supply and terminates in a single connector that plugs into the motherboard.

Replacing the power supply is only a little harder than changing a light bulb. Remove the AC-cord and the four screws. Lift out the supply. Squeeze the motherboard connector and pull it free. You may need to remove a card to get a good grip on the connector. Reverse this procedure to put a new supply back in. The hardest part is lining up the screw holes on the new supply.

Apple's power-supply specifications for North American computers call for the supply to perform adequately as long as the input AC power is within the range of 107 to 132 volts. The power supply provides the computer with four voltage levels. According to the Addison-Wesley edition of the ***Apple IIe Technical Reference Manual*** *(page 159), the power supply is designed to perform at the following specifications:*

nominal voltage	true voltage	maximum current
+ 5V	+ 5V plus/minus 3%	2.5 amps
+12V	+11.8V plus/minus 6%	1.5 / 2.5 amps *
- 5V	- 5.2V plus/minus 10%	.25 amps
-12V	-12V plus/minus 10%	.25 amps

* The +12V source can provide 2.5 amps intermittently. Intermittent operation is considered to be up to 20 minutes at the higher load if followed by at least 10 minutes at the normal load. Maximum power consumption of the power supply as a whole is 60 watts continuous, 80 watts intermittent.

The Fall 1980 issue of ***Apple Orchard*** *had an article called "Don't Overload Your Apple II" by Ken Silverman (page 67). The article lists a number of peripherals that obtain their power from the Apple (by means of the slot connectors) and the amount of +5V and +12V current used by each peripheral when off and when in use.*

Although the article is somewhat dated, the device that puts the most load on the +12V supply is still pretty common—5-1/4 inch floppy drives. Silverman says an active drive uses about .425 amps of +12V current. However, during the first second or so, while bringing the disk up to speed, a drive will draw about .7 amps, Silverman says. This is more than a quarter of the intermittant +12V power available from a normal Apple power supply.

If that much power isn't available because of a weak power supply, the rate at which the disk comes up to speed could be too slow. Another possibility is that drive speed could be erratic. Data written to the disk under these conditions could easily be unreadable later.

But why does the track 0 problem seem to occur only with ProDOS? I suspect the culprit here is the machine language code ProDOS uses when starting up a disk. It's different from that used by DOS 3.3. In DOS 3.3, the motor is started and RWTS waits a

specific period for the motor to come up to speed. The wait is determined by a byte called the "motor-on time count." This byte is defined in the "Device Characteristics Table," which itself is an attachment to the Input/Output Control Block, or IOB, used by the RWTS routine.

In standard DOS 3.3, the motor-on time count is stored at $B7FE. It is normally $D8 (216). Just for fun, start up DOS 3.3 and do a few CATALOGs. Pay close attention to how long it takes from the moment you press return until the catalog begins to appear on your screen. Then POKE 47102,0. This will change the motor-on time count to zero. Do some more catalogs. Notice how much faster the catalog appears on your screen?

You gain the additional speed by eliminating the motor-on wait. DOS begins trying to read the disk immediately. As soon as the disk is spinning fast enough to get readable data, DOS will grab the catalog.

However, don't try to make this poke a permanent part of DOS. When reading, an error caused by erratic disk speed isn't critical. The data associated with the error is thrown out and DOS tries to read the sector again. Uncle DOS doesn't throw up his hands and yell I/O ERROR until several unsuccessful attempts to read the sector have been made.

When writing, on the other hand, no check is made to see if the sector written is valid. No retries are ever made. Thus, a write that occurs before the disk speed has settled would be unreadable, but you wouldn't find out till you tried to use that sector again, later.

As noted, the ProDOS motor-delay code is different from DOS 3.3's, however, it functions in a similar manner. The drive motor is started and a timing loop causes a delay. Once the delay is over, ProDOS checks the data port for changing data. If the data changes, it assumes the disk is ready and proceeds to attempt the read or write. If the data has not changed after 255 comparisons, ProDOS sends an I/O error.

Because of the complexity of the code, I can't easily prove that the ProDOS motor-on delay is shorter than DOS 3.3's. In fact, based on the speed at which a catalog appears on the screen, DOS 3.3 appears to be the zippier of the two. But I suspect that some subtle difference in the coding makes a disk not yet fully settled at the correct speed look ok to ProDOS, while DOS 3.3, for whatever reason, gives the disk more time.

Why track 0? This is where ProDOS keeps a disk's root directory and free-sector map. The blocks on track 0 are constantly being rewritten. They get far more use than any other blocks on a disk. DOS 3.3's catalog is in track 17 ($11). If the power-supply problem was going to occur with DOS 3.3, it would probably be track 17 that got destroyed, rather than track 0.

It may also be significant that track 0 is the outermost, and thus longest, track on a disk. The magnetic media normally passes by the disk head faster when the head is on this track than on any other. This may amplify the effect of erratic disk speed, and could be another reason why ProDOS seems to be having problems with weak power supplies while DOS 3.3 isn't.

In addition to the Jameco supply mentioned in Patterson's letter, there are two other sources of new power supplies that I am aware of. One is your Apple dealer, who will exchange a rebuilt power supply for your old one at a price approximately double what Jamco is asking.

The other source is Applied Engineering (P.O. Box 798, Carrollton, Texas 75006 214-241-6060), which has started selling a heavy-duty replacement power supply for $69. Applied Engineering's ads for the new supply make it clear that most machines with all slots filled have probably reached the limits of Apple's +5V line.

(Based on Silverman's article, AE's ad appears reasonable, though the only memory card Silverman tested back in 1980 was Apple's 16K language card. It drew .168 amps on the +5V line. AE rates its own 1 megabyte RAMworks II at .4 amps and "typical" 1 meg cards at 1.1 amps. AE's estimates for things like clocks and printer interfaces appear a little high compared to Silverman's figures, but these items draw relatively little power compared to memory cards, speed-up cards, and disk interfaces, most of which weren't available to Silverman.)

Applied Engineering's replacement power supply provides 6 amps on +5V, 2.5 amps on +12V, and 1 amp on -12V and -5V. The intermittant ratings and total wattage weren't available when we asked. As input, the supply accepts either 90 to 135 AC volts at 60 hertz, or 180 to 270 volts at 50 hertz.

Another possibility, for the hacker types among you, is to get a hand-me-down IBM-compatible power supply. You'll have to change the connector and it won't fit inside the Apple II case, but the standard IBM power supply is heavy-duty compared to the Apple II's. You may be able to find such supplies at a good price because many IBMers had to replace their standard supply with a more powerful one in order to add a hard disk drive.

A final note: Apple hasn't yet notified dealer technicians of the power supply/track 0 crash problem. One would hope that Apple replaces all defective power supplies at no cost. Apparently they haven't yet figured out how to tell whether a power supply is defective or not, however. For the time being, don't expect your dealer's technicians to know as much about this problem as you do.

Is 300 baud dead?

Do you feel that 1200 baud is becoming the standard—that anyone who buys a 300-baud unit is wasting his or her money? I have been offered a (functioning) 1983 vintage Hayes MicroModem IIe, but with prices of 1200 baud units dropping (ever so slowly), perhaps it is better to wait. Obviously this is a personal thing, but I'd rather seek some advice than none at all.

Gerald Potkin
St. Charles, Ill.

Prices for 1200 baud modems are indeed creeping downward, but seem to have stabilized at about the $200-$400 range for now.

In calculating the price for a modem, remember to add up ALL the costs of the modem hardware. For an internal type, like the Hayes Micromodem, usually you only need the modem.

Most 1200 baud modems for the Apple are external types, however, like the Hayes Smartmodem. Cermetek, Prometheus, and Novation make internal modems for the Apple II. The internal modems are usually more expensive, but the external types require you to buy a serial card and, usually, a cable to connect the interface card and the modem together. Expect your minimum investment to be about $300, with the opportunity to spend more.

In the face of this, an economical 300 baud modem may be a good investment for several reasons.

First, you don't put yourself into severe debt while finding out if you really want to become a telecommunications junkie. I bought a Hayes Micromodem and a subscription to the Source the day I bought my first Apple II (no disk drives, printer, or software, but, by golly, I was online). However, I never used it enough to crave more speed until I started sending this newsletter to a typesetter over the phone. (That takes the better part of an hour at 300 baud.)

Second, if the things you want to do with a modem will happen while you are connected, 300 baud is a nice reading speed. The words kind of fly by at 1200. Thus, if you are going to read the material as it comes in (rather than quickly downloading some stuff and reading later), there's little advantage to 1200 baud. Consider how much of your "connect time" will actually be spent transmitting data, as opposed to how much will be spent reading, considering, attending to distractions, and typing.

Since many of the big commerical systems have a higher dollar-per-minute charge for 1200 baud users, it can actually be cheaper to sign onto these systems at 300 baud if you intend to read what comes in while you are connected. The speed of 1200 baud will offset the higher 1200 baud cost if you are spending most of your time transferring data, but if you are reading mail and have to stop every few lines to catch up, 300 baud is cheaper and just about as efficient.

Finally, 1200 baud may not always be usable. Depending on the condition of the phone lines in your area and other factors such as the phase of the moon, you can find some trash creeping into your 1200 baud communication line. Slowing down to 300 is often the solution to making things readable again.

So maybe 300 baud isn't a bad place to start. However, if you are serious about telecommunications, or if you have a lot of data to transfer, then the extra cost of a 1200 baud set-up may be justifiable. While 1200 baud does seem to be rapidly emerging as the standard telecommunications speed, 300 baud is universally supported, too.

Now, to make you crazy—part of the reason 1200 baud modems are dropping in price is that 2400 baud modems are out and getting cheaper. Maybe you should start at 300 and switch to 2400 later. Or maybe wait for 2400 to come down a bit more and start there. Or maybe... but decide soon—less expensive 4800 baud modems are already under development.

Making quit reboot

Is there a way to patch ProDOS, BASIC.SYSTEM, or any file to cause BASIC.SYSTEM to reboot upon exit (instead of prompting for a new prefix and pathname)?

Tommy Durham
Spartanburg, S.C.

One solution to your problem is to use a "program selector." This is a program that runs automatically when you exit a ProDOS program (it replaces the "new prefix/pathname" prompt). It allows you to choose what you want to execute next from a menu.

I've been using Glen Bredon's **ProSel** and I am convinced it is far superior to the other two widely-available program selectors, **MouseDesk** and **Catalyst**. Bredon has continued to enhance his program—he recently added DOS 3.3 support and raised the price to $40 (see March, page 2.10, "Don't try to sort ProDOS directories" for a more complete description of the program and what comes with it—521 State Rd, Princeton, NJ 08540).

A program selector replaces the "quit code" that is contained within ProDOS at $D100 of bank 2 of language card memory. This code is invoked by the ProDOS MLI QUIT call—all ProDOS system programs are supposed to end by making this call. For example, from Basic.system, type BYE to invoke the quit code.

The built-in ProDOS quit code prompts you for a prefix and pathname for the next application you want to run. This is fine, unless you want to run something using Apple Pascal, CP/M, or DOS 3.3, which makes the quit code seem awfully stupid for allowing only ProDOS responses.

I assume this is your motivation for asking for a "reboot" option.

With ProDOS 1.1.1 (the version number is important!), a short patch will cause the system to automatically boot when the quit code is activated:

```
BLOAD PRODOS,A$2000,TSYS
CALL-151
512D:CE F2 03 6C FE FF
3D0G
UNLOCK PRODOS
BSAVE PRODOS,A$2000,~~L1456B~~,TSYS
LOCK PRODOS
```

The machine code translates as DEC $3F4 followed by JMP ($FFFE); the first instruction sets the power-up byte so that the Apple will reboot when a RESET is executed, the second instruction executes the RESET cycle.

This is not very user-friendly (type BYE and boom—the system reboots without warning), but will work.

Because the unmodified quit code points the RESET vector at itself, there is no way to force a boot without patching the quit code. Decrementing the power-up byte before calling quit and then pressing reset doesn't work—you merely get that lovely prefix prompt back again. On an Apple II-Plus, which doesn't support the open-apple-control-reset reboot, the only way to start a disk from another operating system is to power down and back up. What a pain in the chips.

AppleWorks lock up

Why does AppleWorks lock up when I delete a report format? It is most annoying and could be anything, as my IIc sports a replaced motherboard (modem upgrade), which I don't trust at all (the outfit that did the upgrade just folded!), and a 512K Z-RAM, which I always suspect of adding its own particular confusion to everything else.

This may be old hat to many AppleWorks users, but here's a little convenience I've found when creating a data base—I create a bunch of extra categories with blank names and just leave them bunched-up down in the bottom right corner of the record—then if I want to add categories at a later date, I can march them out into the layout and open-apple-N(ame) them without having to redo the entire layout and without losing any report formats. This saves a lot of time and keeps things flexible.

Dan Seifert
St. Paul, Minn

I've also had AppleWorks seem to lock-up while deleting a report format. However, I'm now starting to wonder whether I didn't wait long enough. AppleWorks takes an unbearably long time to remove files from an expanded desktop—maybe this applies to report formats, too. Whenever AppleWorks seems to lock up it is probably worthwhile to give it some time—at least five minutes—to see if it is really "just thinking."

I can't think of any good reason not to design all data base files with the full 30 possible categories, just as you suggest. The file will be slightly larger if you do this, but with an expanded desktop the small increase in size will be of little consequence.

AppleWorks page nos. (cont)

According to our sources at Apple, AppleWorks numbers pages from 1 to 255, then starts over from zero. I guess they didn't anticipate the effects of memory expansion. There's no easy trick to fix this other than the obvious: breaking up the document. Apple says that it is aware of the problem and that Rupert Lissner has already fixed it. The fix will appear in their next release...who knows when?

Christopher Van Buren
Q-mar Group
P.O. Box 11215
San Diego, Calif. 92111

*The Q-mar group publishes a monthly newsletter dedicated to AppleWorks called **AppleWorks Exclusive Reference**. Subscriptions are $39.95 in the U.S. and Canada.*

System Utilities bugs

Apple has released the UniDisk 3.5 system utilities package on a 5-1/4 inch disk. It's available, with an updated version of the ProDOS Users Disk on the back, under the name *Apple II Utilities Guide* (A2D2053, $40—*Apple II System Utilities*, A2D2054, is the 3.5 version of this package; it comes with the UniDisk 3.5).

On the /USERS.DISK side of the disk, CONVERT is updated to bypass the MouseText characters. I don't know if any other fixes have been applied. Otherwise, PRODOS, BASIC.SYSTEM and FILER are from the version 1.1 and subsequent 1.1.1 updates. The CHAIN problem has not been fixed.

On the /UTILITIES side of the disk, we have both the IIe and IIc versions of the system utilities package that originated with the IIc. The programs are nice, but very slow. They work with ProDOS, Pascal, DOS 3.2, and DOS 3.3 diskettes.

The COPY function is the most interesting. It allows file copying between these operating systems. Well, anyway, some limited file copying. I have successfully copied a text file from ProDOS to Pascal to DOS 3.3 and back to ProDOS. The file arrived back and matched the file that was sent. There were some small glitches.

When copying the 76-byte file from ProDOS to Pascal, two extra blocks were added (at the end; I have already taken into account the two-block Pascal text header). By the time I got to DOS 3.3, the file had 1,024 extra bytes of zeros tacked onto it (five extra sectors). The opposite direction had more difficulties.

ProDOS to DOS 3.3 went OK, but DOS 3.3 to Pascal had two bugs. The file increased in size from five DOS 3.3 sectors to 6 Pascal blocks (same output size as from ProDOS to Pascal). However, the file name was stored in the Pascal directory with high bits set ("A" as $C1 instead of $41). Thus, a Pascal program could not find the file. In addition, when going from DOS 3.3 to Pascal, 35 bytes of text were chopped off at the beginning of the text. The chopped file arrived back at ProDOS just minus the 35 bytes (no extra blocks).

Bigger problems arise when attempting to convert a Pascal program source file to ProDOS. The utility cannot handle the Pascal editor's special control-codes and occasionally crashes into the Monitor. At other times, the file converts, but loses some data and gets other data records out of order.

The documentation is poor in that it does not describe the file conversion process. It does not indicate what file types can be converted properly. It will not convert a BIN file to Pascal. So what is left?

What I wanted mostly was documentation of the ampersand routines the program uses and which have been discussed in ***Open-Apple***. No such luck.

I have written to Apple about these problems (Tell Apple About, P.O. Box 1143). They also got a big stack of file dumps, block dumps, and directory listings to support the bugs.

My Apple dealer just informed me that MousePaint Version 1.2 is being made available to MousePaint owners. I obtained the update. I had version 1.1 which was dated March 1984. I bought my Mouse in January 1985. Version 1.2 is dated April 1984. Ouch! Double ouch!! What gives? Why does it take Apple Computer two years to put out an updated version that was ready two years ago? This sort of thing is a real downer.

Ken Kashmarek
Eldridge, Iowa

If CONVERT, FILER, and the System Utilities are any example, we can assume Apple did not spend two years in testing the product.

*I have rapidly become addicted to the new ProDOS version of **Copy II Plus** (March, page 2.10) and now use it exclusively for file manipulations. It is vastly superior to Apple's own system software. It can convert any type of file, including random access files, between DOS 3.3 and ProDOS. It doesn't handle Pascal disks, however.*

*Incidentally, according to the April 21 **InfoWorld** (page 14), Apple has taken legal action to prevent further imports of the Laser 128 Apple-compatible computer that Central Point has been selling. Apple claims both copyright and patent infringments.*

Joyful programming

Help! How do I program my Apple IIe to use a joystick instead of a keyboard?

Charles Besari
Ortonville, Mich.

The Apple keyboard and joystick are separate devices. In your own Applesoft programs, you read the keyboard with INPUT or GET. You read a joystick's position using the Applesoft PDL function. The position of a joystick's pushbuttons can be detected by PEEKing:

```
10 X = PDL(0) : REM  X axis (0 to 255)
20 Y = PDL(1) : REM  Y axis (0 to 255)
30 B1 = (PEEK(-16287) > 127) : REM  button 0
40 B2 = (PEEK(-16286) > 127) : REM  button 1
```

B1 and B2 will be set to 0 if the button is up or to 1 if the button is pressed (or, on a II-Plus, if nothing is connected to the game I/O connector).

*Further information is available in **The Applesoft Tutorial** and **The Applesoft Basic Programmer's Reference Manual**, as well as in many other books on Applesoft. There are also ways to do this in machine language and Apple Pascal.*

In commercial programs that are not user-modifiable, you are limited to use of the keyboard unless the program includes a joystick option.

Another fast garbage trick

In the January '85 ***Open-Apple*** (page 4-5) you described a technique for repeatedly poking FRETOP to eliminate garbage collection. While this works well,

it requires some finesse on the part of the programmer to know when to poke FRETOP. If most of the strings the program uses are truly constants then an easier way is available.

First DIM and load all constant string arrays and simple string variables. Execute an X=FRE(0) command, then set HIMEM: equal to the contents of FRETOP like this *HIMEM:PEEK(111) + 256*PEEK(112).* That's all! The garbage collection routine will not move these constant strings, as they will always appear to have already been moved.

Conscientious programmers would put a HIMEM: or MAXFILES command on the first line of their program to recoup memory on multiple RUNs. For example:

```
1    PRINT CHR$(4);"MAXFILES 3" : REM reset HIMEM
10   :
100  DIM BIG$(500) : D$= CHR$(4) : REM constants
110  :
120  REM  read in the permanent strings from disk
                    .
                    .
                    .
160  X = FRE(0) : REM pack in the new strings
170  HIMEM:PEEK(111) + 256*PEEK(112) :
         REM set HIMEM to value in FRETOP
180  REM all strings and string arrays created
         past this point will be collected normally.
```

If you change the value of a string defined before line 170, it will move outside the permanent storage area, so make sure all strings defined before line 170 are strings you do not intend to change.

Frank G. Andrews
Kalamazoo, Mich.

To help people understand how this trick works, let me remind them why the Applesoft garbage collector is slow. For ***each*** *string variable you use, it must look at* ***all*** *string variables once. Thus, the more variables you have, the longer it takes to look through all variables, and the more times it has to be done.*

With this trick, you "hide" your permanent variables so that the garbage collector doesn't make passes for them. When the collector makes passes for your "unhidden" variables, however, it still must look through ***all*** *variables, both the hidden and unhidden ones. This trick works best, then, when you have just a few temporary, unhidden strings.*

The more complicated trick explained in the January '85 ***Open-Apple*** *tries to avoid garbage collection entirely. The trick given here allows collection to occur when necessary, but tries to fix things so that it happens so quickly that no one notices.*

SU2.OBJ bug found

All the correspondence that has appeared in ***Open-Apple*** regarding the SU2.OBJ routine from the System Utilities disk has aroused my curiosity (December, page 96; March, page 2.13-14). Enough, in fact, that I have dug into the routine to figure out how it works, and along the way I've found a few surprises and I've squashed the bug in &INPUT.

My IIc (purchased in late 1985) apparently came with the new version of SU2.OBJ that supports the UniDisk 3.5, even though my IIc itself does not. Although my System Utilities disk sets HIMEM:32256 ($7E00) before BRUNning SU2.OBJ, and the routine itself doesn't change this, it works equally well if you use HIMEM:34560 ($8700) instead. The lowest memory location it uses is $8BD6. Since HIMEM in ProDOS must sit on a page boundry ($xx00), you might think that $8B00 would work. But it doesn't, because ProDOS puts its first file buffer in the four pages starting at HIMEM. Therefore, HIMEM must be set four pages below the lowest address used by the routine. The System Utilities' use of $7E00 implies that everything above $8200 is used by the program. If it is, it isn't because of SU2.OBJ, so your own programs that take advantage of it can use $8700 instead.

Besides &INPUT, &GET, &PRINT, and &EXIT, SU2.OBJ also includes an &HTAB command. If you want to &PRINT something at a screen location other than column 1, you need &HTAB instead of the normal HTAB. Whereas &HTAB works for both PRINT and &PRINT, the normal HTAB does not.

Everything that you &INPUT or &PRINT passes through a 160 character buffer located at the end of SU2.OBJ itself. Code that initializes the routine gets overwritten with the first use of either of these commands. This limits the &INPUTted or &PRINTed stuff to 160 characters. Worst of all, the routine itself doesn't check for overflow, so everything in excess of 160 characters will merrily overwrite the ProDOS code starting at $9A00. The programmer must be very careful not to &PRINT anything longer than the buffer. Apparently &PRINT was designed to print strings that might extend as far as two 80-column lines, but no more. &PRINT's primary purpose is for printing lines that are compatible with both the 40-column and 80-column screens (&PRINT checks what mode your Apple is in and provides word-wrap at the appropriate point).

Now, for that elusive bug in &INPUT—as Christoper Hogue pointed out in the March issue, the first character of the string often gets clobbered. But the affected string isn't the one received by &INPUT, it's the one that was entered just before &INPUT was used! This happens because &INPUT copies one too many characters from its buffer to Applesoft's string storage area. The excess character can be a fill character as defined by FL$ (if the field indicated by fill characters was completely used) or a space (if it was not). Where Hogue's strings appeared to be missing the first character, it was because the first character had been replaced by a space. The very last string entered will always be undamaged because &INPUT hasn't yet had the opportunity to clobber it.

In my version of SU2.OBJ, the code responsible for moving the sting data into the string storage area resides at $974E and looks like this:

```
974E: A0 00     LDY #$00      Get length from string
9750: B1 83     LDA ($83),Y      descriptor
9752: 8D E6 8B  STA $8BE6     Save length in data area
9755: EE E6 8B  INC $8BE6     Add 1 (WHY??!!!)
9758: C8        INY
9759: B1 83     LDA ($83),Y   Get low byte of string
975B: 85 FE     STA $FE          pointer, put at $FE
975D: C8        INY
975E: B1 83     LDA ($83),Y   Get high byte of string
9760: 85 FF     STA $FF          pointer, put at $FF
9762: A0 00     LDY #$00      Start with character 0
9764: CC E6 8B  CPY $8BE6       all copied?
9767: F0 09     BEQ $9772       yes, branch to RTS
9769: B9 60 99  LDA $9960,Y     no, fetch next and
976C: 91 FE     STA ($FE),Y        copy it
976E: C8        INY           Set Y for next char
976F: 4C 64 97  JMP $9764       loop back
9772: 60        RTS
```

The bug occurs at $9755. Increasing the length by 1 makes an extra character get copied from the buffer to the string storage area, overwriting the first character of the previously-defined string. Since the entire buffer initially gets filled with FL$, then gets filled up to position FL with spaces if the entered string was less than FL characters long, this extra character can be a fill character or a space depending on whether or not the entered string filled the field.

To fix the bug requires only that we NOP out (replace with $EA's) the three bytes starting at $9755. To make this permanent requires a little detective work, since the SU2.OBJ code isn't BLOADED to its ultimate operating location but is moved there during initialization. In the version I have, SU2.OBJ loads in at $2000, where it immediately moves code from $203B through $2E64 to locations $8BD6 through $99FF. This means the string-moving subroutine (listed above) begins life at $2BB3, and the offending three bytes are at $2BBA. To fix it, just BLOAD SU2.OBJ, verify that $2BBA.$2BBC contains EE E6 8B, change those bytes to EA EA EA, and BSAVE SU2.OBJ. You don't need to specify the A and L parameters because ProDOS will use the values shown in the directory. If your version doesn't have EE E6 8B at $2BBA, you'll have to go searching for it.

Have fun with SU2.OBJ and be sure to observe the 160 character limit. This routine has no mercy and will gladly send ProDOS off into never-never land if you let it. And thanks to Christopher Hogue for revealing a most useful and interesting routine.

Clay Ruth
Dyer, Ind

In the original IIc version of SU2.OBJ, the three offending bytes appear at $2DAF-$2DB1.

Open-Apple

[redacted]

is written, edited, published, and

© Copyright 1986 by
Tom Weishaar

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

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

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

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

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

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

Source Mail: TCF238 CompuServe: 70120,202