Open-Apple

Releasing the power to everyone.

March 1988
Vol. 4, No. 2

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

A month of firsts

Late January and early February saw a number of firsts for the Apple II kingdom. The first finished 16-bit Basic for the IIgs appeared, the first serious "desktop publishing" program for the Apple II arrived at our offices, and the first-ever ads for AppleWorks were published.

AC/BASIC, the 16-bit Basic for the IIgs, comes from a company called Absoft ($125 plus $4 shipping, 2781 Bond St., Auburn Hills, MI 48057 313-853-0050). *AC/BASIC* is a compiled Basic. Its biggest strength is that you can sit down with it and start programming windows, menu bars, dialog boxes, and buttons without first reading 70 pounds of IIgs manuals.

A second strength is that the syntax of *AC/BASIC* is almost totally compatible with that of Microsoft's Basic compilers (and interpreters) for the Macintosh and MS-DOS (Absoft actually wrote the Macintosh Basic compiler that Microsoft sells). It's also compatible with Absoft's own *AC/BASIC* for the Commodore Amiga. *Microsoft Basic* programs originally written for the Macintosh can be compiled with *AC/BASIC* and run on the IIgs with few or no changes.

At least five other companies are still trying to get 16-bit Basics for the IIgs to market. They include:

- Apple itself with *IIgs Basic*, an interpreted Basic, derived from the Apple III's *Business Basic*, that is available now in an unfinished, beta version from the Apple Programmers and Developers Association (see November, page 3.80);
- TML Systems with *TML Basic*, a compiled Basic that's to be syntax compatible with Apple's *IIgs Basic;*
- Zedcor with *ZBasic*, a compiled version that will be syntax compatible with the *ZBasic* compilers for DOS 3.3, ProDOS, CP/M, the Macintosh, and MS-DOS;
- The Byte Works with *Orca Basic*, which will be available in both interpreted and compiled versions and which Byte Works calls "an extended version of Applesoft;" and
- Micol Systems with a IIgs version of its ProDOS-based *Micol Basic* compiler.

Based on our preliminary examinations, it appears *AC/BASIC* would be an easy language for Applesoft programmers to migrate to. Programs can be written with any editor that can use standard text files. *AC/BASIC* comes with a modified version of the Apple Programmer's Workshop editor. To compile a program, you simply run the compiler, pick the file you want compiled, change any compiler options you want to change, and push a button.

AC/BASIC doesn't require line numbers. It supports the IIgs tool box and assembly language subroutines. However, it doesn't support the "object file formats" of the Apple Programmer's Workshop (for the few of you who know what that means). Something else it doesn't support is the text screen, although its own editor is text-screen-based.

It does include its own commands that provide direct access to IIgs graphics, such as CIRCLE and LINE; to IIgs sound, such as SOUND and WAVE; and to the IIgs user interface, such as MOUSE, MENU, DIALOG, and BUTTON. GEnie users Alan Hoffman and Marian Petrides report that setting up programs that use the super-high-resolution screen, windows, and pull-down menus is simple with *AC/BASIC*.

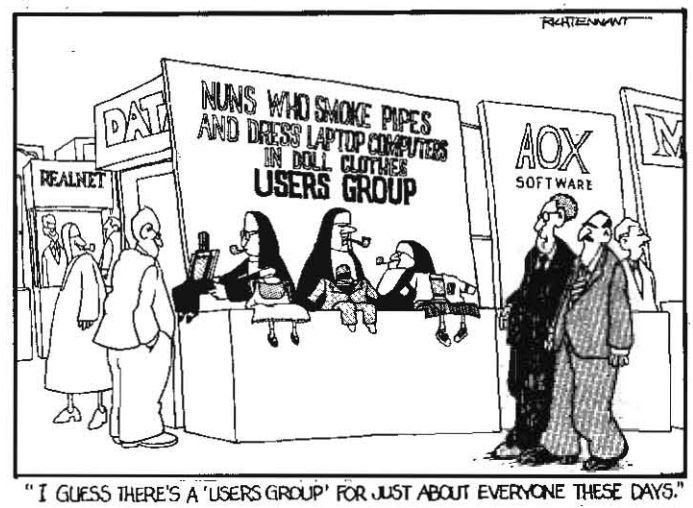
Program size and data space are limited only by the amount of RAM you have. Interestingly, *AC/BASIC* does not use Apple's SANE math toolbox but instead uses Absoft's own binary and binary-coded-decimal math package. Stand-alone, self-running programs can be compiled and distributed without paying a licensing fee. Like all serious software, *AC/BASIC* isn't copy-protected.

***Publish It!* is the name of the first serious desktop publishing** program for the Apple II–it's not copy protected either ($99.95, Timeworks, 444 Lake Cook Rd., Deerfield, IL 60015 312-948-9202). It uses double-high-resolution graphics and, consequently, requires a 128K IIe or a IIc or IIgs. It features the Apple Interface; what-you-see-is-what-you-get; 9 point to 72 point fonts with adjustable kerning and leading; column-to-column and page-to-page text flow; page views at actual size, double size, half size, or full page; importing of AppleWorks word processor files and standard text files; importing, cropping, and sizing of any standard double-high-resolution graphic; built-in word processing and a built-in graphic toolbox; and high-quality dot-matrix print outs.

What distinguishes its Macintosh desktop publishing brethren from *Publish It!* is PostScript. *Publish It!*, like the other Apple II desktop publishing packages upcoming in 1988, doesn't support PostScript, which is a "page description language." Macintosh desktop publishing programs use PostScript to tell Apple's LaserWriter printers (except for a new cheap one) and state-of-the-art typesetting machines what to print.

Instead of PostScript commands, what *Publish It!* sends to your printer is a series of dots. *Publish It!* figures out exactly which "dots" on a blank sheet of paper need to be inked to create the document you want to see. *Publish It!* uses dots that are 1/72nd of an inch high by 1/120th of an inch wide–8,640 dots per square inch or about 800,000 dots per page.

Once it figures out which dots should be inked and which should be left as is, *Publish It!* has to tell all this to the printer. Sending information on 800,000 dots takes a little less than two minutes at 9600 baud, the typical speed at which such stuff is sent to printers in the Apple II world (actually, the computer-to-printer connection is the fastest part of all this–it usually takes longer than two minutes to print a page because the computer can't calculate which dots should be inked and the printer can't actually ink them that fast).



"I GUESS THERE'S A 'USERS GROUP' FOR JUST ABOUT EVERYONE THESE DAYS."

Apple's LaserWriter printers can print dots 1/300th of an inch high by 1/300th of an inch wide. This is 90,000 dots per square inch or about 8 million dots per page. Sending information on 8 million dots takes about 17 minutes at 9600 baud. Obviously, this presents a big problem–not many people are willing to wait 17 minutes per page for printer output.

The solution to the problem is not to send dots, but to send commands, in terms more general than dot-by-dot, that indicate what to ink and what to leave blank. PostScript is a "language" for doing exactly that. Under PostScript, the desktop publishing program doesn't worry about dots. It tells the printer what letters to print, how big to make them, where to put lines, where to put curves, and so on. By interpreting these commands, a PostScript printer can develop an image at the maximum resolution that it's capable of. This maximum resolution can be anything–some typesetters have dots 1/1200th of an inch square (at 9600 baud, sending a page of dots this small would take four-and-a-half hours).

Timeworks sells an accessory pack for *Publish It!* that allows it to use laser printers, but, since *Publish It!* doesn't support PostScript, it has to send dots to the laser printer. And the resolution of these dots is always 72 by 120. For personal documents, 72 by 120 resolution at the price of an Apple II, *Publish It!*, and a dot-matrix printer is very good. But don't get your hopes too high–it's not, for example, good enough for *Publish It!*'s own magazine ads.

If you examine a dealer's copy of the program, be sure to print something out and look at it. Text on the screen is much "blockier" looking than the same text on paper. If *Publish It!* had as many usable fonts as *Printrix* (September, pages 3.59-61), the output quality would be equivalent. Being able to see what you're doing and to get full-page, no paste-up print outs is well worth the extra $34. I tested *Publish It!* on a IIgs and was impressed with its speed. If you're interested in desktop publishing on the II and you're not interested in copy protection, this program is the one to get.

Let's give Claris a hand for introducing itself in the Apple II magazines with three-page ads about AppleWorks–the first AppleWorks ads ever published. "The best-selling Apple software just got the ultimate upgrade. Its own company," the ads are headlined.

I did squirm a little bit when I read the line in the ad that goes, "And here we are, while the halls are still filled with the scent of new carpet, introducing a more powerful version of the best-selling Apple program in history" (Apple had been selling this version of AppleWorks for 16 months already). But hey, this is be-nice-to-Claris-month.

Claris apparently intends to base its reputation on customer support. Everyone I've talked to who has called the Claris technical support line has come away impressed by the technical staff's knowledge of AppleWorks. The technical support number is 415-962-0371. Hours are 6 AM to 8 PM (Pacific time) Monday through Thursday, and 6 AM to 4 PM and 5:30 PM to 8 PM on Fridays.

Claris also has a customer relations number for questions that aren't technical. It's 415-962-8946. Hours are 8 AM to 5 PM Monday through Thursday and 8 AM to 3 PM on Fridays.

I've talked to the AppleWorks Product Manager at Claris, Alison Elliott, and she really is interested in what kinds of enhancements those of us who actually use AppleWorks every day would like to see. I'm sending her a letter and I encourage anyone else who feels strongly about AppleWorks to do so, too (440 Clyde Ave., Mountain View, CA 94043).

However, it's clear that the executive officers at Claris still have the delusion that they are running a Macintosh software company. After a year or two of competing with every other well-funded software company in the world, I predict they'll come running back to the Apple II market. It's the Apple II that made Apple, and now Claris, a major software company. I just don't understand why it's us *customers* who have to keep explaining it to them.

***A+* recently completed a study of new IIgs buyers.** You know, the computer Apple markets to kids. Here's a few facts about these kids–77 per cent are between 30 and 54 years old; 46 per cent have completed some post-graduate study; half have annual household incomes of $50,000 or more and a quarter have incomes of $75,000 or more. No less than 76 per cent have owned microcomputers before; 57 per cent have owned Apple IIs. Of those who are employed, 80 per cent use a computer at work.

Although Apple absolutely refuses to market the Apple II as a business machine, 96 per cent of the new IIgs buyers say they'll use their computer for word processing, 70 per cent for data base management, and 61 per cent for spreadsheet analysis. By comparison, only 50 per cent say they'll use the IIgs for children's education, 49 per cent for personal education, and 37 per cent for household management.

Does anyone out there know how to contact Paul Lutus, author of *Apple Writer*? Since Apple is no longer in the software business, it has stopped selling this word processor. Claris, for reasons known only to its management, decided not to publish it either. The way these things usually work is that the author gets the rights to the program back when a work goes "out of print." I suspect there are a number of software companies that would be willing to risk publishing a few thousand copies of a IIgs-compatible *Apple Writer*, if they could just find Lutus and get the rights to the program.

***HyperCard* for the Apple II** will have to come from third-party developers, John Sculley, Apple's president, said at a developer dinner at last fall's Applefest. I ridiculed this idea in November (page 3.74), but the odds are growing (very slowly) that I may be the fool after all. Owl International, developers of a Macintosh *HyperCard*-like program called *Guide*, is getting out of the Macintosh market (they can't compete with free software") and is writing a new product for the IBM PC code-named *William Tell*, according to *The MACazine* (October 1987, page 111). "Further plans for *William Tell* include versions that run on the Amiga, Atari ST, and Apple IIgs." Let's hope not in that order.

But maybe the last were first all the time and nobody knew it. David Thornburg's "Learning Curve" column in the March issue of *A+* is all about a *HyperCard*-like program called *Tutor-Tech*. It's been available for 128K, mouse-equipped Apple IIs for two years already ($195, Techware Inc., P.O. Box 1085, Altamonte Springs, FL 32715 305-834-3431).

This product was designed to be an authoring system, but "on the surface...*Tutor-Tech* looks exactly like *HyperCard*–or, to be more accurate, *HyperCard* looks just like *Tutor-Tech*." According to Thornburg, the program provides a menu-driven system for creating frames of text and graphics. You can also put buttons or text-response fields on the frames. As with *HyperCard*, buttons have a destination frame that is loaded when you click the button. The text-response field, on the other hand, leads to one of two frames, depending on whether the response that's typed in is what the teacher/programmer said it should be. Buttons can also increase or decrease student's scores.

"Because *HyperCard* works with a full-blown programming language (HyperTalk), you can use it to build quite sophisticated applications. *Tutor-Tech*, on the other hand, is limited to the application type that is simplest to create in *HyperCard*: collections of frames you can link with buttons.

"Both products can link to frames within a given set as well as to other sets of frames. A single *Tutor-Tech* lesson can be 250 pages long (provided you don't run out of RAM first). *HyperCard* documents are disk-resident, so the size limitation depends on the disk rather than the available RAM.

"Some people may think that *HyperCard* is a rip-off of *Tutor-Tech*, but I sincerely doubt it," Thornburg says. "Even though Apple's booth was located a few feet away from Techware's at the January '86 Florida conference (where *Tutor-Tech* was introduced), I've yet to find an Apple employee who knows about the product.

"I find it amazing that a product that shows off the Apple II as beautifully as *Tutor-Tech* does has gone virtually unnoticed in the press....*HyperCard* was on the cover of a lot of magazines last year and, as far as achievements go, I think *Tutor-Tech* is just as newsworthy. Success in the educational-software field is hard enough to obtain. If the only programs that get attention are the ones with massive ad and PR underpinnings, we will drive the small, talented companies out of business."

Well said, David....and as far as this small piece of the press is concerned, thank you for not getting mad at me for quoting so much of your article.

I've been reading a lot of good stuff about a $99 device from Orange Micro called the Grappler C/Mac/GS. It makes a variety of less-expensive parallel printers look like serial ImageWriters. In other words, not only does it convert your IIc/IIgs serial signal into parallel, it also converts all the ImageWriter codes into those appropriate for

the printer its connected to. Supported printers include the Epson LX, EX, LQ, JX, MX, RX, FX80, FX100, FX Plus; Okidata 192, 193, 292; Star SD, SG, SR, 10X; and C. Itoh 310XPR (in Epson mode). It plugs into the parallel connector on the back of your printer and comes with a software program on disk that allows you to do IIgs screen dumps. Since most ProDOS 16 software expects you to own an ImageWriter, this sounds like an easy way to get a less-expensive printer while avoiding compatibility hassles. (Orange Micro, 1400 N Lakeview Ave, Anaheim CA 92807 714-779-2772.)

Orange Micro also makes a IIgs memory card called the RamPack 4 GS. It uses 256 x 4 one-megabyte chips. RamPack owners and GEnie users Gary Antoine and Keith Jaskiewicz report that these chips are nearly impossible to obtain, even from Orange Micro–which is probably a good reason to avoid this card.

Apple released a new MIDI interface for the IIgs and Macintosh in January. MIDI is a standard originally developed for connecting electronic musical instruments together. It also works for connecting electronic instruments to computers. Apple's new interface is a small box that goes outside the computer (it's *not* slot-based). A cable goes from one of the IIgs serial ports and plugs into the box. The other side of the box has outlets for two MIDI cables, one incoming, one outgoing. It requires no power supply. Full retail is $99 at your local Apple dealer, cables included.

Unfortunately, according to Apple's press release on this interface, it's illegal for professional or amateur musicians to use it on an Apple IIgs. The press release makes it quite clear that MIDI on the IIgs may be used only for "music education both in school and at home." If the device determines it is being used for non-educational purposes on a IIgs it self-destructs. I sure hope the Macintosh is able to stand on its own two feet soon so that those of us who prefer the Apple II can return to using it professionally without worrying, as Apple does, that doing so will cause the Mac to sink beneath the waves.

CD-ROM engineers and product managers from Apple are going to bring third-party developers up to date on CD-ROM development technologies for the Macintosh and Apple II at a special conference on Friday, March 4. I'm invited, but I can't make it. If *you* go, I'd appreciate a three-paragraph report on what *you* think of CD-ROM for our next issue.

The next Applefest will be held in Boston, May 20 to 22. Call 800-262-FEST for more information.

Three Apple gotchas. If you've ever been puzzled by which way the write-protect tab on a 3.5 inch disk goes to indicate write-protect, it's probably because the *Apple 3.5 Drive Owner's Guide* has it backwards (page 18). The *UniDisk 3.5 Owner's Manual*, on the other hand, has it right in five different languages.

Many Apple dealers will be happy to sell you an Apple 40 megabyte SCSI hard disk at the suggested retail price of $2,208. Few know enough to warn you, however, that 8 megabytes of the drive will be wasted because the largest device ProDOS can support is 32 megabytes. And Apple doesn't have any "partitioning" system that would allow dividing the 40 meg drive into two 20 meg volumes. This month ***Open-Apple*** bought a 60 meg SCSI hard drive made by CMS, on the other hand, that can be partitioned into two 30 megabyte volumes. Not only does CMS provide a way to split the drive, it also provides ways for two or three different computers to share the disk–each with their own area for writing and shared areas for reading. If this sounds incredible so far, get this–it costs half as much as Apple's 40 meg drive and it's quieter than a IIgs with a fan (CMS Enhancements Inc, 1372 Valencia Ave, Tustin, CA 92680 714-259-9555).

With our new CMS drive we got a copy of version 1.1.1 of Apple's *Backup II* program for backing up ProDOS hard drives. We were glad we did, because *Backup II* version 1.1 works only with versions of ProDOS 8 prior to 1.2. Even though Apple's *Tech Tidbits* recently had a question about this and answered that *Backup II* would work with any version of ProDOS 8, they obviously didn't try to combine the older *Backup II* with newer versions of ProDOS. That combination habitually crashes into the Monitor.

In addition to our CMS hard drive, we've also been experimenting with an AppleTalk network. We bought one of Apple's $139 AppleTalk cards for one of our ImageWriter IIs and we bought four PhoneNET PLUS AppleTalk connectors. One connector was for the ImageWriter and the other three were for IIgs printer ports. Each connector is a small box with a short wire coming out of it that plugs into the mini-8 jacks on the IIgs and the ImageWriter.

On the other end of the box are two standard RJ11 modular telephone jacks. You take telephone wires and connect all the boxes together in a string. For the ends of the string, where there is an incoming telephone wire but no outgoing wire, you get a little RJ11-mounted resistor to plug into the spare jack. This makes an AppleTalk network with three computers able to share one printer.

The PhoneNET connectors are made by a company called Farallon Computing (2150 Kittredge, Berkeley, CA 94704 415-849-2331) and cost about $65 each. Apple itself makes similar connectors called, until a few weeks ago, AppleTalk connectors. Suddenly, however, they are called LocalTalk connectors. Besides being about $10 more expensive per connector, LocalTalk also requires a special cable that Apple sells for $5 a meter. Since PhoneNET can use existing telephone wiring (see July 1987, page 3.43, for a technical explanation of how AppleTalk and a phone could use the same telephone cable) it is *much* cheaper. Geez, even if you have to go down to Radio Shack and buy a spool of phone wire it's still *much* cheaper.

Once you have all the wires connected together, get yourself a new disk, format it as /APPLETALK, and copy the following files from your IIgs /SYSTEM.DISK into /APPLETALK's main directory:

```
/SYSTEM.DISK/SYSTEM/P8
/SYSTEM.DISK/SYSTEM/SYSTEM.SETUP/ATINIT
/SYSTEM.DISK/APPLETALK/all files

rename P8, PRODOS
rename CHOOSER.II, CHOOSER.SYSTEM
```

Now boot this disk. The program formerly called CHOOSER.II will start up automatically. This little jewel then says hi to all its friends on the network and asks them what their names are (there is another program in the IIgs AppleTalk folder called NAMER.II that you can use to change the default names of the machines on the network; if you have a network big enough to require this, try it and tell us how it works). Next CHOOSER gives you a list of all the printers on the network and asks you which one you want to use. In our case there was only one to pick from, so this part was easy.

Next CHOOSER writes the name of the printer you picked in the file called ATINIT. A(pple)T(alk)INIT(ializer) is a file that has to be executed every time you reboot. It initializes the AppleTalk port on the IIgs and tells it what printer you will be using. Versions 1.2 and later of ProDOS 8 and all versions of ProDOS 16 will execute ATINIT when they start up. Under ProDOS 8, however, ATINIT can't be inside a folder, it must be in the root directory. Under ProDOS 16, on the other hand, ATINIT should be in the /SYSTEM.DISK/SYSTEM/SYSTEM.SETUP folder.

Now that you've run CHOOSER and have ATINIT prepared with the default name of your printer, copy ATINIT into the SYSTEM.SETUP folder of all the ProDOS 16 disks you intend to boot and into the main directory of all the ProDOS 8 disks you intend to boot. You've already updated your ProDOS 8 disks to ProDOS 1.4, right?

Now, the only other thing you have to do to print on the networked printer is to tell your programs that your printer is now in slot 7, not slot 1.

So far, what we have here should be called PrinterTalk. It's nothing more than a system whereby several computers can share one or more printers. If you have more than one printer on your network, you have to run CHOOSER to switch between them.

Apple hasn't documented what exactly it is that CHOOSER does. Theoretically, you could have one computer and three printers on an AppleTalk network, with the computer automatically switching output to the different printers (one with continuous invoices, one with continuous letterhead, and one with continuous blank checks, for example), but Apple just hasn't told us how to do this from inside a program.

In addition to suffering from lack of documentation, IIgs AppleTalk suffers from a lack of following Apple's own protocols. AppleTalk is assigned to slot 7 in the IIgs, yet it saves data in the screenholes for slot 1 or 2, depending on which serial connector you're using for the AppleTalk connector. This means AppleTalk effectively takes up two slots. This is an abominiation. Didn't anybody tell the AppleTalk engineers about the IIgs memory manager? What in the world are they doing messing with memory assigned to slots 1 and 2 ?

Expectations are high that Apple will announce a more advanced form of AppleTalk for the Apple II at AppleFest. It is expected that this version will support disk drives as well as printers. However, don't expect this to be the kind of network that schools have been hoping for to download software to students in a 30-computer lab. Based on what I've seen and heard of AppleTalk, it will be entirely too slow for that application.

Living Legends Software, the shareware cooperative, has moved to P.O. Box 4313, La Mesa, CA 92044 714-676-1940. A new version of Jerry Hewett's *Hyper.FORMAT* (source code you can put in your own programs that will format disks), which supports all ProDOS devices and includes optional detection and lock-out of bad blocks, is available from Living Legends.

Lee Hayward, the guy primarily behind TAWUG (The AppleWorks User Group) for the last couple of years, isn't anymore. You can reach him at RAW.APPLE, P.O. Box 24146, Denver CO 80224. TAWUG itself has moved to P.O. Box 37313, Denver CO 80237. For more on what these groups have to offer, send each a stamped, self-addressed business-size envelope.

The Software Publishers Association released a little pamphlet in honor of Computer Learning Month called "Everything you need to know (but were afraid to ask your kids) about computer learning." It's a well done introduction to computers for total novices. However, one important question is missing from the section on "How to Select Software: Questions You May Want to Ask." You may want to add it to any copies of this pamphlet your friends have. At the top of their list of questions should be, "Is the software copy-protected?" My own kids have had both *Where in the U.S.A. is Carmen Sandiego?* and *Walt Disney Card & Party Shop* give up booting on them in the last 30 days. If I was a total novice, I'd probably throw the whole system in the closet before I'd throw more money away on software.

Crossing telephones with computers

With the exception of our April 1987 issue, in which I put the Binary II file format up on the examining table, I haven't talked much about using an Apple II for data communications here in ***Open-Apple***. Because of our new GEnie alliance, however, this deficiency is like having a pimple on my nose. To cover up the imperfection, this month I'll acquaint you with the benefits and problems that result when you connect your Apple II to a telephone jack.

Everyone is familiar with the world-wide telephone network. It's a random-access real-time point-to-point medium that transmits sounds from where you are to just about anywhere else in the world. Most people use it every day for *verbal* communications.

By accessing the network with your computer instead of a telephone receiver, however, you can also use the network for *written* communications. These written communications can take the form of mail or messages, they can be computer files or programs, they can be encoded graphic images, or (oh perversity) they can even be encoded sounds. Any information you can store in a file on your computer can be transmitted to another computer by using the world-wide telephone network.

In order to hook your computer to the network you need a device called a modem. In order to control the modem and access the network you'll need some "communications" software. Finally, in order to get any value out of the network, modem, and software, you'll need the phone numbers of some friendly computers that will respond to your call.

Five kinds of friendly computers. There are five basic types of phone-answering computers. First, many communications packages can be set up to answer the phone and turn control of your computer and its online disk files over to anyone who calls. If you were an associate professor writing a paper with a colleague in the next state, for example, your colleague might call to ask you to set up your computer so that she could call back and "download" the section of the paper you had just edited or the data you had just analyzed. This first kind of friendly computer usually answers the phone only by special arrangement.

The second group of friendly computers consists of "Bulletin Board Systems." These computers typically answer the phone 24 hours a day and are typically operated by people whose primary motivation is fun. However, some BBSs exist to provide product support for customers, some exist to provide income to the operator, and some exist for other reasons. Some systems are public, some are private, and most cost nothing to access.

The third group of friendly computers consists of private mainframe systems. Most institutional mainframes are connected to the public telephone network. If you work with a university or company mainframe, in all likelihood you can access it with your Apple from the comfort of your home or office–you don't have to go to the computer center unless the mainframe isn't accessible through the public phone network.

The fourth group of friendly computers consists of consumer information services. GEnie, CompuServe, The Source, and Delphi are a few names that spring to mind. These services are mainframe-based, but are running special software to make them easier to access than the typical private mainframe. They offer information and entertainment of interest to just about anyone. They are public systems. They charge fees based on how long you are connected to their mainframe. The fees start at about $5 an hour and go up from there. And they're often higher during the day than at night or on weekends. (The mainframes that these information services use spend their daylight hours working for big business.)

The fifth group of friendly computers consists of professional information services. DIALOG, BRS, NEXIS, VR/TEXT, and the Dow Jones News/Retrieval Service are a few names that spring to mind. These are public systems much like the consumer information services in the fourth group, except that they offer information of interest primarily to professional audiences, they emphasize software that can *search* for specific pieces of information, and they tend to charge higher prices.

An observation you might find helpful about these five groups is that the first three tend to connect directly to the local telephone network. If you're in Boston and you want to call a BBS in San Diego, you'll have to make a long distance call to the BBS's San Diego phone number and pay the long distance charges.

The consumer and professional information services, on the other hand, tend to connect to national or international "data" networks (as opposed to the regular "voice" networks) that are, in turn, connected to *many* local telephone systems. For example, if you're in Boston and you want to call GEnie, you make a local call to the Boston "node" of General Electric's data network. The cost of using the network is included in the cost of using the service. There are several different data networks in the U.S. Like GE, CompuServe has its own network. Two other names that spring to mind are Telenet and TYMNET.

Turn your Apple into a terminal. Before you can actually "log on" to any of these friendly computers, you'll need some communications software for your Apple II that turns it into a "communications terminal." At a minimum, this software has to make your Apple act like a Teletype (TTY) machine. This means it has to be able to send the remote computer anything you type on your keyboard and it has to be able to display on your screen anything the remote computer sends to you.

A real Teletype machine has an important advantage, however, over terminals that use display screens. Everything received by a Teletype is immediately printed out on paper. Everything received by your Apple, on the other hand, will immediately scroll off the top of the screen. We'll solve this problem in a moment.

At this minimum level of ability, your Apple can also be called a "dumb terminal." This is computer talk for a screen and keyboard hooked to a computer. Sometimes the screen-keyboard combination is also called a "console." What's dumb about a dumb terminal is that it can only display what the computer tells it to. It has no capacity to capture anything the computer sends to it. And it has no capacity to send anything to the computer other than what you type on the keyboard.

An "intelligent" or "smart" terminal, on the other hand, does have at least some of these abilities. And your Apple II can easily be turned into an intelligent terminal by the right software package. Such software will let you capture the ASCII character stream the remote computer sends to you and save it in a disk file or print it out on paper. Such software will also let you type very rapidly on your keyboard (for all the remote computer knows) by sending a text file from your disk.

all the remote computer knows) by sending a text file from your disk.

Such software can also give you the ability to send ("upload") or receive ("download") non-text files to or from a remote computer using one or another of several special protocols that check for errors. The most popular of these protocols is called XMODEM. XMODEM transfers files in 128-byte blocks. The sending computer calculates a check sum on the block and sends it along, too. The receiving computer also calculates the check sum and if it doesn't match the one received from the sending computer, the receiver tells the sender to send the block again.

Transmission errors or "line hits" are a fairly common problem when using the public telephone network. A string of garbage characters in the middle of the character stream you are receiving may indicate that lightening just struck in Memphis. XMODEM was developed to keep these kinds of errors out of files.

Some Apple II communications packages also support a protocol called Binary II, which was mentioned at the beginning of this article. Binary II provides a standard way for the directory information about ProDOS files (filename, file type, aux type, creation date, and so on) to be embedded in a special 128-byte block at the beginning of the file. Binary II allows ProDOS files to be sent to a remote computer and stored there as standard binary files, then downloaded and converted back into ProDOS files with all the right directory information.

XMODEM and Binary II don't overlap. They are two different protocols that do different things, both of which are good for us.

In addition to the ability to capture and send text files and to do protocol transfers of any type of files, most communications packages also support macros. Macros allow you to enter complicated sequences of characters with one or two keystrokes. Communications macros are able to wait until a specific character string comes in from the remote computer before responding, consequently, they can be used to automatically send things like account numbers, passwords, and command sequences. I use a long macro "script" every morning before I eat breakfast that logs on to GEnie for me, captures my mail, captures all the new bulletin board messages in the Apple II areas, and then logs off.

Some communications packages allow you to edit text. This can be helpful for composing messages while online. Others don't allow editing, but will allow you to review what has been captured.

One other feature offered by some packages is "terminal emulation." Dumb terminals have been manufactured for a long time by a number of different computer companies, and, as usually happens, at one time or another most of these companies decided to add "advanced features" to their terminals. These advanced features have included things like giving certain ASCII control-codes the ability to erase the screen or position the cursor.

The problem with the advanced features is that each manufacturer made up its own control codes and each company's codes are different. Consequently, even today, most *public* computer systems assume that you are using a generic TTY machine. *Private* systems, on the other hand, such as the mainframe at your university or business, may assume everyone calling in will be using a specific brand of terminal that always responds to an incoming "control-C" by clearing the screen and that sends out an "escape 5*" when someone presses the key marked "DELETE."

Many of the communication software packages you can buy for the Apple II are able to emulate various terminals. However, if you intend to access any private systems, find out *exactly* which terminal the private system you want to use expects you to have–before you buy–then find a software package that can emulate that specific terminal.

A few years ago a package called *ASCII Express* ($129.95) dominated the Apple II communications market, but nowadays the successor to *ASCII Express*, *MouseTalk* ($99.95), shares the market with Pinpoint's *Point-to-Point* ($129, written by ***Open-Apple*** subscriber Gary Little, who was recently named editor of *A+*), Checkmate Technology's *ProTerm* ($95), PBI's *CommWorks* ($95), Softronics' *Softerm 2* ($195, pricey, but very powerful–particularly for terminal emulation), and several other communications packages. The most recent comparison review I can find ran in the December 1987 *A+* (there's that name again, hey Sally, did they buy an ad from us this month or something?), pages 65-72. The November 1987 *inCider*, pages 48-57, gives addresses for the publishers of all the products mentioned here, but doesn't actually review any of them.

At the moment I'm using, and am very pleased with, a "shareware" communications program called *Talk is Cheap*, by Don Elton ($30, 3207 Berkeley Forest Drive, Columbia, SC 29209). In the CP/M, MS-DOS, and Macintosh markets, communications software has always been dominated by shareware (software that may be freely copied, distributed, and tried out; those who decide to use the software send a fee to the author under the honor system).

Talk is Cheap can capture and send text files and it supports XMODEM, as well as another protocol called YMODEM (and it can add Cyclic Redundancy Checking to either). It can automatically recognize and reconstruct Binary II files. It has the most powerful communications macros I've seen in an Apple II program. And it can emulate a variety of terminals. Applied Engineering licensed a specially modified version of *Talk is Cheap* from Elton to distribute free with its modem.

Who is this Hayes guy, anyhow? Having a computer to call and a good communications package to call it with won't do much good until you have a modem. Rapidly skipping past all the gobbledygook, a modem is an electronic device that allows you to connect your computer to the telephone network. In the Apple II kingdom there are two fundamental types of modems and a third type called "all other." Unfortunately, the two fundamental types go by the names "Hayes compatible" and "Hayes compatible." Mr. D.C. Hayes, it turns out, has developed two generations of precedent-setting modems.

The first generation was based on a Hayes device called the "Micromodem II." This generation was developed way back in the dim dark ages of the Apple II and II-Plus, before Apple taught us to capitalize the second half of newly-coined trade names. This modem was an "internal" modem–that is, it slipped into a slot in the Apple. A ribbon cable attached to the modem came out the back of the Apple and plugged into a box, about the size of a cassette tape box, called a "microcoupler." You could plug a phone cord into a standard modular jack on the microcoupler.

Back in its day, the Micromodem II was a remarkable device. At the time, Fortune 500 companies were still using devices called "acoustic couplers" to hook modems to the phone network. You actually took the handset from your telephone and stuck the mouthpiece and the earpiece into rubber cups on the acoustic coupler. The modem would then use the acoustic coupler to whistle into the telephone mouthpiece and would use the telephone earpiece to hear the whistles coming from the other modem. Plugging a non-AT&T modem right into the telephone network was on the cutting edge of technology when the Micromodem made its debut.

In addition to hooking directly to the phone network, the Micromodem II was special because it had dumb terminal software built right into it. Its manual recommended that it be installed in slot 3. Then a simple IN#3 from the keyboard, followed by a control-A and control-F, would turn on the terminal software. After that, using other control-codes, you could dial the number of a big mainframe and appear to the mainframe to be a simple, dumb, Teletype machine.

Before long, however, software developers had released Apple II software that allowed the Apple to capture the incoming text and save it to a disk. The same programs could read files and type them into the remote computer. Suddenly meek little microcomputers were able to use their intelligence over the phone and the days of the Teletype machine were numbered.

(The release of Apple Pascal, incidentally, did its own number on the time that internal modems would spend in slot 3. Pascal expected an external terminal in slot 3 and would lock up tight if you had a Hayes modem there. Slot 2 soon became the "standard" slot for modems, and slot 3's external terminal soon became an 80-column card.)

Hayes still sells an updated version of its original modem called the "Micromodem IIe." Other companies make clones of the Micromodem. What's confusing is that sometimes people call these modems "Hayes compatible." True Hayes compatibility, however, has come to mean something entirely different, as we'll see in a moment. Modems of this first generation, or first fundamental type, are better referred to as "Micromodem II-compatible."

In addition to Hayes, a number of other companies have made

slot-resident, internal modems for the Apple II over the years. Most of these belong to the "all other" compatibility (or should I say "non-compatibility") group. A few of them, however, such as Applied Engineering's DataLink, are true Hayes compatibles. Others, as mentioned earlier, are Micromodem II compatibles. Thus, the fact that a modem goes in a slot tells you absolutely nothing about which compatibility standard, if any, it follows.

Most of today's "external" modems, on the other hand, are compatible with Hayes' second modem generation. This isn't *necessarily* true, but almost all communications software on the market today is designed to work with modems compatible with this second Hayes generation; thus, non-compatible modems haven't been well received in the marketplace and have nearly ceased to exist.

An external modem usually looks like a small flat box and sits on your desk, under your phone, or hangs from your power outlet. This kind of modem has one plug (sometimes two) for a phone cord and another plug for a cable going to your computer's serial port.

Unlike Micromodem II-compatibles, external modems have no direct connection to the computer. If you want to send commands to an external modem, such as instructions to dial or answer the phone, you have no tools at your disposal other than the standard ASCII character set–the same bunch of characters you want to transmit and receive.

What the "Hayes protocol" is all about is nothing more than a standard set of modem commands that can be embedded in ASCII character streams. We've talked about this technique for controlling devices connected to computers–and its limitations–many times in the past (most recently in our series on serial card standards–see, for example, "Command syntax," in November 1987, page 3.75).

Under the "Hayes protocol," modems that are "offline" (or *not* active on the telephone network) wait for the sequence "control-M (same thing as a Return) AT". The "AT" stands for "attention." You may sometimes see the Hayes protocol referred to as the "AT command set." The most-frequently used command is probably the one to dial the phone "ATDnnn-nnnn" where "nnn-nnnn" is the phone number you want. Add a "T" after the "D" for touch-tone dialing (ATDTnnn-nnnn). Use commas to insert pauses between numbers to be dialed.

Since "control-M AT" is a string that could easily appear in the character stream you are sending to another computer, the Hayes protocol requires an additional code when you are "online" (connected to the phone network). This code consists of a one-second delay, followed by "+++", followed by another one-second delay, followed by "AT". An example of a modem command you'd want to send while online is the command to disconnect or hang up. It goes: "(one-second pause) +++ (one-second pause) ATH". Most communication programs have a command that will send this sequence for you.

In addition to supporting standard commands sent from a computer, a Hayes compatible modem also passes standard characters back to the computer to report what's going on. For example, the string "CONNECT 1200" is what a Hayes compatible modem sends to a computer to tell it that a 1200 baud connection has been established with a remote system.

"Baud" or "baud rate" has to do with how fast a modem can send or receive data and is roughly equivalent to words-per-minute. The early Hayes Micromodem was able to transmit and receive at two speeds, 110 baud and 300 baud. 300 baud is about as fast as the average person reads. However, most modems sold during the last couple of years have had a maximum speed of 1200 baud (they could also do 110 and 300). Why would you want a modem that operates faster than you can read? Much online-time is spent uploading and downloading software. It can also be cheaper to quickly download text and read it at your leisure.

Today, the low-end price of modems capable of 2400 baud (in addition to the three slower speeds) is below $200; these will probably be the best sellers in 1988. (The 2400 baud feature isn't always usable, however, because the remote modem you're connected to must use the same baud rate as you. Many of the computers you can call either don't offer 2400 baud service or they have surcharges associated with it.)

When you use an external modem, you actually have two different communication links to contend with. One is the phone link between your modem and the remote modem, the other is the serial link between your modem and your local computer. Most external modems are smart enough to sense what baud rate you have told your computer to use for the computer-modem connection and they set the modem-modem connection to match it. Sometimes, however, this isn't possible, as when the modem answers the phone. If the computer-modem link is set for 300 baud and a 1200 baud call comes in, Hayes compatible modems will send (at 300 baud) the message "CONNECT 1200" to your computer, then switch to 1200 baud. The communications software in your computer must be smart enough to recognize that "CONNECT 1200" means a 1200 baud connection has been established and must respond by changing the baud rate of the computer-modem serial connection accordingly. ("CONNECT" alone means 300 baud.)

The best kind of modem for Apple II users today is a modem compatible with second-generation Hayes protocol. This is because all but the oldest communications software available for the Apple II will work with this kind of modem. If you have a Micromodem-compatible or an all-other-compatible, it is more difficult to find communications software that will work with your equipment. (*Talk is Cheap*, for example, works only with Hayes compatibles.)

If you have an Apple II-Plus or Apple IIe, one problem with an *external* Hayes-compatible modem is that you'll also need to buy a serial card. This isn't a problem with the IIc or IIgs, which have "serial cards" built in (however, it can be difficult to get the right cable for these computers because of their connectors). II-Plus and IIe owners can get around the serial card requirement by buying an *internal* modem that *appears* to be an external modem connected to a serial card. As mentioned earlier, Applied Engineering's DataLink is a popular example of this type of modem.

Besides baud rate, there are some other communications parameters known as "data format" and "parity" that tend to scare beginners. But there's really nothing to be scared of here–just remember the three letters "8N1" and throw them like a magic spell whenever you encounter the data format and parity demons. (If this doesn't work, or if you insist on understanding this stuff, see Uncle DOS's introductory comments in the October 1987 issue, page 3.69.)

The only other scary thing beginners have to know about is called "duplex." Duplex has to do with whether the remote system you're connected to bounces the characters you send back to you. (Obviously, if you both do this, the first character you send will bounce back and forth like a ping-pong ball and consume your connection.) The easy way to handle this duplex stuff is to tell your communications software to use half duplex. Then, if what you type appears twice (llliikkee tthhiiss), switch to full duplex to stop the stuttering. It's also good to know that when you can't see what you're typing, the problem is probably that your communications software is set for full duplex but you're actually connected to a half duplex host.

Three layers of complexity. Using your computer for communications can get pretty complicated because of the layers of devices and commands you have to eat through. The first layer is your own computer and communications software. Most Apple II programs use the open-apple key or the mouse for commands–since you can't transmit the open-apple key or mouse movements by mistake, they make good ways to tell the software itself what to do. Configuring the software for your modem and writing macros is often the most difficult part of using communications software.

The next layer is the modem and its commands. While these can seem pretty complicated if you actually read your modem manual, the two most important commands–the ones for dialing and hanging up–were covered in detail here earlier.

The final layer is the remote computer and its commands. Since each and every remote computer seems to have its own unique software, there's a lot of learning to do to get the most out of the friendly computers that answer the phone. It's a big help, however, if you can keep the three layers separate in your mind. Learn how to use your communications software, learn how to use your modem, and next month I'll start teaching you how to use GEnie. If you're one of ***Open-Apple's*** paid subscribers, there was a slip of paper your envelope this month. The slip explains how to get a no-initiation-fee GEnie account. Since GEnie has no minimum monthly charge, there's no longer any reason for you to avoid computer communications.

Ask (or tell) Uncle DOS

On page 86 of Vol. 1, No. 10, the reference to page 73 in the twelfth line should be to page 72. This is a correction to a correction. Can I lay claim to Weishaar's Law (The probability of a mistake in a correction is an order of magnitude greater than the probability of a mistake in other text) or is this already taken?

65816 assembler standards

(Editor's note: the following letter originally appeared in the January 1988 ***APDAlog****, Vol 3, No 1. It is reprinted here with permission. For more information on the Apple Programmer's and Developers Association, call 206-251-6548.)*

Although the original 6502/65c02 microprocessor chips used a streamlined and elegant set of instruction mnemonics, the 65c816 is plagued with a set of inconsistent mnemonics and addressing modes. The result is a difficult to learn instruction set and programs that are hard to debug. Faced with these problems, many 65c816 programmers would love to work with a redesigned mnemonic set for the 65c816 microprocessor. This letter is an introduction, report, and invitation concerning the work occurring towards that goal.

Have you ever wondered why the 65c816 used the mnemonics SEC and CLC to set and clear the carry flag, SED and CLD to set and clear the decimal flag, SEI and CLI to set and clear the interrupt disable flag, and then (totally ignoring convention) SEP and REP to set and clear bits in the processor status word? Why not SEP and CLP? Other idiosyncrasies also exist. For example, why is the same syntax used for data operations, e.g.

```
LDA Label ;Label is a direct page value
LDA Label ;Label is an absolute value
LDA Label ;Label is a long value
```

but different instructions are used for transfer of control?, e.g.

```
BRA Label ;Label is a short (near) value
BRL Label ;Label is an absolute value
JMP Label ;Label is an absolute value
JML Label ;Label is a long value
```

The current (de facto) standard for 65c816 assembly language is the APW assembler (which is, for all intents and purposes, identical to ORCA). The existing standard did not come about because a group of people sat down and agreed on it. Rather, it happened by default because Byte Works was the first company to produce a 65c816 assembler and Apple Computer adopted it. Now ORCA isn't that bad, it's just that its design is based on the IBM 360 assembler, a technology that is over 20 years old at this point. In the past 20 years there have been several advances in machine and software architectures that have rendered the design of the 360 assembler obsolete. Therefore, adopting the philosophy of the 360 in a brand new product, while useful to those who've used 360 assembly language in the past, is a rather precarious thing to do.

Anyone who had the opportunity to use Microsoft's 8086 assembler under MS-DOS has used a truly modern and powerful product. I mention this assembler not to rankle those who detest IBM, but to point out that IBM PC programmers are using powerful tools, and (quite honestly) laugh at the "toys" that Apple IIgs programmers are using. This reflects poorly on the hardware on which such tools run, and is (no doubt) one very big reason why software developers on the IBM PC would never consider writing programs for the Apple IIgs.

For the past several months, I've been enlisting the interest of several people in the Apple and 65c816 communities concerning the formation of a committee to produce a powerful and definitive 65c816 assembly language syntax standard. The response has been very good. Some very important people are interested in either participating in or monitoring the work of the standards committee: William Mensch, the designer of the 65c816 and President of the Western Design Center, has agreed to host a conference at WDC to iron out the standard and act as a clearinghouse for information. Others interested in participating include Mike Westerfield (Byte Works/ORCA), Glen Bredon (author of Merlin), Roger Wagner (publisher of Merlin), Bob Sander-Cederlof (SC Software/SC Assembler), Brian Fitzgerald (HAL Labs/Lisa816 Assembler), along with several others (including some authors of 65c816 books and other programs). With support from most of the major 65c816 assembler and book authors, Apple Computer's IIgs programmers, and the Western Design Center, the only major group missing from the design process is the people who are actually going to purchase and use the product, i.e., you, the user/developer community. This is your invitation to participate in this important event, which will, undoubtedly, affect the way you write programs for the Apple IIgs in the near future.

The 65c816 assembly language syntax standard will be developed in two phases: during the first phase (which is occurring right now), all participants are encouraged to submit proposals and suggestions for the standard. These suggestions will be collected and disseminated to all the participants. This collection-dissemination cycle will be repeated as often as necessary to allow revisions and new ideas to enter into the discussion, up until May, 1988. During the summer, a conference will be held at the Western Design Center, at which point the participants will settle on the final standard. Hopefully, the first products implementing the standard will appear in late 1988.

The scope of the standard covers two areas: the features to be implemented in the assembler itself, and the library of routines to be provided with each assembler meeting the standard. If you would like to participate in the design of the standard, send a stamped, self-addressed envelope along with your name, address, phone number, and a brief paragraph describing your interest. If you have any other suggestions or comments, these would be welcome as well. Send all materials to:

Randall Hyde
65c816 Standards
2271 Indian Horse Dr.
Norco, CA 91760

I editorialized a bit on the apparent lack of thought that went into the words used to describe 65816 assembler operations back in August 1986, pages 2.49 and 2.50. I fully support this effort.

Classic desk accessories

I've written a few classic desk accessories for my IIgs and they work fine when they are BRUN from Applesoft. But the Desk Manager documents say that if I make my CDAs file type $B9, add a header, and put them in the DESK.ACCS subdirectory of my System Disk, they will be installed automatically at start up. What it doesn't explain is whether the Desk Manager moves the code somewhere or if the code has to be relocatable. Anyway, it doesn't work.

Andrew Benson
Gladstone, Mich.

The missing piece of the puzzle is that the file also has to be in ***Apple Programmer's Workshop*** *"object module format." This format includes everything the System Loader needs to know to relocate the file. APW object module format is supported at the moment by only two assemblers, APW's and* ***Merlin816*** *from Roger Wagner Software.*

Overstrike cursor at startup

When AppleWorks boots up, the default cursor is the insert cursor. Those of us who are less than perfect typists soon find ourselves pushing text all over the screen. Is there a patch that would allow AppleWorks to start up with the overstrike cursor?

Rich Brossman
Bay Village, Ohio

Our AppleWorks patch guru Alan Bird suggests that you do this (using Basic.system 1.1!):

```
POKE 768,1
BSAVE APLWORK.SYSTEM,A768,TSYS,L1,B$x

For "x" in the above line use:

AppleWorks 2.0: $18D8
AppleWorks 1.3: $17A8
AppleWorks 1.2: $1657
```

IIgs dual speed keys

The IIgs keyboard has a wonderful option that I haven't seen discussed in ***Open-Apple***. Open a letter or large data base in AppleWorks. Then hold down on an arrow key to scroll through the document and press on the control key. I have always been impatient with the slow speed of the cursor. Now I find myself sitting on edge trying to stop it where I want it.

Gary P. Bungart
Deltona, Fla.

The IIgs Control Panel's "Options" menu has four items that control the speed at which keys repeat. "Repeat Speed" is an adjustable control that affects how fast all keys repeat. "Repeat Delay" is an adjustable control that affects how long it takes for a key to start repeating after you press on it and hold it down.

What you've pointed out is that the arrow

keys repeat at two different speeds. (The only reason we hadn't mentioned this before was that none of us had read that page of the manual.) You get the faster speed when you press on the control key while holding an arrow key down. You can add this dual speed capability to the space and delete keys, too, by changing "Fast Space/Delete Keys" in the "Options" section of the Control Panel to Yes. And you can make the control-key trick work even faster by changing "Dual Speed Keys" in the Control Panel from "Normal" to "Fast."

Slashed zeros for ImageWriter II

Thanks go to Bill Shuff and Bill Tudor for showing me how to get slashed zeros out of my ImageWriter II when using AppleWorks. Go to "Specify information about your printer" on the Other Activities menu. Select "Change Printer Specifications," "5. Interface cards." Add the following codes to the end of the codes you are already using: "Escape D control-@ control-A Escape Z control-@ control-@".

Quentin R. Packard
Troy, N.Y.

Time to set the clock

I had a call the other day from a frustrated AppleWorks user. She had just gotten a new IIgs and was having difficulty getting past the "Enter today's date" screen. This sounded a lot like the "Time stopped short for ProDOS 1.1.1" problem you discussed last month (page 4.2), except that she was using a IIgs and an updated version of ProDOS. So put in your notebook that another diagnosis for these symptoms is that the clock in the IIgs is wrong. Neither the factory, the dealer, nor she had bothered to set it–it thought the year was 1978.

I have a written a program that permits editing of the AppleWorks SEG.PR file. It has been circulated locally for almost a year now, and while it has a few rough edges and lacks proper documentation, it seems to work reasonably well. I'll provide a copy and documentation on how to use it to anyone sending me a blank disk and return postage.

Nevin Diener
Rt 1, Box 213
Keezletown, VA 22801

We'd be happy to upload a program like yours to our software library on GEnie, where it would get much wider distribution. However, we need those of you who send us such stuff to include written permission to upload when you send in the program.

Database bug, first line squashed

Regarding data base crashes (December, page 3.87; January, page 3.96), I have discovered something relevant. If you try to save a large file to an almost full disk, AppleWorks may say, "Insufficient room on this disk–is it okay to delete the old copy." If you answer yes, things seem to proceed normally, but these are exactly the data base files that won't load later. It seems that AppleWorks may be messing up the file header while doing this automatic delete and save. Better to always answer "No" and use Other Activities to delete the old version of the file yourself.

Our office uses AppleWorks on an Apple IIe that's connected to both a dot matrix and a daisy wheel printer. At home I use my own copy of AppleWorks on a IIgs with an ImageWriter. Both systems invariably garble the first line of type on a new page, regardless of which printer is being used. In the case of the dot matrix printers, the first line usually comes out rather squashed looking. The daisy wheel printer almost always gives only half a linefeed after the first line.

C.L. Roberts
Lafayette, Calif.

We have now had several reports that the data base bug we've been tracking happens only when you allow AppleWorks 2.0 to automatically delete a file. It is not related to any enhancement programs. The solution, as you point out, is to never answer "yes" to the auto-delete question.

While the only thing that appears to be common about your three printers is AppleWorks, my bet is that the problem is elsewhere. The squashed-line effect is caused by the paper not advancing correctly. Make sure the paper can flow easily into and out of the printer. Obstructed paper paths cause all kinds of problems. Also check all the printer settings that have to do with paper grip. In particular, there's a control inside most printers for adjusting the distance between the print head and the platen. It's usually used when you change the thickness of the paper. Four part forms, for example, are a lot thicker than a single sheet. Try adjusting this. Another possibility is to change to a heavier or lighter paper stock. If you are using single sheets of paper rather than continuous forms, the platen can get so slippery that it won't advance the paper correctly. I have a dim memory that printer/typewriter technicians have some kind of chemical solution to remedy this.

CHAIN, STORE bug fix

You've discussed the bug in the Basic.system CHAIN and STORE commands a couple of times in ***Open-Apple*** (December 1987, page 3.87). Thanks to your information and past installments of Sandy Mossberg's "Disassembly Lines" column in Nibble, I now have a patch to fix it.

I searched through Basic.system and discovered that the sequence 20 7F A3 (JSR MEMUP, the subroutine that causes all the problems) occurs only twice. One of these occurrences, at $A270, is part of the garbage collection routine. The other, at $A44F, is part of the PACKVAR subroutine, which packs variables prior to a CHAIN or STORE.

I patched this second occurrence of JSR MEMUP to call my patch. The patch performs the machine language equivalent of POKE 41859,3 before calling MEMUP and does a POKE 41859,7 afterward. This is functionally equivalent to the method given in December, except that now it's totally automatic. Since I didn't patch the other occurrence of JSR MEMUP, it calls the unmodified version of the routine, which is the correct one for that section of code.

I put the patch in a section of Basic.system at $BB4C that seems to be completely unused. Mossberg used this area to fix the BSAVE bug in the August 1986 issue of *Nibble* (page 93). I put my patch right after his so that they could be used together without any conflicts. Here's how to install the patch:

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

459D:A9 03 8D 83 A3 20 7F A3 A9 07 8D 83 A3 60
2E4F:20 9D BB
2D83:07

3D0G
BSAVE BASIC.SYSTEM,A$2000,TSYS
```

With Mossberg's TRACE patch (***Open-Apple***, July 1985, page 1.56), his BSAVE patch, and my CHAIN/RESTORE patch, I believe we have finally squashed all the bugs in Basic.system. Apple may or may not release a revision, but until that happens we at least have a completely functional version.

Here's an interesting tidbit for Applesoft programmers. If you want to modify your program but don't want to lose the values in your variables, CALL 41997 before making your changes and CALL 42098 afterward. These two CALLS are to PACKVAR and UNPACKVAR, the subroutines that CHAIN and STORE use to put variables in high memory. After typing CALL 42098, use GOTO to re-enter your program. Warning: this may not work if your have a very large program that uses lots of variables.

Jerry E. Kindall
Grove City, Ohio

To your patches we would add only:

```
2283:D2
```

This will make the startup screen say "BASIC.SYSTEM 1.1R".

Open-Apple

[redacted]

is written, edited, published, and

with help from

Tom Vanderpool **Sally Dwyer**
Dennis Doms **Steve Kelly**

Open-Apple has been published monthly since January 1985. World-wide prices (in U.S. dollars; airmail delivery included at no additional charge): $28 for 1 year; $54 for 2 years; $78 for 3 years. All back issues are currently available for $2 each; bound, indexed editions of our first three volumes are $14.95 each. Volumes end with the January issue; an index for the prior volume is included with the February issue.

Please send all correspondence to:

Open-Apple
P.O. Box 11250
Overland Park, Kansas 66207 U.S.A.

Open-Apple is available on disk for speech synthesizer users from Speech Enterprises, P.O. Box 7986, Houston, Texas.

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

WARRANTY AND LIMITATION OF LIABILITY. I warrant that most of the information in ***Open-Apple*** is useful and correct, although drivel and mistakes are included from time to time, usually unintentionally. Unsatisfied subscribers may cancel their subscription at any time and receive a full refund of their last subscription payment. The unfilled portion of any paid subscription will be refunded even to satisfied subscribers upon request. MY LIABILITY FOR ERRORS AND OMISSIONS IS LIMITED TO THIS PUBLICATION'S PURCHASE PRICE. In no case shall I or my contributors be liable for any incidental or consequential damages, nor for ANY damages in excess of the fees paid by a subscriber.

ISSN 0885-4017 **GEnie mail: OPEN-APPLE**
Printed in the U.S.A.