A2-Central™

formerly Open-Apple

December 1992
Vol.8, No.11

ISSN 0885-4017
newsstand price: $3.00
photocopy charge per page: $0.20

A journal and exchange of Apple II discoveries

Coping with fast modems

By Dennis Doms

You may have noticed that several companies are now carrying high-speed modems at drastically lower prices than a few months ago. Although normally advertised for (and sometimes packaged with software for) Mac and PC compatible systems, these modems can also be used on Apple II systems.

There are many communications standards in use and their names look like alphabet soup, so grab a spoon and let's get a taste of the prominent modem standards.

The early personal computer modem standards were Bell 103 (300 "bits per second" or "bps," also called the "baud rate") and Bell 212A (1200 bps). These standards were used mostly in the United States and Canada.

As modems became more common, standardization moved into the international realm of a committee known as the CCITT. CCITT v.22 is the international 1200 bps standard (it's similar to Bell 212A). At the next level, things merged; CCITT v.22bis (2400 bps) is used pretty much around the world.

At 9600 bps several standards were tried but CCITT v.32 seems to be emerging as the "winner" by popular consensus. Some high-speed modems also support CCITT v.32bis for 14,400 bps. These standards, implemented on a chip set available to modem manufacturers, are the ones supported by the recent plethora of high-speed FAX modems.

As with all these standards, the modem must be communicating with another modem supporting the same standard. (That is, a v.32 modem can't communicate at 9600 bps with another modem using a non-v.32 standard.) This is especially confusing for the several 9600 bps protocols marketed, but v.32 is becoming the defacto "standard".

(If you think 14,400 is fast: a v.32fast standard for 28,800 bps is being proposed.)

Many modems also support error detection/correction and data compression protocols. An error-correcting modem sends additional information with the data to allow another error-correcting modem (of the same standard) to determine if something has gone wrong in transmission and to attempt to compensate for it. A modem with data compression is smart enough to "compact" the data so it takes less time to transmit, and the receiving modem (of the same standard) is smart enough to recognize and decompress the data before passing it on to your computer.

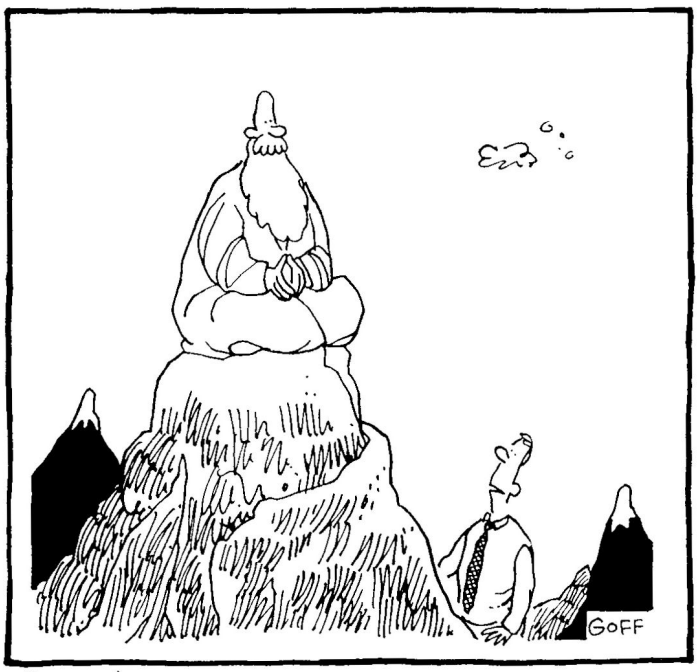
Several proprietary standards evolved into the Microcom Networking Protocols 1 through 5 (MNP 1-5). MNP protocols 1 through 4 encompass various types of error checking and MNP 5 (which includes the 1-4 standards) adds data compression. MNP 5 is the most common implementation today and offers data compression up to about 50 per cent of the data stream. That is under ideal circumstances, however. On a precompressed file (such as most files archived for downloading) MNP 5 will actually increase the time needed to send the file. Many BBS's and online services, including GEnie, use MNP 4, which provides error correction but not data compression, for this reason.

The common standards you'll see on high-speed modems are v.42 for error correction and v.42bis for data compression.

When you set up a high-speed modem connection you need to use two modems with compatible standards and you need to make sure that your terminal (for most Apple II users, your computer with communications software) is capable of communicating at the maximum speed of the modem. There are some facets to this last part — the maximum communication speed of your computer, that aren't obvious. Let's take a look at them.

Because high speed modems can compress data, they can actually transfer more data through the phone line than the bps rate itself would suggest. With v.42bis a total compression of 4 to 1 is possible. This means a 14400 bps modem can conceivably transfer at a rate of 57600 (4 times 14400) bps, at least for short bursts when the data can be compressed by that ideal 4 to 1 ratio. This means your computer's serial port needs to be set to 57600 baud so that your modem can communicate with your computer as fast as it does with the other modem. Whether your serial port can meet this challenge depends on what computer and what communications software you're using.

The Apple IIe Super Serial Card is pretty much limited to about 19200 bps on a "stock" IIe. Using an accelerator to speed the



"THAT'S HOW TO ATTAIN ENLIGHTENMENT? COMMAND OPTION CAPITAL E?"

computer beyond the "factory original" one megahertz may help reliability with some programs.

The serial port software built into the IIgs will support communications up to 19200 bps; this limit is primarily imposed by the fastest speed at which the programs can grab data from the serial port. The built-in routines are "general purpose" so that diverse programs can use them; a communications package can improve upon these rates by using its own specialized routines that bypass the built-in software. A programmer who does this on the IIgs has to be very careful to insure that the serial port is available; although the IIgs is technically not a multitasking machine, it's conceivable that a co-resident program such as a desk accessory might be using the serial port.

The Mac serial port hardware is nearly identical to what's in the IIgs, though the Macintosh system software is obviously not the same and very fast models of the Mac exist. We decided to test some Apple II combinations along with both slow (an 8 MHz Mac Classic) and fast (a 20 MHz IIsi and 25 MHz PowerBook 170) Macs to see how they fared.

We used an MS-DOS PC for the host in our tests. The PC has an advantage in that it can use a version of a serial chip that "buffers" (remembers) characters when the computer isn't quite ready to deal with them. By using a serial port with this type of interface chip (a 16550) we were able to get much better results than with non-buffered chips. Although the microprocessor in a PC can run at very high speeds, in most cases the internal data transfer circuitry, known as the bus, is limited to 8 MHz for compatibility.

We performed some speed tests using the fastest Apple II drivers we could find (which happened to be in *Proterm 3.0*) to compare with some results using Mac and PC systems, which we'll report later. Since Apple II users are a diverse lot it's hard to be specific about the best combination and ultimate achievable speed, but overall, using optimized drivers (primarily a function of the software you're attempting to use) and faster hardware (usually achieved by accelerating the processor) will help.

Since we plan to push the serial port for all it's worth, we have to be sure that it can shout "Wait!" if it can't keep up. As usual in the computer world, there's a software and a hardware way to do this. Software handshaking involves sending control code characters in the data stream to indicate when to start or stop the data flow. The most common is "XON/XOFF" handshaking. When the device is nearly "full" it sends the "XON" character (usually defined as control-S) to the sender to tell it to stop sending data as soon as it can. When the receiver is ready again, it signals the transmitter by sending it an "XON" character (usually a control-Q).

XON/XOFF is very simple and requires only that the transmitter be ready to accept the signal and stop transfer until told it's okay to resume. The trouble with XON/XOFF is that it requires the software on the transmitting side to take some time out of sending data to occasionally watch for the XOFF character. It also means that the transmitter has to be careful not to send more data between checks for the XOFF character than the receiver might reasonably be expected to still have room for (this is subject to tricky limitations and there are no precise "rules" for the reserve space to expect). These factors cause XON/XOFF to interfere with the rate of data flow as the communication speed increases.

The other solution is hardware handshaking, where the receiver uses a hard wired connection to tell the transmitter to stop or start transmission. In this case, when the "stop" signal is applied, the transmitter's peripheral interface will tell the transmitting software it's not ready to send the next character. Since the software normally checks to see if the interface is ready before stuffing the next character in it anyway, this doesn't require the software to expend extra effort (and time) checking for readiness like XON/XOFF does. Also, since the signal is transmitted at the speed of electricity through wire (instead of at the communications bps rate) the transmitter is able to stop without any risk of overrunning the receiver's buffer. The drawback for this timeliness is that, since the handshake signal can't be generated on the same wires as the data, you need more wires in your cable to implement it. You can make a cable for a connection using software handshaking with as few as three wires (transmit signal, receive signal, and signal ground); hardware handshaking requires at least two extra wires (more if you want to continuously insure your connection is intact) for bidirectional control.

There are several schemes for hardware handshaking; with modem communications, several different handshakes may be used for different purposes. Overall, communications equipment is divided into two general classes: Data Terminal Equipment (DTE) and Data Communications Equipment (DCE). In general you can think of a device that produces output in human-readable form as DTE and a device that moves data to another location as DCE. You computer is DTE because it presents information on a screen. Likewise, printers are usually considered DTE since they put information on paper. A modem is DCE since it sends data to another modem but doesn't display or print data itself.

Handshaking is often defined in terms of these types of equipment. Overall, the common signals are:

Signal	Acronym	Signals
Ground	PG	protective (shield or frame) ground
ring indicator	RI	indicates phone line "ring"
Signal ground	SG	common baseline for signal reference
Transmit data	TxD	data signal output
Receive data	RxD	data signal input
Data carrier detected	DCD	device is connected (from DCE)
Data terminal ready	DTR	device is ready to send data (to DCE)
Data set ready	DSR	device is ready to send data (from DCE)
Request to send	RTS	device wants to send data (to DCE)
Clear to send	CTS	device is ready to accept data (from DCE)
Secondary CTS	SCTS	(alternate) device is ready to accept data (from DCE)

These signals are defined here as applying to data communications equipment, but the other end of the signal will connect to data terminal equipment (that is, RTS is a signal defined as to DCE, but could also be thought of as a signal from DTE). Some are redundant, such as SCTS.

Exactly how these signals are used for handshaking (in the case of modem communications) varies with the implementation of the DTE (computer) to DCE (modem) connection. Differences in the way manufacturers have decided to design their serial ports has caused more than a little fragmentation of the RS-232 "standard"; the limitation usually comes from trying to minimize the size of the serial connector.

The use of these handshake signals is defined by the RS-232 specification of the Electronic Industries Association. These can be explained in an analogy to the telephone.

DTR and DSR indicate whether the person speaking (DTR) and the phone (DSR) are each ready to initiate a conversation.

RI is the bell to indicate someone wants you to pick up the phone.

DCD indicates whether the phone at the other end of the line is on the line and available for conversation. (Even if neither side is actually saying anything.)

RTS and CTS indicate whether or not the speaker wants to speak (RTS) or is ready to listen (CTS).

TxD and RxD correspond to the phone's mouthpiece (for transmitting data) and earpiece (for receiving data).

A typical exchange may go like this:

The phone rings. (RI asserts itself; the modem passes this on to the computer).

You pick up the phone. (DTR is asserted to indicate the terminal is responding to the ring.)

The phone isn't dead. (The modem is asserting DSR as "true," indicating the modem is ready.)

The phone gives you background noise instead of a dial tone. (DCD is indicated by the modem to verify it has established communication with a remote system.)

You prepare to speak. (RTS is asserted.) There's no voice coming through at the other end, so you know they're waiting for your response. (CTS is asserted.) You say "Hello". (TxD.)

The other party is ready to answer. (RTS.) You're ready for them to speak. (CTS.) You hear "Hi, this is Beth." (RxD.) You want to respond (RTS) but you're waiting for them to finish (CTS). Then you speak (TxD).

This loop could go on for hours. One way it could end is for the phone at the other end to die (DCD is lost). Another is that your phone could suddenly fail (DSR is lost). There are other more grisly possibilities, but let's assume everything precedes normally...

At the end of the conversation, either the remote party hangs up first (DCD is lost, telling you the other modem has hung up) or you hang up (DTR is set false, telling your modem to hang up). At this point, you're ready to start again.

Back in the real world, it isn't always this simple. Not all serial devices use all these signals, and so both computers and modems may use alternative methods of monitoring events. You can conduct a serial conversation with as few as three signals (TxD, RxD, and ground) by using software handshaking. However, this isn't ideal for high speed communications; the telephone equivalent would be explicitly saying "Okay, you can speak now...please wait...go ahead" to let the other party know you're keeping up with their conversation.

Hardware handshaking, even with interface peculiarities, is the more efficient solution, and becomes increasingly desirable (and even necessary to maintain speed) as the communication speed increases. Hardware handshaking is also necessary in situations where software handshaking codes might interfere with interpretation of the data stream, such as when raw data is being transferred. (There are software methods to coordinate these transfers, but they require diluting the data with further control codes.)

To use hardware handshaking you need the proper connections and coordinated software support. We took several examples out of the *ProTERM* manual since it actually recommended the cabling it preferred.

The Apple IIe Super Serial Card and most PC serial ports recognize nearly all of the defined handshaking signals. The Apple Super Serial Card uses a DB-25 (D-shaped connector with 25 pins), but only 10 wires are actually carried from the card to the connector. The missing signal is the "ring indicator".

The PC recognizes 9 signals on the common "AT-style" DB-9 connector (D-shaped connector with 9 pins); SCTS and frame ground are omitted. Some PCs serial interfaces may use the older XT-style DB-25 connector; on ones we've seen, you are often limited to the same nine signals.

With plenty of wires, hardware handshaking for the IIe is no problem, a standard RS–232 cable with all 25 pins wired straight through will work fine. (Actually, you only need the wires defined on the Super Serial Card interface, but full cables aren't hard to find and generally cost little or no more.)

The Mac and the IIgs have a problem due to a dual nature of their serial port. To support AppleTalk, the Mac and IIgs use a high-speed serial interface design that uses a serial standard called RS-422. There is enough correlation with RS-232 and RS-422 to allow connecting the two, but the RS-422 design calls for two transmit and two receive signals. The transmit and receive lines are configured with "negative" and "positive" pairs and the signal is expressed as a difference in the signal levels across the pair. This means noise is less likely to interfere with the proper detection of the signal data and the signal is more reliable. It also means that with RS-422 it is necessary to dedicate four wires (instead of two) to the data flow.

Apple designed the IIgs and Mac serial ports using an 8-pin connector. The supported signals are DTR (handshake out), "HSKI" (handshake in), TxD-, TxD+, ground (reference and supply), RxD-, RxD+, and "GPI" (general purpose input). Apple seems to like non-standard nomenclature, so we went to the IIgs schematic (in the *Apple IIgs Hardware Reference, Second Edition*) to find out that HSKI is connected to CTS and GPI is connected to DCD.

You have to be careful with modem cables on the IIgs and Mac. If you plan to use a high-speed modem (9600 bps or higher) you'll want to invest in a hardware handshake cable (some companies sell these designated as "Hayes V-series modem cables"). *ProTERM*'s manual does caution that you can't use hardware handshaking, transmit interrupts, and true carrier detect all at the same time on the IIgs interface. *ProTERM* does not enable the modem's carrier detect feature ("&C1" in most modern modem's command set) for this reason.

The Apple IIc only has five signal lines on the modem port: DTR, DSR, TxD, RxD, and ground (power and signal common). The IIc port uses a DIN standard connector with only five pins; the IIc-Plus uses the same five signals mapped to the IIgs-style mini-8 DIN connector. Five signals might seem to give the IIc a minimum complement of signals for bidirectional handshaking.

But, to complicate matters, Apple's labels for those pins aren't strictly kosher. Each "DSR" pin is actually connected to the "DCD" pin of its respective 6551. The real "DSR" pin from the Port 1 6551 ACIA (the serial interface component used in the IIc) is connected to an EXTINT* signal on the IIc's disk port. The "DSR" pin from the Port 2 ACIA is connected to the KSTRB signal to generate keyboard interrupts. Each of these signals is treated as an input from an "external" source (disk port or keyboard) in order to generate an event interrupt. (If you need to know more about this, consult the *Apple IIc Technical Reference* manual, especially Appendix E on interrupts.)

According to *ProTERM*'s manual, the limited number of signals allows hardware flow control in one direction only. *ProTERM* simulates bidirectional hardware handshaking in its driver.

Once the handshaking is right, the issue is *throughput;* how fast the information flows through the system. Using hardware handshaking instead of XON/XOFF ensures the maximum amount of data can be pumped through the modem/computer connection. The remaining question is, "Can the computer keep up with the modem?"

The theoretical maximum throughput of the v.32bis/v.42bis modem is 57,600 bps (14,400 bps times a possible data compression factor of 4). Therefore the modem will communicate with its computer terminal at bps rates up to and including that speed.

The actual amount of data passing through the modem may be somewhat less. Any increase over the basic bps rate (9600 or 14,400) from the data compression aspect will be limited by how "compressible" the data is. Files compressed with *ShrinkIt* (or a similar high-efficiency compression utility), for example, are already about as compact as a reasonable "lossless" (without any loss of information) compression algorithm can make them and further effort in compression is just wasting time. In this case, the modem will effectively be transferring data at its basic speed.

Another limitation is the error rate of communication across the phone line. These modems monitor the signal quality of the phone line and can "fall back" to a lower bps rate if the line quality causes so many errors that data could actually be sent faster at a lower speed. When the line quality seems to improve, the modems can step up to a faster speed. (You might not think that 2400 bps can be faster than 9600 bps, but it may be if a block of information can be sent clearly the first time at 2400 bps as opposed to having to be sent several times at a higher speed before it arrives intact at the other end.)

In other words, using the maximum theoretical speed between the modem and the computer is not an indication that all information is

transferred at that speed, or even that much of it is. Using the maximum bps rate just insures that the data will always flow with as little obstruction as possible.

In the real world, the computer may have limits, too. We tried to determine the maximum bps effective rate at which several popular Apple II configurations (as well as a few other computer systems) could transfer data reliably. It turns out that, depending on your configuration, you may be better off setting your computer/modem communication speed as low as 19200 bps to get the best performance. In fact, certain configurations may require that you set the rate even lower (to 9600 bps).

The systems tested were: an Apple IIgs with an 8MHz *ZipGSX* with 16K cache, a Mac SE, a Mac IIsi, and a Mac PowerBook 170. The software used was *ZTerm* on the Macintosh, and *Talk is Cheap* and *ProTERM* on the Apple II systems. Files were downloaded over a null modem cable from a PC running *ProComm Plus* (DOS) in host mode. The file transferred was an uncompressed application file of about 214016 bytes in size.

For *Talk is Cheap* we used the YMODEM transfer protocol. YMODEM transfers blocks of data of 1024 bytes along with a verification value calculated from the data in the block and then waits for an acknowledgement from the recipient. If the block is not received intact it is sent again. The more times a block isn't correctly received the more times it must be resent, so the number of errors and transfer rate is an indication of how successful the connection is.

For *ProTERM* and *ZTerm* we used ZMODEM transfers, a more robust and complicated protocol. The easy way to think of ZMODEM is that it sends data without waiting until the recipient identifies an error. At that point the recipient tells the sender which block of data was bad and it is resent. Since ZMODEM doesn't wait for acknowledgement of each individual packet it can proceed much more quickly until an error actually occurs.

With the Apple II system we timed tests with and without acceleration to see how much aid more CPU speed would be. Also, on the IIgs we tried some transfers with AppleTalk enabled to see if AppleTalk's use of CPU time would hinder high-speed transfers (look for the "AT" setting in the comments). Transfer times were reported in seconds and in characters per second (cps values in parentheses were calculated in situations where the receiving program didn't report the figure itself). Here's what we came up with:

System (bps rate)	Time	cps	errors	comment
Ymodem				
Apple SSC (9600)	failed			1Mhz; Talk is Cheap
Apple SSC (9600)	236	(907)	0	8Mhz; Talk is Cheap
Apple SSC (19.2K)	failed			
Apple IIgs (9600)	314	(682)	0	2.8MHz; Talk is Cheap
Apple IIgs (19.2K)	313	(684)	0	
Apple IIgs (9600)	236	906	0	8Mhz; Talk is Cheap
Apple IIgs (19.2K)	191	1133	0	
Zmodem				
Apple SSC (9600)	241	888	0	1MHz; ProTERM 3
Apple SSC (19.2K)	199	1075	2	
Apple SSC (19.2K)	195	1097	0	8MHz; ProTERM 3
Apple SSC (38.4K)	failed			
Apple IIgs (19.2K)	124	1725	0	AT off; Zip off
Apple IIgs (38.4K)	158	2183	9	AT off; Zip off
Apple IIgs (19.2K)	120	1783	0	AT off; Zip on
Apple IIgs (38.4K)	177	1209	15	AT off; Zip on
Apple IIgs (38.4K)	132	1346	9	AT off; S2 fast
Apple IIgs (19.2K)	134	1597	18	AT on; Zip off
Apple IIgs (38.4K)	872	245	136	AT on; Zip off
Apple IIgs (19.2K)	126	1783	0	AT on; Zip on; S2 fast
Apple IIgs (38.4K)	530	403	74	AT on; Zip on; S2 fast
Mac SE (9600)	211	925	0	
Mac SE (19.2K)	failed			
Mac IIsi (19.2K)	123	1735	0	
Mac IIsi (38.4K)	81	2624	8	
Mac PB170 (19.2K)	117	1820	1	
Mac PB170 (38.4K)	failed			

The upper end on the Super Serial Card seems to be around 19200; even though *ProTERM* completed the transfer the time was not significantly faster than at 9600 bps. *Talk is Cheap* didn't complete (it usually got the file header and died with continuous errors) at 19200, but worked fine at 9600.

The IIgs serial port firmware seems solid at 19200. *ProTERM*'s low-level driver seems to be pushing things at 38400 but does complete the transfer. Setting slot 2 to "fast" (marked by "S2 fast" in the comments) seemed to help, so we left it there for the tests with AppleTalk on. AppleTalk does seem to interfere some; 38400 becomes slower than 19200 due to the number of errors *ProTERM* has to compensate for.

According to *Guide to the Macintosh Family Hardware, Second Edition* (page363) the "classic" Macs (it wasn't clear whether this meant only the 8MHz compact versions such as the Plus, SE, and Classic) support a maximum bps rate of 57600 through the Mac ToolBox (higher rates are possible by using externally clocked synchronous transfers, but consumer modems are asynchronous beasts). However, it looks to us like the Classic Macs could only muster a sustained throughput somewhere between 9600 and 19200 bps. The IIsi made it to 38400 with transfers cleaner than the IIgs, but the PowerBook 170 didn't, dying about 6 to 10 percent of the way into the transfer. (We've successfully used these systems as UNIX terminals at 38400 bps; the limits show up during raw transfers.)

Most common current serial ports on PC's will work at least to 38400 bps (the limit for our test). Older serial ports based on a non-buffering chip may not make it that high. The operating system may also influence the throughput; operating systems like Windows and OS/2 that are graphics-based may lower the maximum attainable speed on some systems.

What we're trying to convey is that the new high-speed modems do work on Apple II models with varying degrees of success. And in cases where the Apple II does have limits, moving to another type of computer may not solve the problem. If you plan to use a software package or interface other than the ones we have mentioned, check with the respective manufacturer to see if the package will support sustained high-speed transfers and what kind of cabling you will need.

Also recognize that most people are interested in high-speed transfers for the downloading of files. Such files are usually provided as compressed archives and as such will not be further compressed by v.42bis. For these files a successful transfer rate of 19200 bps will exceed the maximum attainable throughput on a 14400 baud modem and even 9600 bps may not represent a severe dropoff.—DJD

Miscellanea

Apple has finally joined the modern world by going mail-order. Apple has a new 40-page catalog that includes Mac and Apple II software, peripherals, and accessories as well as PowerBook computers.

With traditional Apple dealers seemingly an endangered species in some areas and Apple II dealers being even rarer, the new Apple distribution channel finally allows the rest of us to actually get hold of true Apple software and peripherals without having to search high and low. The computer and electronic "superstores" that Apple products seem to be moving toward just don't seem to always have a complete array of Apple products available, especially Apple II products.

Several accessories that are Macintosh peripherals also work on the Apple II and are listed in the catalog:

ImageWriter II
Personal LaserWriter NTR
Apple MIDI Interface
Apple Extended Keyboard
StyleWriter (IIgs with System 6.0 only)
Apple Desktop Bus Mouse
AppleCD 150
Apple SuperDrive
Apple Keyboard II

Add to this several Apple II specific peripherals:

Apple Joystick II
AppleMouse IIe
Apple II High-Speed SCSI Card
IIe Extended 80-column Text Card
Apple IIe Accessory Kit with 5.25 Controller Card
Apple IIe Memory Expansion Card (slot 1-7 card)
Apple II Workstation Card (AppleShare network interface for IIe)
Apple II SuperDrive Controller Card
Apple Disk 5.25
Super Serial Card
Apple II Video Overlay Card

And a few software items:

HyperCard IIgs
Apple II Aristotle Menu Software (AppleShare network menu)
Access II 1.3.1 (archaic communications software, not recommended)
Apple IIgs System Software 6.0

The mere presence of some of these items in the catalog may wake up some customers to the fact that the Apple II is still around. There are some things we'd like to have seen included (like Apple II CPUs) but being able to point someone to a location to order common Apple II interface boards and peripherals is still a great relief.

Apple has established a toll-free order number at 800-795-1000 (FAX 302-678-9200) which is in operation around the clock; the voice number doubles as the Customer Service number. (There's no international phone number at this time, although they're working on that.) Orders called in by midnight eastern time arrive the next business day, according to the catalog (Saturday and Sunday orders roll over to arrive the following Tuesday). For more information, call and order the catalog.

And, if you've picked up a used Apple... we're starting to get calls from new Apple II users who've picked up their systems used. Often the previous owner couldn't locate the original user manuals. Apple doesn't print and package these manuals for resale, so they aren't available through book dealers and similar sources.

However, near the back of Apple's new catalog, there's a sidebar advertising replacement manuals for several computers. Among them are manuals for the Apple IIe and IIgs. If you're missing original Apple manuals for your Apple II, call Apple's order number to see if they have the ones you need.

Also, if you have any scrap of your original manual (or disk) left, you may be eligible for Apple's Media Replacement Program.

Beagle Bros has scattered to the four (well, two) winds. In March Beagle Bros transferred most of its Apple II products to Quality Computers for distribution, upgrade and support. Beagle was hard at work on *BeagleWorks*, its integrated Macintosh software package (with word processor, database, paint, draw, communications, spreadsheet, and business graphics modules).

Despite glowing preliminary reports from many Macintosh magazines, *BeagleWorks* arrived late and with some bugs and performance problems. This month we heard that Beagle President Mark Simonsen has sold the package to WordPerfect Corporation. WordPerfect is a major player in the word processing arena with a reputation built on outstanding product support. Mark Simonsen is now the Product Manager for the new Works package; Mark Munz also follows to WordPerfect as the head programmer for the package.

Several former Beagle luminaries had already turned to other projects, including the founding of two companies still active in the Apple II market: WestCode Software (Alan Bird and Rob Renstrom) and JEM Software (Randy Brandt). Beagle is a beacon for the Apple II that will be missed.

Double-wide loads

Although most of our readers have moved to the IIgs, many still use IIe or IIc computers or run older programs on their IIgs systems. A question that recurs with astonishing frequency is how to enable the double high resolution mode available on the 128K IIe and its progeny, and more specifically how to display pictures created using double high resolution drawing programs.

The loading and displaying of double high resolution graphics (let's start using "DHR" as an abbreviation) is interesting in itself because it requires several preliminary steps to avoid problems. Most significant is that half of the DHR screen is taken from the auxiliary 64K of the 128K Apple II it is working with. That's the same area that the ProDOS /RAM volume occupies. If you have files stored on /RAM and move something into the DHR screen, the odds are very good that your files will be destroyed. So one requirement is that we have to make sure we don't start copying things to the DHR screen before we have the memory reserved.

The simplest way we've found to check /RAM for files is to BLOAD its directory (ProDOS lets you do that) and check the file count in bytes 37 and 38. If it's not "0" then /RAM is not empty; the user has to move the files somewhere safe first.

If /RAM is empty, we need to reserve the auxiliary memory portion of the DHR screen. Apple has made this absurdly easy; just BSAVE a file the size of a high-resolution screen to the /RAM volume. /RAM is arranged so that not only is the first file stored in the DHR screen area, it is stored in the correct order for display.

Using that knowledge, displaying DHR is easy. Save a dummy file to /RAM to reserve the memory. To load a picture, BLOAD the auxiliary memory half first, then BSAVE it over the dummy file. Then BLOAD the main memory half to the primary high-resolution screen area. Actually, it's easiest to load the auxiliary half to the main memory high-res display buffer at $2000 (8192 decimal) before saving it to /RAM, then just load the main half over the auxiliary half to complete the picture.

The fun and games is often in trying to guess how the DHR picture is arranged. Some programs save them as two files (one for each half of the picture). Some save the whole picture in one file, but either half may be first. In the following program you can set the type to ""1, "2", or "3" to indicate one of the three popular DHR program formats (*TimeOut Paint*, *Beagle Graphics*, or *Dazzle Draw*); one of the three subroutines at the end of the program will load and display the two halves of the picture.

All that's left is to flip the softswitches to display the picture. You can let BASIC handle most of this by using "PR#3" to initialize the screen to double-wide text (80-column) mode and HGR to display the high-resolution graphics screen. Then POKE 49246,0 to enable the DHR display and, optionally, POKE 49234,0 to display the graphics full screen (without four lines of the text screen mixed in at the bottom).

Of course, when done, put everything back to normal: POKE 49247,0 to disable DHR (and maybe reset the "mixed screen" setting with POKE 49235,0), use "TEXT" to turn off the graphics display, and issue "PRINT CHR$(21)" (print a Control-U character) to turn off the 80-column firmware.

Here goes:

```
1000 REM == load/display double high-res ==
1010 DIM B(7): REM use for parsing byte to bits
1020 TEXT : HOME
1030 IF PEEK ( - 1101) < > 6 THEN PRINT "Requires 128K Apple IIe/c/gs.": STOP
1040 GOSUB 1460: IF (MEM < 128) THEN PRINT "Requires 128K!": STOP
1050 PRINT "Checking for empty /RAM..."
1060 PRINT CHR$ (4);"BLOAD /RAM,A$2000,L512,TDIR"
```

```
1070 IF ( PEEK (8192 + 37) + 256 * PEEK (8192 + 38)) THEN PRINT "/RAM not
     empty!": STOP
1090 REM — the following enables double high-res —
1100 PRINT CHR$ (4);"PR#3": REM activate 80-column firmware
1110 PRINT : REM send something through COUT to initialize firmware
1120 REM (above also initializes 80STORE)
1130 HGR : REM activate high-res
1140 PRINT CHR$ (4);"BSAVE /RAM/HR.X,A$2000,L$2000": REM reserve DHR page in
     aux mem
1150 POKE 49246,0: REM activate double high-res
1160 REM (You may want to select "full page" here, too.)
1170 REM For more information, see the appropriate Hardware Reference
1190 REM — You have to figure out which format —
1200 REM Use T = 1 for Beagle Graphics
1210 REM     T = 2 for TimeOut Paint
1220 REM     T = 3 for Dazzle Draw
1230 REM One of these will probably work. Otherwise...
1240 REM ...you'll have to figure it out how to load it yourself.
1250 REM put filename into F$
1260 T = 2: REM Our example is a TimeOut Paint file
1270 F$ = "TESTPIC"
1280 REM Line 1340 calls subroutine appropriate to the file format in T.
1290 REM In all cases, we:
1300 REM  a) BLOAD the aux mem section to main mem high-res page 1
1310 REM  b) BSAVE the main memory page 1 to aux mem file /RAM/HR.X
1320 REM     (This works due to the way ProDOS treats the /RAM device!)
1330 REM  c) BLOAD the main mem section of the picture.
1340 ON (T) GOSUB 1600,1680,1760
1350 POKE - 16368,0: WAIT - 16384,128: POKE - 16368,0: REM wait for keypress
1370 REM — following disables double high-res —
1380 POKE 49247,0: REM deactivate double high-res
1390 PRINT CHR$ (4);"DELETE /RAM/HR.X": REM remove our file
1400 PRINT : PRINT CHR$ (21): REM turn off firmware
1420 REM — outa here —
1430 TEXT : HOME
1440 END
1460 REM — get memory size from ProDOS (MACHID) —
1470 MACHID = PEEK (49048): REM $BF98
```

Ask (or tell) Uncle DOS

Playing slots (again)

Normally, when you go to the Control Panel and create a RAM disk, an icon called RAM5 appears in the Finder. I have had a RAM disk icon in Finder, along with the hard drive icons and the 5.25 icon, for years.

The problem started when I added the Apple 3.5 Disk Controller Card to my system with the SuperDrive. Now all of a sudden, when I switch slot 5 from "Smartport" to "Your Card" the RAM disk icon called RAM5 disappears. I may have a RAM disk, but without the icon how do I know?

I have had to split my drives up to be able to run all programs and have the option of keeping my RAM5 disk in the Finder. The SuperDrive is still connected to the controller card but the original drive is back to the Smartport hookup. Any suggestions on how I can daisy-chain my 3.5 drive and still have access to the RAM5 disk option and have Copy II Plus recognize the drives? I'm tired of switching to slot 5 and rebooting every time I need to change directions in my work.

Rusty Schneeflock
St. Joseph, Mo.

As comedian Steven Wright has said, "You can't have everything...where would you put it?"

The IIgs Disk Port requires some built-in programming ("firmware") assigned to slot 5 to support any attached 3.5 disks; this firmware also supports the RAM5 volume. The built-in hardware and firmware was only designed to support the 800K (UniDisk 3.5 and Apple 3.5) drives, the /RAM5 disk, and (using additional firmware assigned to slot 6) Apple 5.25 drives. Drives that sufficiently resemble these drives also will work; there isn't a third-party UniDisk 3.5 clone (unless you count Chinook's SmartPort-attachable hard disk), but companies like Applied Engineering and AMR make alternatives to the Apple 3.5 and 5.25 drives (though we know of one minor glitch with non-Apple 3.5s; the IIgs will wait indefinitely when trying to read the drive when no disk is present).

You need additional hardware and firmware to support the SuperDrive, which is why you need the new Apple Disk 3.5 Interface card. But if you install this card in slot 5 in the IIgs, it has to be activated in place of the internal slot 5 ("SmartPort" support). When you disable the internal support, you disable its RAM5 support. That's why RAM5 is unavailable.

As you've seen, you can't have slot 5 set to both options at once. You have to decide which option is more important to you and stick with it, or rearrange your slots to accommodate the 3.5 interface in a different slot, or switch back and forth. This is life.

*If you have an available slot you can install an Apple II Memory Expansion Card ("Slinky" card) or work-alike such as the Applied Engineering **RamFactor** to get a RAM disk back without moving the 3.5 controller. Alternatively, you could move the controller to a slot other than 5; properly written programs shouldn't care, and ProDOS 8 will "phantom" a third slot 5 drive to another slot anyway.*

Assuming you have just these devices, slot 2 can be set to "Your Card" and the 3.5 controller can be installed there. Or put in a Slinky card for a RAM disk of up to one megabyte. (If you like the latter idea, we've got 2-3 of these cards around here that we'd sell.)

*Due to a problem we located with the Apple SuperDrive Controller and the AMR 3.5 disks (you can't do a low-level format of an 800K disk in the **AMR 3.5**, though it will read and write them fine), I spent some time working on an update to Tom Weishaar's original article on Smartport device identification. Look for some more information in the future.—DJD*

multi MIDI; advanced BASIC

Congratulations to Dennis Doms on his articles on MIDISynth and MIDI generally. At last I know what Multi mode is and that is the way my synthesizers work. I hope there are to be more such articles and I look forward to reading them.

W. J. Pattison
Adelaide, S.A.

Once you have the basic components of MIDI on your IIgs you may want to scout around for other music software. We've found a couple of programs of professional quality and several less complete (but also less expensive) programs for experimentation.

***At the professional level, choices are strictly limited.** One thing you may want is more capable sequencing software. The best we've seen is **Master Tracks Pro** from Passport designs; there are actually IIgs and IIe versions. If you find the $399 retail price frightening, there is a lower priced **Master Tracks Jr.** for about $99. Unfortunately, both of these programs have been discontinued by Passport Designs.*

*The other professional tool is software to produce actual musical notation (scores). The best we've seen for this is **MusicWriter** from PyWare (800-222-7536), which is actually a series of three products that can produce scores of varying complexity. **MusicWriter** doesn't require MIDI, it can also be used to edit scores directly. MIDI just provides an alternative method of communicating with the program. (As with several IIgs applications, the use of an accelerator will greatly reduce the frustration factor of using this graphics-intensive program.)*

```
1480 BYTE = MACHID: GOSUB 1520: REM parse byte into bits
1490 MEM = 0: IF (2 * B(5) + B(4)) = 3 THEN MEM = 128
1500 RETURN

1520 REM — parse byte into bit array —
1530 FOR I = 0 TO 7
1540 NBYTE = INT (BYTE / 2)
1550 IF (BYTE < > (NBYTE * 2)) THEN B(I) = 1
1560 BYTE = NBYTE
1570 NEXT I
1580 RETURN

1600 REM — Format: Beagle Graphics (two files) —
1610 REM Main mem half is in F$
1620 REM Aux mem half is in F$.AUX
1630 PRINT CHR$ (4);"BLOAD ";F$;".AUX,A$2000": REM load aux portion to main
     memory
1640 PRINT CHR$ (4);"BSAVE /RAM/HR.X,A$2000,L$2000": REM save it to aux
1650 PRINT CHR$ (4);"BLOAD ";F$;",A$2000": REM load main memory portion
1660 RETURN

1680 REM — Format: TimeOut Paint (single file) —
1690 REM Main mem half is second half of F$
1700 REM Aux mem half is first half of F$
1710 PRINT CHR$ (4);"BLOAD ";F$;",A$2000,L$2000,B$0": REM load aux portion to
     main memory
1720 PRINT CHR$ (4);"BSAVE /RAM/HR.X,A$2000,L$2000": REM save it to aux
1730 PRINT CHR$ (4);"BLOAD ";F$;",A$2000,L$2000,B$2000": REM load main memory
     portion
1740 RETURN

1760 REM — Format: Dazzle Draw (single file) —
1770 REM Main mem half is first half of F$
1780 REM Aux mem half is second half of F$
1790 PRINT CHR$ (4);"BLOAD ";F$;",A$2000,L$2000,B$2000": REM load aux portion
     to main memory
1800 PRINT CHR$ (4);"BSAVE /RAM/HR.X,A$2000,L$2000": REM save it to aux
1810 PRINT CHR$ (4);"BLOAD ";F$;",A$2000,L$2000,B$0": REM load main memory
     portion
1820 RETURN
```

These products also exist in versions for the IIe using Passport Design's (also discontinued) IIe MIDI interface .

Many of the less expensive "hobbiest" products are available through The Big Red Computer Club (402-379-4680). *These are programs that may satisfy many users though lacking professional features. They also include a group of programs that use MIDI as an alternative sound input and output channel.*

Electronic Art's ***Music Construction Set*** *allows you to enter your music by placing notes on a musical staff and then playing them. Constructed songs can be played back through the IIgs speakers (using two instrument voices, one "treble" and one "bass", for the notes on the respective staves) and MIDI via the modem port. You can also play the voices using the MIDI keyboard but the notes aren't added to the staff.* ***Music Construction Set*** *has no means of selecting MIDI configuration and there's no mention of MIDI in the manual. In addition, the program runs as a ProDOS 8 application and doesn't implement standard IIgs interface features (such as a "quit" option from the menus).*

Activision's ***Music Studio*** *also allows you to enter notes on a staff and play them through both the IIgs speaker and MIDI port (modem port only), but the MIDI implementation is much more complete (and documented). Through the "MIDI parameters..." item of the "Options" menu you can set up up to 16 instruments and assign their basic MIDI channel, the preset (voice program) number to use, and a range of notes that will be accepted. The MIDI parameters are saved as part of the file so you can archive them with your music. This gives you more control over the instruments; you can actually play several instruments through the use of program changes to alter voice assignments (in the MIDI device playing the data) or by assigning output to play back on more than one unit (by assigning the basic channels for different voices).*

Music Studio *also has several non-standard quirks that cause headaches. The user interface is barely recognizable as a IIgs desktop implementation (it really isn't one), even failing to call the Desk Manager so that New Desk Accessories (and the New Control Panel) show in the Apple menu.*

It uses its own MIDI driver (APPLEMIDI) that assumes use of the modem port for output instead of Apple's (APPLE.MIDI) driver that can be assigned to either the printer or modem port. (Things work much better if you create the "APPLEMIDI" driver by duplicating the 6.0 "APPLE.MIDI" driver rather than copying over the driver supplied on the ***Music Studio*** *disk. (also try patching the name in Music.Sys16)).*

I had to disable the ***ZipGSX*** *before launching Music.Sys16 to get the MIDI functions to initialize and recognize the synthesizer (once the program has started, you can go into the Classic Control Panel and re-enable the* ***ZipGSX*** *using its CDA).*

Also, the pathname for a "WAVES" folder that ***Music Studio*** *uses is hard-coded so that it must exist in the root directory. (The installation procedures supplied with the Big Red Computer Club version are careful to point this out.)*

The program assignment values for ***Music.Studio*** *correspond to one plus the actual program number (program 1-128 in* ***Music Studio*** *translates to 0-127 for MIDI).*

In general, there are many things wrong here. But if some of the glaring problems could be "patched out" (or patched in) this program might be the best economical MIDI application out there.

Several of the remaining programs are "performance oriented." *From Electronic Arts there are* ***Instant Synthesizer*** *and* ***Instant Music****.* ***Instant Synthesizer*** *is a simulation of a sampling synthesizer which can play back tracks of music and assign voices from digitized sound files. One of the more hilarious examples is a fugue using animal voices (dogs barking, ducks quacking, lambs bleating). When loading the supplied songs, it physically goes out and checks the 3.5 drive for every file, even when installed on a hard disk (because the full pathnames are imbedded in the "package" file that loads the instruments).* ***Instant Synthesizer*** *supposedly supports MIDI devices, but I couldn't get it to work with the current system software.*

Instant Music *simulates a background combo in several styles (rock, country, classical, jazz, or make your own) playing a cyclic accompaniment that you can "jam" with in a "music minus one" style.*

The major weakness of most of these programs is their relatively limited use of the Ensoniq's sound capabilities in terms of the sampled waveforms provided. Most are capable of decent sound, but many of the standard instruments just don't sound that good. More recent programs (also available through Big Red) from some of our overseas friends fare better.

Oliver Goguel (of the Free Tools Association) programmed ***NoiseTracker****, a sequencer that runs from ProDOS 8 but requires a IIgs since it uses the Ensoniq. It doesn't include MIDI capability, but it does provide the ability to import and play Amiga "MOD" files. This file format is apparently popular; MOD players for both the Mac and PC compatibles (using a standard sound card like the* ***SoundBlaster****) have been spotted.* ***Noisetracker****'s sound reproduction abilities are impressive; the intro music is stunning.*

Hübert Alber's ***SoundSmith*** *is another sequencer program with stunning sound and it does support MIDI output with channel assignment in Omni, Poly, and Multi modes. You can enter data manually; I didn't find a way to use MIDI for input.*

I had two problems: the program seemed to output MIDI signals a few octaves off from the synthesizer (the synth plays C#2 when SoundSmith lists C#5), and occasionally a MIDI voice would be left in a "Note on" state (as if the controller didn't quite shut down all the voices). The latter may be a feature; it occurs when you drag from one note to another on the

screen keyboard and clicking the sustained note stops it. But the 'Stop Sound' command won't, and often doesn't stop all the notes when used to halt the 'player'.

There are at least a couple of companies involved in musical applications of computer systems including the Apple II. One is ***Sound Management, Electronic Music Products,*** *P.O. Box 3053, Peabody, Ma. 01961, 508-531-6192. This company provides product sales and advice on computer music products, MIDI interfacing, and so on.*

Another is Creative Consultation Services (719-687-2210) which supports PyWare and other products. —DJD

Not in Kansas anymore

A2-Central On Disk for November 1992 has a new IIgs *ShrinkIt* program. The new *GSHK* works just fine. I used it to unshrink the GSHK.DOCS file which appears to include new information because it is a couple of kilobytes larger than the former GSHK.DOCS file. Unshrinking it was no problem with the new *GSHK* but when I went to print it, using AppleWorks at first, I got an error indication and the file appears as only 1K to AppleWorks.

I tried using *Sneeze* to print it and got nowhere with it; it kept coming back to the main menu of Sneeze. So I tried to read the file using *Copy II Plus* (v8.2) and "View Files in Text" mode. I got an "I/O ERROR BLOCK $76AE". I guess the new *GSHK* works fine but the documentation file has been messed up in some way.

Could I please get a copy of the GSHK.DOCS file that I can use? Thank you!

George Milonas
Tacoma, Wash.

Okay, everyone who bought a IIgs because you wanted GS/OS features, take one step forward. Now, everyone who's still using ProDOS 8 applications to manipulate GS/OS files, take one step back...

It could be a little more apparent, but the GSHK documentation is actually in a ***Teach*** *file, not a text file.* ***Teach*** *is a IIgs format that supports text with formatting codes (text styles, font types, and so on), and it may not be unusual to see IIgs programs supplied with a documentation file in this format since the* ***Teach*** *application was supplied as part of System 6.0. The reason for adding* ***Teach*** *to 6.0 (as for adding* ***TeachText*** *to the Mac system software) is to give Apple developers a "universal" file format to distribute program documentation in. Not everyone has AppleWorks, some people even have trouble finding a utility to display text files if they're not using a word processor that can convert them.*

Developers probably won't supply ***Teach*** *since it requires 6.0 to run; if you have 6.0, you have* ***Teach****. If you don't have 6.0,* ***Teach*** *files can be read by most IIgs word processors including the one in* ***AppleWorks GS*** *and many of the IIgs text editors distributed as user-supported programs.*

Apple II users upgrading to the IIgs are going to have to be careful to avoid using ProDOS 8 programs on GS/OS files. Although GS/OS supports the ProDOS file system it also supports some enhanced features and at least one new file storage type (the "extended" files with resource forks) that ProDOS 8 does not have any intrinsic ability to handle. ***Copy II Plus*** *apparently tries to follow the file physically (block by block); since the extended file format does not use the same indexing as a "normal" ProDOS file.* ***Copy II Plus*** *misinterprets the index and looks for blocks in impossible places. That's where the "I/O error" came from.*

Reading a file can't cause serious problems as long as the program traps errors. Copying files or manipulating data on your disks can be disastrous if the program makes bad assumptions and doesn't catch errors. When IIgs System 5.0 arrived with extended files many people tried copying system files to their hard disks with their old utilities and ended up with corrupted systems that wouldn't boot. If you are using disks with GS/OS files it is essential that you use only GS/OS utilities to manipulate those files.—DJD

Words over wires

Thanks for your treatment of *Apple Writer II* in your October newsletter. I use it exclusively in my law firm and have for many years. It networks, and the features you describe are great. However, I am very interested in your comments on page 4 that there is a built-in telecommunications module that will work with a modem. I have an Applied Engineering *Datalink* 2400. How do I activate it without leaving *Apple Writer II*? If I print to the slot where the modem is, nothing happens.

Do I need to make special settings off the Control-O menu (choice J) first? If so, what, and then what do I have to do to engage the modem. What other module features exist? I have the original documentation for both the DOS 3.3 and ProDOS versions of *Apple Writer II* and can find nothing on telecommunications.

Brian C. Sanders
Fort Walton Beach, Fla.

The telecommunications capability of ***Apple Writer II*** *is limited to that of a dumb terminal. There are a lot of problems with its compatibility, though; it doesn't like our IIgs at all (even when using a Super Serial Card in slot 2) and it may be selective about the serial interface it wants to use (for example, it doesn't work with the Hayes* ***Micromodem II****, if any of you out there still have one).*

To set it up, type control-O for the "Options" menu, select "J" for "Set Printer/Modem Interface", and pick slot 1 or 2. Then you'll be asked to enter a format string (baud rate, data bits, parity, and stop bits). We tried "2400,8,N,1" (to use the default speed of the interface, use "0" as the baud rate).

To connect, type "control-Q" to go to the "Quit" menu and select "I" ("Connect Keyboard to Printer/Modem"). From here, you type commands to the serial port firmware (and modem) as if you were in BASIC or in a very dumb terminal program. For example, to dial you might type "AT DT<number>" for an AT command compatible modem.

There are four "Escape" commands; you issue any of these by typing the Escape character followed by the first letter of the command. "R)ecord" copies received text to the word processor's buffer. "E)cho" echoes what you type (toggles the duplex between full and half). "F)ilter" filters out control characters other than necessary ones like carriage return and line feed (highly advisable since receiving a firmware control code could crash your system). And "Q)uit" takes you back to the word processor screen.

"R)eceive" gives you the ability to capture text. You can send text to the modem by "printing" it to the modem slot, but this can be tricky since you won't have any way to control aspects of how the data is sent (the data is just "bulk dumped" to the serial port). There are no provisions for file transfers or other common features of real telecommunications programs.

If it works for you, the telecommunications support may be useful in situations where you need to talk to another computer at a very basic level and just don't want to leave ***Apple Writer****. Otherwise, you're better off ignoring it.—DJD*

A2-Central™

[redacted]

© Copyright 1992 by
Resource Central, Inc.

Publisher: **Tom Weishaar**
Editor: **Ellen Rosenberg**

with help from:
Denise Cameron **Dennis Doms** **Sally Dwyer**
Dean Esmay **Mary Johnson** **Jeff Neuer**
Becky Smith **Jean Weishaar**

A2-Central—titled ***Open-Apple*** through January, 1989—has been published monthly since January 1985. World-wide prices (in U.S. dollars; airmail delivery included at no additional charge): $34 for 1 year; $60 for 2 years; $84 for 3 years. All back issues are currently available for $2 each; bound, indexed editions of our first six volumes are $14.95 each. Volumes end with the January issue; an index for the prior volume is included with the February issue.

The full text of each issue of ***A2-Central*** is available on 3.5 disks, along with a selection of the best new public domain and shareware files and programs, for $90 a year (newsletter and disk combined). Single disks are $10.

Please send all correspondence to:

A2-Central
P.O. Box 11250
Overland Park, Kansas 66207 U.S.A.

ISSN 0885-4017
Printed in the U.S.A.

GEnie mail: A2-CENTRAL
Voice: 913-469-6502
Fax: 913-469-6507