A2-Central™

formerly Open-Apple

October 1990
Vol.6, No.9

ISSN 0885-4017
newstand price: $2.50
photocopy charge per page: $0.15

A journal and exchange of Apple II discoveries

Miscellanea

IBM has launched an assault on the home market Apple has long ignored. As Tom Weishaar mentioned in "Personnel moves rock Apple" (March 1990), Apple has repeatedly said it believes that there is a business segment and an education segment to the personal computer market, but that there is no distinct home segment. Apple believes that home computer sales fall out of those education and business sales.

This month IBM's new model PS/1 started appearing in stores and reports started appearing in the press. It is hardly state of the art: a 10MHz 80286 processor in a system that includes a 1.44 megabyte floppy disk, 2400 baud modem and mouse. The basic system is currently sold in three configurations; the floppy-based monochrome system ($1000), the same system plus a 30 megabyte hard disk, and a color system with hard disk ($2000). In deference to the first-time buyer, IBM has left the configuration decisions for the basic system simple: decide if you want a hard disk and decide if you want a color display. Those are the only choices you have to make.

The hardware is not imposing, but the implication is: Bill Machrone of *PC Magazine* reports in his August editorial that IBM believes there are 66 million households that can afford this type of system. That kind of potential has apparently dragged IBM salivating into a marketing strategy aimed directly where Apple isn't: presenting systems as attractive for a home market that is not directly tied to education or business.

Toward that end, IBM has tailored the PS/1 to be easy to use. You bring the system home in one box; the setup manual is packaged already opened to the page telling you how to set up the system. You attach the monitor, mouse, and keyboard, then run the power cord to the wall and the modem phone line to a phone jack. Then turn it on; the system prompts you from there, and help is always available. One of the options the machine automatically gives you is support via the *Prodigy* information service (which will be expanded nationally). Another is a limited MouseText-like version of *MicroSoft Works*, which is a clone, right down to the name, of AppleWorks. The version bundled with the PS/1 includes a word processor, a spreadsheet, a database, and a communications module. These make the system immediately useful.

IBM has even found ways to get around callous dealers interested in moving product but not helping entry-level customers. The units will be sold primarily through Sears retail outlets; IBM is putting this machine in the path of potential customers who would never enter a computer store. Rather than depend on dealer salespeople, IBM also offers PS/1 customers an 800 number available for basic technical questions.

We played with the units at a local Sears outlet; the system design prevents the user from ever seeing the intimidating MS-DOS command-line mode by accident (you can get there on purpose if you like). The system demo that was running on one of the systems did an admirable job of presenting attractive features of the system for home users; of course, much was made of the fact that IBM was bringing its office environment expertise "home". That IBM logo carries weight that will sway customers hesitant to buy an "off brand" MS-DOS compatible.

But the sad thing is that an Apple IIgs setting next to the PS/1 would be very competitive. The animations running on the PS/1 looked stilted. The sound quality, even with the optional audio card, doesn't even get off the ground compared to the IIgs. The screen has an annoying flicker when switching modes. The price is less than a comparable IIgs system, but then the IIgs pricing is set to allow for dealer discounting (it's unlikely Sears' pricing structure allows for dickering). And the pricing gap is close when you consider the capability and expandability built into the IIgs that costs extra on the PS/1 (expansion slots, sound capability, and so on).

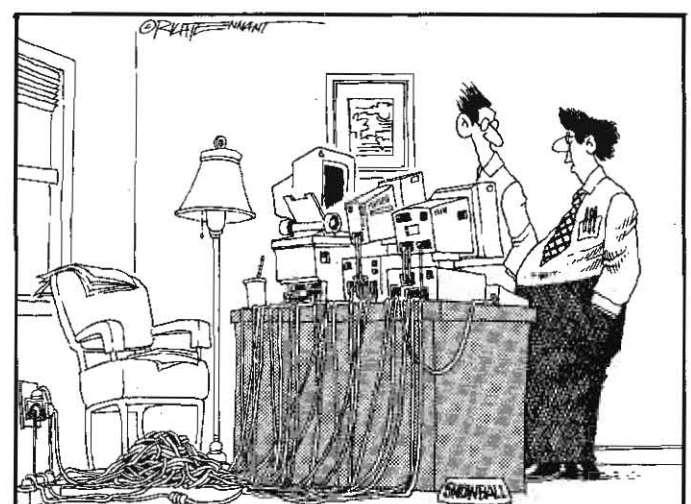
Apple could respond to the competition by bundling AppleWorks or *AppleWorks GS* (or both) with the IIgs, particularly now that Claris is back under Apple's total control. Apple could also have used *AppleLink Personal Edition* as its on-line support vehicle, though it's too late for that now. Apple continues to believe technology alone can carry the company; yet seems to self-destruct every opportunity to make it palatable to the public

The big disadvantages the Apple IIgs has compared to the PS/1 are distribution and software availability; both are closely tied to the myopic way in which Apple has marketed the Apple II line since the introduction of the Mac. Despite the fact that Apple's customers have clearly stated their desires and despite a four-year lead time, IBM has beat Apple to the market that the IIgs is perfect for. I hope Apple finds the results truly embarrassing.

Apple has announced a price drop on the Mac IIci. Price reductions of up to 20 per cent were announced for various configurations. That leaves the least expensive Mac IIci system price at $5,969.

Meanwhile, rumors of the new "low cost" Mac systems to be introduced in October seem to be converging on three models. The Mac Classic will be a replacement for the Mac Plus and Mac SE coming in at about $1,600 retail. The Mac LC will be a 68020-based system that accepts a color monitor and a future Apple II emulator card. And the Mac IIsi will be a 68030-based 16 MHz color system predicted to come in at about $2800. We'll see for sure in October.

This has been the month of IIgs teasers. Coming on the heels



"WE FIGURE THE EQUIPMENT MUST HAVE SCARED HER AWAY. A FEW DAYS AFTER SETTING UP, LITTLE 'SNOWBALL' JUST DISAPPEARED."

of demonstrations at the ***A2-Central Summer Conference***, we have seen several new programs written by a nucleus of true IIgs gurus. In many cases the programs are "under construction", but show that the IIgs is as (or more) powerful than many of us have believed.

First to be seen was *Modulae*, a demonstration of three dimensional animation from the FTA (Free Tools Association), a group of IIgs hackers based in Dijon, France. FTA accrued fame via their previous *Nucleus* and *"Space Harrier"* demos as well as the shareware disk utility, *Photonix*. Like their earlier productions (and somewhat to our chagrin), *Modulae* eschews use of the IIgs toolbox to the point where it is incompatible with GS/OS; it must be booted from its own disk and requires booting the system again to return to GS/OS. On the positive side, both the ROM 01 and ROM 03 IIgs versions are supported.

Modulae cycles through an introductory screen and demonstrations of star field effects, wire frame objects, objects composed of spheres, star flight effects, and use of the IIgs graphics "fill mode" to draw solid objects and in calculated animations. In the final sequence, your vantage point is from the perspective of flying through a maze with colored walls; suddenly you are teleported outside the "maze" to discover that you are flying around and through the FTA logo, which draws back and presents itself in a horizontal roll. Catchy music is flowing from the IIgs during all of the demos, and (by turning off the "automatic" display mode) you are allowed to play with the rotation and sizing of the objects.

Two other striking demos are from a new company still looking for financing called DreamWorld. (Contact Jason Anderson at 319-354-7959.) The first shows *DreamGraphics*, a GS/OS paint program. The second demonstration, *DreamVision*, is not GS/OS based and boots from its own disk. Both of these are only demos currently.

DreamGraphics is designed to create 16, 256, or 3200 color images. The 256-color (with some placement restrictions) mode is possible in the IIgs's 320x200 Super High Resolution graphics by using "dithering," a technique in which colors are mixed within a pattern to give the impression that the pattern represents a conglomeration of the two colors. Dithering is analogous to the use of small, closely placed colored dots to generate color newspaper photographs.

The simultaneous display of 3200 color images is possible in the IIgs, but it isn't easy. The IIgs SHR display includes a series of 16 *palette* values that define available colors, and *screen control bytes* that determine which of these palette values are used for each of the 200 lines of the display screen. By doing some carefully timed fancy footwork as the IIgs video hardware draws the display screen to your monitor, the 16 palette values can be changed as the screen is being drawn so that each line can have a distinct palette. 200 lines times 16 colors (with palette switching) available for each line gives 3200 available colors.

However, changing the palettes "on the fly" like this consumes almost all of the IIgs's processor time just to display the picture;

Ask (or tell) Uncle DOS

The IIgs promise

The latest issue of ***A2-Central*** prompted me to write and add my bit of confusion to the IIgs/Mac controversy. I am an engineering manager for a large aerospace corporation and naturally we depend heavily on computers, ranging from mainframes to "personal computers". Over the past four years Macs have been creeping into the engineering organization, due partly to my urging and example.

I have had an Apple II since about 1980 which I use to provide home analysis and independent review of the work provided by my engineers. For engineering applications, I need the ability to handle large arrays of data, and a good math package. I have been "making do" with Applesoft by virtue of the various compilers which have slowly come on the market. My magnus opus was an industrial grade performance program aggregating 40,000 lines of Applesoft in eight separate but related programs, one of which had 198 variables.

Last year when the time came to replace my trusty Apple IIe, I carefully considered the Macs (which I could purchase through the company for a significant discount), the IIgs, and those "blue" things. I finally bought a IIgs for one reason and one promise.

The **reason** is that I like the *TimeOut* enhanced AppleWorks word processor better than any word processor that I have used, including those on the Mac. It does not have all of the bells and whistles but it does about 90% of the things I want to do.

The **promise** was that of *ORCA/C*. I have been looking for a good compiled language for the IIe/IIgs family and this seemed to be it. I therefore elected to purchase a IIgs and soup it up with a hard drive, a *TransWarp GS* card, and a lot of memory.

Just as I had made the decision to purchase the IIgs Apple announced System 5.0, so I was able to wait a week and buy one of the new models. I now find that I have the best of both the text-based and graphics-based worlds. I use the text-based approach for word processing and technical programming. I use the graphical interface for desktop publishing. I nearly have my dream machine.

The one additional capability that I would like to have on the IIgs system is a good technical plotting program. *Graph-It!* and *TimeOut Graph* won't hack it. A not-widely distributed DOS 3.3 program called *Scientific Plotting Package* by CMI-Cascade comes close but the company is out of business and the package is no longer supported. The only other contender is the plotting capability of *SuperCalc 3a*, but it is somewhat cumbersome to use, and needs a patch which I don't have to make it work on the IIgs. I look with longing at *KaleidaGraph* and *DeltaGraph* on the Mac and wish that we had approximately the same capability on the II family.

Now in your previous issue you mentioned that someone demonstrated a 640x400 pixel screen at the ***A2-Central Summer Conference***. How do we get the message out to Apple that this is needed!? This Mac-like display capability should be the impetus for some good programmer to do the plotting packages I have been needing.

'Nother subject. I would like to send kudos to Mike Westerfield and his group of Byte Workers for the continuing development of *ORCA/C*. I have been treated most courteously when I bring up what I consider a problem to them in spite of being an amateur when it comes to C. The new version 1.1 release of *ORCA/C* is a vast improvement over the past one.

And speaking of kudos, ***A2-Central*** deserves one in my book. I always look forward to the arrival of the monthly issues. Please keep up the good work.

Sidney A. Powers
Valencia, Calif.

The problem with Apple's repressive Apple II marketing "strategy" is that they have forced most Apple II software companies to market almost entirely toward educational institutions. For many companies, this is the only venue they advertise in. Educational companies such as Scholastic do almost all of their sales via direct mail; that is, many of their products are not sold through national marketing agencies and may not show up "on the rack" at your local software store. Those with specialized needs like yourself should wander through a directory like the ***MENU Information Directory*** *to check into possible software sources.*

Meanwhile, we can supply you a copy of the ***SuperCalc*** *patch to allow it to print on an ImageWriter from the IIgs.—DJD*

More *SuperCalc* memory

If anyone has a patch to allow *SuperCalc 3a* to recognize additional memory in the IIgs, it would help me very much. *SuperCalc 3a* recognizes up to 512K on an Apple IIe auxiliary slot memory expansion card, but won't recognize my IIgs memory expansion card.

Paul T. Hogan
San Diego, Calif.

This is one that we don't have an answer for currently.—DJD

ProDOS 8 bad blocks

Will ProDOS 8 write information on bad blocks of a hard disk drive? I am concerned about a program writing information to a bad block and losing important information.

Douglas Bailor
Spokane, Wash.

The occurrence of a bad block and its affect on your file contents is timing-critical, from the time ProDOS has decided which block it is

some of that time has to be stolen back to allow your paint program to function. It appears from the demonstration shots in the *DreamGraphics* demo that the program compromises by letting you see and modify a subset (a horizontal band) of the image in 3200 mode, allowing you to gradually work your way through the entire image. It's a reasonable solution, and we hope the 3200 color image format and display techniques can be standardized (so far, not all images and display utilities are compatible).

DreamVision is more mysterious. The demo boots into a screen simulation of a video monitor and what appears to be a video editing controller (a device to allow you to cut, paste, and copy video sequences). The monitor then erupts into a brief sequence of a video recording of a tap dancer with with synchronized sound; the effect is spectacular, as if the IIgs had become a digital video tape recorder. Then an information screen is presented announcing that *DreamVision* is coming, and the demonstration loops. Some people like to say the Apple II is "ten year old technology"; put this demo next to 1980's *AppleVision*, with its stick-man dancing to a simple tune on a simulated TV screen, and you'll see what a difference ten years has made.

The last demonstration can also be launched from GS/OS; it's called *GS Experience* and displays a very impressive three-dimensional rotating "Z" that author Tim Meekins created ("rendered") using a superminicomputer and then transferred to the IIgs. The object appears to leap out of the screen at you, even reproducing the changes in reflected light that were recorded during its creation.

American Printing House for the Blind has announced a IIgs version of *TEXTALKER*. The IIgs version can run in the background, permitting its use with "friendly" copy-protected programs as well as many non-copy protected programs such as AppleWorks, Copy II Plus, and the Apple System Utilities.

The program installs as a ProDOS 8 application, but can remain system resident to work with other operating systems such as DOS 3.3 and GS/OS text applications. It requires a IIgs with at least 512K bytes of RAM, a 3.5 drive, and an *Echo* (or *DoubleTalk*, in *Echo* mode) speech synthesizer. The cost is $34.50 (catalog number D-89571-00).

We gleaned this tidbit from the *Raised Dot Newsletter* from Raised Dot Computing, 4085 Baldwin, Madison, Wisc. 53703, 608-257-9595 or FAX 608-241-2498, a newsletter covering microcomputer products for the visually impaired. Subscriptions are $18/year in large print, or $20 per year on audio tape (add $20/year air mail outside North America).

Checkmate has resurfaced under a new name: Micro Memory. The new address is 7655 East Gelding, #B1, Scottsdale, Ariz. 85260, 602-998-0227. Former Checkmate president, Earl North is still president.

---

going to write to until the data is actually written to the disk.

Before ProDOS can write a block to the disk, it has to locate the block. It does this by reading the block header (an "address" on the disk that identifies the block) as it moves the disk drive read/write head over the disk. If the header is unreadable and can't be found, ProDOS will report an error.

Once ProDOS has found the header, it tries to write the block. Looking at the code that does this for a 5.25 drive, it doesn't appear that the driver verifies that the data was actually retained by the disk. However, assuming that the disk had been previously formatted and the header was still readable, the odds are very good that the data is there.

ProDOS repeats this for every block in the file it is trying to write. If no unmanageable errors occur, your file is on the disk intact. If not, ProDOS will return an error and your program should warn you that the file was not saved.

However, that does not mean that the file will necessarily be intact the next time you read the disk. Magnetic fields or physical damage to the disk, among other things, can cause the data to fade. There may simply be a "weak spot" in the disk's magnetic coating; not weak enough to cause the write to fail, but too weak to retain the data written there between uses.

You can lessen the chances of some of these problems by keeping the disks away from hazardous environments, and (for floppies) by noticing which brands of disks seem to be the most reliable for you and sticking to that brand where possible. This won't remove absolutely 100% of the risk (there's always the small chance that the disk will decide to forget what it knows immediately after you write to it), but you can get pretty darn close.

Other than that, the only way to make sure you don't lose data is to make sure you have a backup of every file you consider to be important. ***If you fail to follow that simple rule, there is no disk recovery program, virus detector, or other utility on the market that can help you.*** *Period.—DJD*

Wrong number

I have problems in using the new ProDOS 8 v1.9 in conjunction with GS/OS 5.0.2. I renamed the PRODOS file to "P8" and copied it into the System folder. When I then launched a ProDOS 8 application with my *ProSel* selector, I got the message "incorrect ProDOS version". Can you help, or is the new ProDOS not compatible with GS/OS 5.0.2?

Gunther Unger
Emmench, West Germany

There seems to be confusion in the concept of IIgs System Software, GS/OS, and ProDOS 8.

The IIgs System Software is the entire package of GS/OS, ProDOS 8, the tools, drivers, and so on. The most recent version of this is 5.0.2.

GS/OS is the 16-bit native operating system of the IIgs. GS/OS itself is responsible for communicating with devices while the IIgs is in native (16-bit) mode and for managing the rest of the components of the system software (loading tools, for example). The version of GS/OS supplied with 5.0.2 is version 3, but there are several minor components all with their own version numbers. For example, in the change from System Software 5.0 to 5.0.2 the version number of the supplied ProDOS FST changed from 3.0 to 3.01.

ProDOS 8 is responsible for communicating with disk devices from the 8-bit "IIe emulation" mode of the IIgs. ProDOS 8 is also a component of GS/OS; GS/OS launches ProDOS 8 in order to run an 8-bit program. It is here that the version number of the ProDOS 8 "P8" file comes to matter to GS/OS; it needs to check the version number to verify that a version-specific patch can be made to P8's quit code so that it can return control to GS/OS . If you change the version of P8, it is no longer recognized as the "correct" version by GS/OS.

I've always recommended using the latest version of the system software for your system (barring contrary advice from Apple, which has happened only rarely in the last few years). I should now clarify that in saying "system software" I am referring to the ***complete*** *system software environment; individual components shouldn't be modified without Apple's recommendation.*

In other words, IIgs System Software 5.0.2 is the current release version for the IIgs, and you shouldn't modify any component files except per approved updates. This can be a modification to the operating system as released by Apple (the Apple High-Speed SCSI card comes with a minor revision to the IIgs system software used to support new features of the card) or a driver supplied by a third party in accordance with Apple's documented methods of enhancing the system software. However, changing files arbitrarily (even with Apple's own files) may not work to your advantage.

The new ProDOS 8 system disk works fine by itself. But don't use it to "update" the IIgs System Software; Apple has been releasing those updates independently.—DJD

Giving it the boot

I have an Apple IIgs. I would like to know how to configure a program disk so that, when run, it will bypass the Finder. I am trying to run an application from the desktop, but I would also like to know how to do this (from the beginning).

I have at least two applications that I launch from the desktop. To get there, I must use my system disk. Immediately after arriving at the desktop, I remove the system disk and insert, say, *PaintWorks Plus*. I then double-click on the *PaintWorks Plus* application icon and the program loads. During this process, I am continually asked to swap disks from the system disk to the *PaintWorks Plus* one. I know that I have only one drive, but I would like to know why all this swapping is necessary and if there is a way to avoid it?

John Guggenheim
Bryn Mawr, Penn.

There is no real magic to avoiding disk swaps. Unless the operating system software (GS/OS) and the program itself will fit into the 800K of disk space on current IIgs 3.5 drives, you must swap disks once or twice while

starting a GS/OS application on a single-drive system. The IIgs is not unique in this; both the Mac and MS-DOS machines are in a similar quandary.

To see if the program might fit on a single disk, try the following. Make a copy of your usual startup disk, and use the IIgs Installer (on the /System.Tools disk) to configure the disk with the minimum features you need to run the program at issue. There are no hard rules as to what items may not be needed, but common sense should dictate some; for example, a game that doesn't allow you to access the printer should indicate that you won't need a printer installed. When you've stripped everything that you know isn't needed, boot your new startup disk.

Now insert the disk with your program file and launch the program. Finder will "unload" itself and launch the new program. The program may require some of the IIgs tools that are not currently in memory, requiring the first swap back to the system startup disk. Then the program may have additional portions of itself to load, requiring a switch back to the program disk. In most cases, those two swaps should be adequate to get the program going.

If the program does launch and run, it indicates that the program may not require files to be added to the System folder of your startup disk, which greatly simplifies the detective work in finding out what files need to be added to your startup disk. Quit the program (more disk swaps) and boot your normal startup disk.

Insert your program disk and make note of the files that appear in the root (volume) directory of the disk that do not appear to be among the files found in the directory of the startup disk you created to use with that program (PRODOS, BASIC.Launcher, BASIC.System, System, AppleTalk, and Icons).

Insert the new Startup disk you've made for your program and, unless they appear on the program disk, drag the files BASIC.System and BASIC.Launcher into the trash. Now open the System/ folder and drag the file named START (which is actually Finder) into the trash can. Empty the trash, and note how much free disk space you have left. Eject the startup disk (don't drag it into the trash; use open-apple-E or the drive's eject button so that the dimmed image of the disk remains on the desktop.

Insert the program disk, select the files and folders that don't appear on your startup disk. To determine whether the files will fit, you can use the "Icon Info" item of the Finder's "Special" menu (for folders, the "Calculator" icon in the info box will sum the sizes of the files the folder contains). If there appears to be enough space, drag them onto the dimmed startup disk image. Follow the commands to swap disks until Finder has copied the program disk files onto the new startup disk.

Now check the startup disk to see if the program file's name ends in ".SYS16". If not, you'll need to rename it; for example, you may have to rename the file "MyProgram" to "MyProg.Sys16". Once the file is renamed, try booting the startup disk. Either it will work, or it won't.

If it doesn't work, and you received a "volume full" error message during the copy, then there isn't enough room on the 800K disk for GS/OS and the program. All you can try to do is remove files you believe the program doesn't need. Once you've removed as many elements as you can manage with Installer, you can try deleting other "flexible" components of the startup disk's system software. Here are the minimum file requirements needed to boot system software version 5.0.2 (reference: Apple II GS/OS Technical Note #1: Contents of System.Disk and System.Tools):

```
ProDOS
System/
  Start.GS.OS
  GS.OS.Dev
  Error.Msg
  FSTs/
    Pro.FST          required for ProDOS disks
    Char.FST
  Drivers/
    AppleDisk3.5     required for Apple 3.5 drives
    Console.Driver
  System Setup/
    CDEV.Init        required for Control Panel NDA
  Tool.Setup
    TS2              ROM 01 tool setup
    TS3              ROM 03 tool setup
    Resource.Mgr
    Sys.Resources
  CDevs/             as per your system configuration
  Desk.Accs/
    CtlPanel.NDA
  ExpressLoad
  Tools
  Fonts
    FastFont         speeds up Shaston 8
  P8                 required for ProDOS 8
BASIC.System         required for AppleSoft BASIC
BASIC.Launcher       required by Finder for AppleSoft
```

Comments indicate special considerations. Italicized items should be included unless infeasible. Here are some further comments.

The Control Panel NDA program is contained in the CtlPanel.NDA file and uses the CDev (Control Panel Devices) files as data; CDev.INIT loads the CDevs during boot initialization. If CDev.INIT is included the CDEVs folder must be present. If you plan to delete the Control Panel NDA (and consider that this is the only way to select certain functions of the system; the Alphabet, printer, and AppleShare selection protocols are not available in the Control Panel CDA) you should delete CDev.INIT, the CDevs folder, and the CtlPanel.NDA. If you are running a program that does not present you with an Apple menu where you can access the Control Panel NDA, these files are expendable.

ExpressLoad is a system utility that makes it possible to load files in a special compatible format much faster than normally; it should be retained if at all possible. Without it, the files will still load, but more slowly.

The use of files within the Fonts and Tools folders are determined by the program. The best rule of thumb is to see whether the program you are using comes on a bootable disk; if so, open these folders on the program disk and see what files are present. Leave those files in place in the corresponding folders on your startup disk. FastFont is a special form of the default system font (Shaston 8) that allows faster drawing of the font on screen; it should be retained if possible even if your program disk's Fonts folder is empty.

The remaining folders of interest are Desk.Accs, and System.Setup. If you've started with your "minimalized" startup disk, the contents of these folders should approximate the minimum configuration suggested above. Apple's tech note does not recommend it for disks developers are preparing (which need to be compatible with expected system configurations), but if you are creating a disk that will be booted only on your system you can omit the tool setup file not specific to your IIgs ROM version. Specifically, if you are a ROM 01 IIgs user you can delete TS3; for a ROM 03 IIgs you can delete TS2. Once you have done this, the disk will not boot on the "other" system.

These are all stop-gap measures; at some time; the IIgs's 65816 can theoretically address 16 megabytes as easily as the 6502 in earlier Apple II models can address 64K (the IIgs is restricted to using less than 16 megabytes of RAM by system design constraints). Just as Apple II programs such as AppleWorks and Pascal outgrew a single 140K 5.25 disk, IIgs programs are fast outgrowing the use of a single 3.5 disk. The solution is to add more drives, especially a high-capacity drive such as a hard disk. For the current crop of 16-bit (and larger) machines from all manufacturers a hard disk is becoming a near necessity.—DJD

Postscript downloads

I am able to create a *PostScript* file from *Publish-It! 3.0* and get it over to a Mac with a LaserWriter NT to successfully print text and Publish-It! line art. However, I get a *PostScript* error when the *PostScript* contains any graphic images.

Also, the printouts are such that I can't get anything to print in an area one-half inch from each paper edge. Is this a limitation in the LaserWriter printer or do I have to do something extraordinary?

Michael Gooding
Chesterfield, Mo.

*We talked to **Publish-It!** guru Bruce Rosenblum and he clarified some of the "features" of the **PostScript** file generation.*

*Most programs that print via **"PostScript"** actually download a **PostScript** program called LaserPrep that contains subroutines (written in **PostScript**) for common operations. For example, there is no inherent **PostScript** command to draw a rectangle on the virtual page, but a subroutine named "rc" contained in the LaserPrep file will draw the figure. When the computer wants to draw a rectangle as part of the printer output, it can then use the "rc" command.*

*One problem that can crop up is trying to install incompatible versions of LaserPrep. **Publish-It! 3.0** ships with a version of LaserPrep that is compatible with the version used by the IIgs System Software 5.0 and Mac System Software 6.0. When creating your **PostScript** document in **Publish-It! 3.0** by printing to a disk file, elect to include LaserPrep in the disk file when you know the LaserWriter has not been initialized (that is, no one else has printed to it since the LaserWriter was powered up), and don't include LaserPrep when you are going to be downloading the file to a LaserWriter that has been initialized.*

*If you are going to print a file to the LaserWriter via downloading, make sure you have at least one-half inch borders; the **PostScript** output won't support printing closer to the edge of the page than that (the "Larger Print*

Area" command in the Mac's Page Setup won't result in a larger print area for your "preformatted" document file). The absolute limit is about one-quarter inch in any case. If your original document is formatted for an ImageWriter, you may find the page width defined at 8 (rather than 8.5) inches with one-quarter inch borders; to prepare the page for the LaserWriter, widen the page width to 8.5 inches and center the print area within one-half inch borders.

*You should be aware that EPS (Encapsulated **PostScript**) files are different than the downloadable files that **Publish-It!** creates, so you can't print a file to disk and then import it as an EPS file.*

*Another problem you may run into at a service bureau is in getting them to handle an Apple II disk when all they have is Macintoshes. That's relatively easy; just be sure to bring your **PostScript** file on an 3.5 ProDOS disk and ask for a Mac that has Apple File Exchange installed. You can use Apple File Exchange to import the file from the ProDOS disk to the Mac via the text translation option.*

*All that having been said, we have run into some instances where a **Publish-It!** file containing an SHR graphic would generate a **PostScript** file that the LaserWriter gagged on. We've sent an example to TimeWorks to look at.—DJD*

Yummy? Ackk!

My daughter says the gum on the envelope should be apple flavored.

Jerry Giaccai
Canton, MA

Given the typical response we've gotten to the "yummy" glue, if you made her lick the envelope we're surprised she's still speaking to you.—DJD

Gliding mouse

Having to clean my mouse's ball once a week is a real pain. Do you folks know of anyone that makes an ADB compatible optical mouse that works reliably for the Apple IIgs?

For anyone else having the same problem, I find the best way to freshen a dirty ball is to do the following:

1. Open the mouse, remove the ball and clean it with alcohol, per the user's manual instructions.
2. Clean your mouse pad with alcohol.
3. Pour a small amount of alcohol on your mouse pad and play your favorite arcade-style game for about 15-20 minutes.
4. Wipe any excess alcohol from your mouse pad and mouse.

This may sound strange, but it works fairly well.

For the faint at heart, fear not; you won't get shocked or ruin your mouse doing this. Alcohol and Freon are commonly used to clean electronic equipment.

Bill Materse
San Jose, Calif.

A IIgs version of the A+ Mouse is being advertised by several mail-order companies. Its the only Apple II optical mouse we know of.—DJD

Mixed-up backup

Enclosed is a bulletin put out by Applied Ingenuity that says that the *Backup II* program (I believe v1.1) will not handle a hard drive with GS/OS 5.0 or higher.

Also enclosed is an article from the *IIgs Buyers Guide* about the Apple II High-Speed SCSI Card that says that a new version of *Backup II* (and this is v2.0) will handle 5.0 and higher.

However, the manual for the new High-Speed SCSI Card states that the utilities thereon, one of which is the new *Backup II*, will not work on earlier SCSI cards.

As the possessor of an older Apple II SCSI card purchased last March, this leaves me up a tree. What backup program can be used with earlier SCSI cards and System 5.0.2? Several programs are in my collection such as Easy Drive and *ProSel*, but which are suitable? Does Apple expect one to purchase another SCSI card every time they bring out a new operating system?

Incidentally, the documentation for David Hill's new version 3.0.2 of *JumpStart* says that the copy program included can correctly copy GS/OS files with any resource forks.

Robert Allison
Punta Gorda, Fla.

*Two available incremental GS/OS-based backup utilities that will handle System 5.0.2 "extended" files are **Salvation Guardian** (formerly just **Salvation**) from Vitesse and **ProSel-16**'s backup (see "Backup only", p. 6.14, March 1990).*

*Most ProDOS 8 utilities will not work correctly with extended files, and many will not even report an error when dealing with them (an "unknown storage type" error is feasible since extended files do return a unique storage type identifier when the file information is requested). Utilities that do not work on a file-by-file basis can be used; for example, the **ProSel-8** BACKUP utility that copies the entire used portion of a ProDOS volume block-by-block to a series of floppies or a file.*

*The only 8-bit incremental backup program we know of that works with extended files is Apple's **Backup II v2.0**. It is supplied with the new High-Speed SCSI card and is also supplied as part of the Apple II High-Speed SCSI Card Utilities from APDA ($30, part #A02422LL/A; currently classified as "beta" since it includes the beta versions of the Apple Scanner and Apple Tape Backup 40SC drivers for the IIgs). Although the manual cautions against using the new SCSI utilities on the old cards, Matt Deatherage at Apple DTS verified that **Backup II** (itself not actually a "SCSI" utility) specifically works with any ProDOS (but not AppleShare) volume, regardless of interface.*

The manual's warning is to prevent users from doing things such as using removable-media drives (which the new drivers will recognize) with the older Rev. C SCSI card, which doesn't work properly with some aspects of the new drivers. For example, I can eject and swap volumes on a SyQuest drive attached to the High-Speed SCSI card without problems; the Rev. C card becomes confused. Our experience is that a warning not to eject a disk with the Rev. C card may not be followed by a customer, so Apple probably decided not to support the use of the new drivers with the older card rather than deal with liability issues.

In fairness, and in response to complaints about the new card's price, it is only $129 including the new software. The Rev. C card is not functionally obsolete; the DMA version just adds some capabilities, while maintaining the same price (You don't get to upgrade cars when the new model comes out, though it would be nice.)—DJD

BASIC, C?

I ordered from your organization and received the Addison-Wesley Technical Library Set. It is most impressive. However, it is for someone much more advanced in computer techniques than I am. I have owned an Apple II Plus, IIe, and now the IIgs. Since the first Apple I have tried to learn the inner workings and have tried to learn (on my own) machine language programming—unsuccessfully. So when I got the library set I was rather frustrated to find that the textbooks gave examples on how to use the text in machine language and C. Is there any way that the ToolBox can be accessed via BASIC? The instructions in the Programmer's Introduction are rather convoluted and make little sense to me. Is it possible to get a translation into simple and understandable English of this "high-level explanation"? I cite "Calling from a high-level language" on page 65: "The interface libraries that allow C programmers to access the Apple IIgs ToolBox are included in *APW* C. Those libraries contain the function definitions for the tool...

"1. Make the routine accessible by using an #include statement that includes the appropriate file...

"2. Invoke the call by entering its name and supplying the correct parameters..."

Etc. What does it mean? Could I please get some help as to how to access the powerful ToolBox from BASIC?

Peter H. Smith
South Dartmouth, Mass.

Let's start with the concept of the ToolBox routines in general, then move on to the language-specific considerations.

*The ToolBox is a library of routines supplied by Apple with the IIgs and Mac system software. The specific routines for the two computers correlate reasonably well, making some of what you learn about the tools on one machine useful for programming the other. The purpose of the ToolBox (for either machine) is to provide standardized ways of accessing system features. Each routine is a "black box" that performs a specific function; all your program has to do to use the routine is to set up a list of information (or data structure) that the routine can use to perform its function, and then to call the desired routine. An example of the ToolBox call **ReadAsciiTime** to read the IIgs clock is presented in "Time to look in the toolbox", pp. 3.21-22, April 1987.*

Using these standardized routines provides two definite benefits. The more you use the ToolBox, the more your program is independent of the specific hardware features of the computer it is running on. This frees Apple engineers to make improvements in the computer hardware without having to worry about existing programs "breaking". By only requiring a list of data to be provided to the routine, the workings of the routine itself can also be "hidden" from your program, so that the ToolBox routines themselves can be improved without your program having to be re-written. Occasionally, completely new routines may be added that a program can take advantage of

*(**The Apple IIgs ToolBox Reference, Volume III**, is full of these), but the important factor is that existing programs don't have to (indeed, shouldn't, if properly written) fail to work as the system is upgraded.*

The third benefit also has a dark (well, slightly gray) side: by using standardized routines, most programs will have a similar "look and feel". Programmers should feel encouraged to use the existing tools for the reasons in the previous paragraph, and because they are there in the system waiting. By encouraging programmers to do this, Apple gets the benefit of making their computers appear easier to use by "unifying" the look of programs that run on the system. Sometimes there is too much uniformity, which can be a bit boring, so the recent upgrades in IIgs System Software 5.0.2 include some new functionality to allow programmers to design more "independent" elements for the user interface.

*That's the ToolBox philosophy. The implementation of accessing the ToolBox within a specific language varies, but basically it consists of building the data structure that the ToolBox routine requires and then calling the routine. Some languages, like Pascal and C, allow calling a function (or procedure) and passing data to the function as it is called. Most BASIC implementations, like Applesoft, don't do this and a method of passing the data must be contrived. Some extended BASIC implementations, like **Micol Advanced BASIC**, do have a function-calling mechanism that allows passing data items. In any case, the eventual form of the call is broken down into machine language and executed.*

*The ToolBox manuals present the machine language and C forms of the calls because these are the high-level languages supported by the **APW** tools. The first line of each definition in the manual gives the function number and tool set number in the form of a four-byte hex word ("$0F03" for function "$0F of tool set "$03") followed by Apple-specified name for the call ("ReadAsciiTime"). A description of the function follows. Below that is a "Parameters" section that shows what values are passed on the IIgs stack, and what values are returned. Next is an "Errors" section that reports what error values may be returned.*

Next comes the "C" version of the call. The C calls are actually converted to a Pascal calling convention (which is why you continually see the declaration of the "C" function as a "pascal" call in the ToolBox References).

*IIgs high-level languages usually are enhanced with some way of executing a "generic" ToolBox call, and then a series of interface routines are built upon the call. The **APW**, **Merlin**, **Complete** (formerly **TML**), **ORCA**, and some other languages for the IIgs include such interfaces as part of the language package.*

*Let's use ReadASCIITime as an example, since Tom Weishaar has already described the machine language form (p. 3.22). In C, the form of the call is given as (page 14-14 of **The Apple IIgs ToolBox Reference, Volume I**):*

```
extern pascal void ReadASCIITime(bufferPtr)
pointer bufferPtr;
```

The first line defines a C function "ReadASCIITime" that accepts the data item bufferPtr (a pointer to memory that you have reserved to accept the ASCII data) and returns nothing ("void" in current C parlance). "extern" means that the function is defined somewhere besides the current C file (the one you're using the function in), and "pascal" refers to the fact that "bufferPtr" is to be passed to the ToolBox using Pascal, not C, conventions.

The second line tells us that "bufferPtr" is a variable of type "pointer"; this tells the C compiler how large the data item is so the compiler can treat it properly. By knowing the type, the compiler can also do some checking for errors in our program (for example, if you try to assign the pointer value to an integer variable, the compiler can warn you).

*Here's a trivial program using ReadAsciiTime in **ORCA/C**:*

```
#include <stdio.h>
#include <misctool.h>

int main(void) {
   char Buffer[21];            /* allocate 21 bytes */
   pointer OurBufPtr = a;  /* set pointer to buffer */

   Buffer[20] = '\0';  /* set last byte to "null" */
  ReadAsciiTime(OurBufPtr);  /* get time/date string */
  puts(OurBufPtr);           /* and display it */
  return 0;          /* return "no error" to shell */
}
```

For simplicity, this program runs in text mode from the ORCA shell; all we're trying to do is look at an isolated example of one tool call.

The reason you probably didn't grasp "#include" is that it doesn't have an equivalent in Applesoft. The C compiler includes a preprocessor that looks for certain patterns in the source text and performs certain actions before the actual program compilation takes place. "#include" accepts a filename and causes the preprocessor to locate and load the named file. If the file is enclosed in "<>" characters, the preprocessor looks in a specific defined directory; if the name is only enclosed in quote marks, the preprocessor looks in the current directory.

"#include <stdio.h>" tells the preprocessor to include the contents of the file "stdio.h", which contains definitions used for standard input/output operations (we need this to access the display screen). "stdio.h" is a part of the standard C language library, and details on its use are included in any book on C.

*"#include <misctool.h>" is the IIgs-specific operation that the **Programmer's Introduction** was referring to in item 1. The ToolBox function ReadAsciiTime is not inherent to C, so we need to have it defined as a C function before we can use it in our C program. That definition (similar to the example in the ToolBox Reference) is in the file "misctool.h" that is included with both current Apple IIgs C compilers (**APW C** and **ORCA/C**). We use "#include" to copy that file into our program source so that we have access to use ReadAsciiTime later in our program.*

*If you know C, you can figure out the rest; if not, there are several good books that can help you with the overall program structure (among them **The C Programming Language** by Kernighan and Ritchie, and either of two books by Kelley and Pohl, **A Book on C** or **C by Dissection**). Our program contains one function, main, that creates a buffer for the ASCII data returned by ReadAsciiTime, sets OurBufferPtr to the buffer address, and stores a null byte ('\0' in C parlance) into the last byte of the array (C expects strings to be null-terminated in most cases).*

Next we pass OurBufferPtr to the ReadAsciiTime function defined in misctool.h. This corresponds to item 2 of high-level language access: calling the named function with the appropriate parameters.

Finally we use the C puts() function from stdio.h to print the string to standard output (in this case, the IIgs text screen "console"). "return 0" sends a zero to the ORCA/C shell as the error return code; we've defined main as returning a value of type "int" (integer), which the shell expects.

This is about as brief an introduction to C as I can manage in an attempt to show what the Apple IIgs ToolBox Reference definitions mean. What I hope it conveys is how the definition in the manual translates into the syntax of a real program.

Now let's try translating ReadAsciiTime into a BASIC dialect. Here's a similar program in Micol Advanced BASIC for the IIgs:

```
PROGRAM PrATime

DIM Buffer(20)

PROC ReadAsciiTime[OurBufPtr]
  MSW = INT(OurBufPtr/(2^16))    {get bank into MSW}
    MSW% = MSW
  LSW  = OurBufPtr - (MSW*(2^16))
        {get address into LSW}
  LSW% = LSW
  TOOLBOX(3,15: MSW%, LSW%)
ENDPROC

PROC puts[OurBufPtr]
  LocalPtr = OurBufPtr
  REPEAT
    CHAR = PEEK(LocalPtr)
    CHAR$ = CHR$(CHAR)
    IF CHAR = 0 THEN CHAR$ = CHR$(13)
    PRINT CHAR$;
    LocalPtr = LocalPtr +1
  UNTIL CHAR = 0
ENDPROC

OurBufPtr = ADDR(Buffer())
POKE OurBufPtr+20,0
GOSUB ReadAsciiTime[OurBufPtr]
GOSUB puts[OurBufPtr]
PRINT
END
```

***Micol BASIC** requires that we start our program with an identifying name, so we used PrATime (print ASCII time). We allocate a 21-byte buffer (imaginatively named "Buffer") for the ASCII data. Then we define two procedures (bracketed by "PROC" and "ENDPROC") that we'll use later.*

The main program starts by using Micol's "ADDR" function to return the starting address of the array Buffer, assigning it to OurBufPtr, and setting the last byte of the array to zero. Then we call the procedure ReadAsciiTime, passing it the value of OurBufPtr.

***Micol BASIC** doesn't include interface files for the ToolBox, so we have written our own interface function. ReadAsciiTime accepts a pointer to a buffer, and we break it into two words (two-byte integers) representing the bank number and the address of the array within the bank. We pass these to the ToolBox via Micol's TOOLBOX function, which accepts the tool number ("3" for the Miscellaneous Toolset), tool function number ("15" for ReadAsciiTime within the Miscellaneous Tools), and (following a colon) the parameters (a word*

at a time) in the order that they need to be pushed onto the 65816 stack to be passed to the tool. In our ReadAsciiTime call, we pass the pointer as the most significant word (MSW%) followed by the least significant word (LSW%) of the four-byte OurPointer value. It is here that we need to know how the ToolBox accepts the data in assembly language format; that knowledge is required to develop the form in which the data needs to fit into the TOOLBOX parameter list.

When ReadAsciiTime() returns, we call our procedure puts() to print the string returned in our buffer. Via a REPEAT—UNTIL loop, puts() PEEKs successive bytes out of our buffer and prints each one, converting the terminating "0" value to a carriage return (ASCII 13) and aborting (the UNTIL condition fails).

It gets very tedious to sit down and write the interfacing code such as our ReadAsciiTime example each time you want to use a ToolBox function. That's one reason I (personally) prefer C or Pascal; you have a high-level language that includes the necessary interface files, ready to go.

A third-party company, Codesmith Software, is working on a set of Micol BASIC procedures to aid in accessing the IIgs ToolBox: Tthe ***Codemaster Collection ToolBox Library***. Codesmith Software. Codesmith also has started a ***Micol Advanced BASIC User Group***, Membership Dept. #1316, 107 E. Vallette, Elmhurst, Ill. 60126, that shares source code and tips with members (this group is not affiliated with Micol Systems). New members receive a library disk of ToolBox procedures, such as routines to allow using 16 dithered colors in 640 mode on the IIgs SHR screen,

And if you want to hack at ToolBox access from Applesoft, So What Software's ***CallBox*** (see "GS/OS Resources", pp. 6.5-6, Feb. 1990) includes the tools to do it.

Some of the data structures used by the IIgs ToolBox get quite complex. The other advantage of C and Pascal is that each language allows description of more complex data structures than typical BASIC. For example, here's a window record definition as a C structure:

```
#include <window.h>
  /* where ParamList structure is defined */

ParamList RemoteWin = {  /* new window record */
  78,                    /* paramLength */
  0xC0A0,                /* wFrameBits */
  title,                 /* wTitle */
  0L,                    /* wRefCon */
  {0,0,0,0},             /* wZoom */
  NULL,                  /* wColor */
  0,0,                   /* wYOrigin,wXOrigin */
  0,0,                   /* wDataH,wDataW */
  0,0,                   /* wMaxH,wMaxW */
  0,0,                   /* wScrollVer,wScrollHor */
  0,0,                   /* wPageVer,wPageHor */
  0,                     /* wInfoRefCon */
  0,                     /* wInfoHeight */
  NULL,                  /* wFrameDefProc */
  NULL,                  /* wInfoDefProc */
  NULL,                  /* wContDefProc */
  {30,20,190,260},       /* wPosition */
  (void *) -1L,          /* wPlane */
  NULL                   /* wStorage */
};
```

The comments (bracketed by "/*" and "*/") refer to the names of the "fields" within the record RemoteWin as defined in the C header file "window.h". Accessing an item requires only that you use a period to separate the name of the field from the name of the record, so that "RemoteWin.wPlane"

If you prefer using simple arrays rather than abandoning BASIC, however, that option is open to you. You'll need to access the elements of the array "manually" by determining the offset to the item you need; you could define the offsets in "standard" variables to help systematize things.

Incidentally, several readers have also asked where they can get started learning C or Pascal on the IIgs. The Byte Works (4700 Irving Blvd. NW, Suite 207, Albuquerque, N.M., 87114, 505-898-8183, GEnie BYTEWORKS, America Online MikeW50) has been offering on-line and correspondence courses in C and Pascal featuring their languages (***ORCA/C*** and ***ORCA/Pascal***) on the IIgs. The lessons take you through the elements of the language up to the point where you can start concentrating on learning the IIgs ToolBox. From there, you should probably try to acquire as much IIgs source code as possible (specifically the examples provided by Apple DTS) and dissect how it works.

I have a lapel button that reads "Calm down...it's only ones and zeros". The ToolBox is there to help; don't let it intimidate you.—DJD

dBASE III file translation

Using the *PC Transporter* I would like to translate files from *dBASE III+* to an AppleWorks database file. Can it be done?

Jeffrey S. Bock
Newbury Park, Calif.

We've mentioned ***CrossWorks 2.0*** as a method of moving files between and Apple II and an MS-DOS machine, but since the file conversions are actually done on the MS-DOS machine you could run the program with the ***PC Transporter*** (converting the files in the MS-DOS "partition") and then copy the files over to ProDOS. This is faster than using the modem transfer method, but would also require you to correct the filetype of the converted files after you copy them to the ProDOS side of your schizoid computer's world. There is a handy freeware utility called ***FAZ*** (File Attribute Zapper) that will facilitate assigning the correct types.

The other option would be to convert the ***dBASE III+*** data to a text file format suitable to read into AppleWorks. You will lose some data in the transfer this way that CrossWorks will attempt to convert, such as report formats.—DJD

Pushing the envelope

What effect would replacing my 7 MHz TransWarp GS's present 28 MHz crystal oscillator with a 32 MHz one as shown in Applied Engineering's advertisement have?

Stephen Gant
Manteca, Calif.

We have never looked that closely at the advertisement, but... our suspicion is that it may (or may not) speed up your system slightly, if the chip will operate reliably at the higher speed. It would probably also void your warranty.

There have been instructions circulating on accelerating the ***TransWarp GS*** by replacing the 65816 processor on the card with a faster (higher MHz rating) chip, increasing the speed of the crystal oscillator, and (when necessary) installing faster static RAM to support the speed increase. One of the more detailed documents we have seen is credited to Andrew Hall of Western Design Center, and adequately explains the hazards of the "enhancement". We haven't done the modification, but you should realize going in that it isn't simple.

The 65816 should operate at one-fourth the frequency of the crystal oscillator; that is, for a 7 MHz 65816, a 28 MHz oscillator is the correct value. It might be that the 65816 may operate at a slightly higher speed than its rated value, but we couldn't presume to second-guess Applied Engineering or Western Design Center on this. According to Tony Vece at Zip Technology (Zip is also working on a IIgs accelerator card), the static RAM cache used by the IIgs accelerators should accept accesses at the crystal oscillator's speed (four times the processor speed); the corresponding access rate for a 28 MHz oscillator is once every 35.71 nanoseconds. That means if you speed up the oscillator (and processor) too much, you'll need to unsolder and replace the static RAM chips with faster versions.

Don't put too much faith in advertising photos; we remember the original Zip Chip promotional flyers where the chip was shown plugged into the MMU socket (rather than the processor socket) on a IIe motherboard.—DJD

More orphan help

I sold my II Plus and purchased a used enhanced IIe and now I need a patch which will enable me to use Checkmate's *MultiView 80/160* card with AppleWorks. I wrote Checkmate and they told me they no longer support their card. Also, I recently inherited a Qume LetterPro 20S printer and am in need of a bidirectional forms tractor. Qume no longer supports that printer so I am on my own for parts. If anyone can help me or put me in touch with a source I will be forever grateful.

Richard Washington
P.O. Box 253
Fort Union, Va. 23055

Control Panel shortout

Is there any way to disable access (and later, of course, restore it) to the IIgs CDA menu? For example, a game that selectively enables a TransWarp GS would be thrown off if it was manually set.

John Morrison
Philadelphia, Penn.

It's better to tell the user not to alter the Control Panel settings in your documentation that to lock them out of access; let the user accept responsibility for how the computer is used. The particular feature you don't want them to alter is only one of many options the Control Panel controls.—DJD

Hard disk interleaves

How can I determine the best interleave for my hard disk? Do I have to try all possible combinations? I am using a Seagate ST-157N.

Chang Yuh Kang
Republic of Singapore

You don't have to try every combination, but you will have to try several in order to "frame" the best time. This may vary from drive model

to drive model as access characteristics change. You can, for example, start with the highest possible interleave setting, reduce it by five every time until you find the interleaves at which performance gets worse, and then try all the settings between the two where the "drop" in performance occurs.

***A2-Central**'s Tom Vanderpool has done some timings on a Seagate ST-277N, with and without his **TransWarp GS** active, and these were his results (all timings in seconds):*

Interleave	TWGS On	TWGS Off
1:1	14.81	17.80
5:1	13.29	19.61
7:1	10.69	18.09
9:1	13.71	15.66
11:1	12.51	16.96

These timings were accomplished in the same way (move a large file onto a freshly formatted HD and then load it while timing the load).

Here are Tom's timings for an 80 meg ST296N:

Interleave	TWGS On	TWGS Off
1:1	17.24	18.70
5:1	14.10	20.00
7:1	12.12	18.26
8:1	11.17	18.58
9:1	11.20	17.98
10:1	11.36	17.36
11:1	11.91	16.65

*You should pick an interleave at or slightly above where the timing seems to increase significantly; for example, for the ST296N with the **TransWarp GS** "on" Tom picked the 9:1 interleave. However (as we've said), be aware that your specific configuration and usage determine the best timings; the interleave that facilitates the fastest time for writing a 5 megabyte file from BASIC may not be the one that is optimum for loading a huge AppleWorks database file.*

Chinook's new SCSI utilities will include an option to "scan" access under ProDOS 8 and the best interleave to use. You may still want to verify the result with your own timings, however.—DJD

A2-Central™

[redacted]

Edited by:

Dennis Doms

with help from:

Tom Weishaar **Sally Dwyer** **Dean Esmay**
Joyce Hammond **Jeff Neuer** **Jay Jennings**
Tom Vanderpool **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): $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 four 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 $84 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

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

Printed in the U.S.A.

More on DMA SCSI

As a fellow DMA SCSI card owner, I must reply to Udo Huth's letter ("Don't hang up", June 1990). I have a Jasmine 40 megabyte drive attached to the DMA card in slot 7 and I originally thought the system was hanging if the drive was not switched on. In practice, it "times out" after 20-25 seconds and continues from the 3.5 disk.

I have some comparative times for the old and new cards. My IIgs system has 3.25 megabytes of memory, a *TransWarp GS*, and my Jasmine's seek time is 28 milliseconds. The interleave is 1:1, which is better than 2:1 and the (unknown) original value, but I'm not sure if it's the best! Both times use the new drivers (which are slightly faster for the old card as well!). A fascinating thing is that the times often vary up and down by several seconds under otherwise fixed conditions. The times listed are average timings:

	Rev. C	DMA
Boot to Finder - 550K of DA's, etc.	44.0	36.0
AppleWorks GS - WP, SS, Comm	24.5	17.5
ProSel volume stats		
- linear read	9.5	2.2
- random read	28.5	31.5
- OS overhead	4.5	11.1
Digitized sound files		
- 200+ blocks	3.0	1.0
- 600+ blocks	6.0	1.0
- 670+ blocks	6.5	1.5

The sound files were what convinced me to buy the card. I can play through a directory of sounds in (almost) real time-- very impressive. The simple fact is that the card speeds up the transfer rate. If the drive spends its time seeking, as during a boot or loading a segmented file like *AppleWorks GS*, then the speed-up is minor. Give it a large file to load and you are talking "greased lightening". I am hoping Claris will provide an installation option to allow "true" preloading of *AppleWorks GS* so that people with enough memory and a DMA card will be up and running (all modules) in around 12 seconds!

Peter Watson
Box Hill North, Vic.

More power to you

I have a 512K IIgs and have noticed a slight problem. My system is configured as follows:

Basic 512K IIgs
Apple RGB Monitor
Kensington *System Saver*
GS-RAM Plus with 1 megabyte
Applied Engineering *Datalink 2400*
HyperStudio digitizer
CMS 20 megabyte hard disk
Apple 3.5
Apple UniDisk 3.5
Apple 5.25

I didn't notice this problem until I daisy-chained the UniDisk from the Apple 3.5. However, when I boot the system from the hard disk the read/write light on the UniDisk pulses! I took the UniDisk off of the system and then noticed that the 5.25 read/write light did the same thing but wasn't as noticeable.

I then put the UniDisk back in the chain and noticed that the UniDisk and 5.25 drive were both pulsing at the same time!

I stripped the whole system down to a basic machine with only the Apple 3.5 drive connected and tried all versions of GS/OS (v2.0 through 5.0.2) and found that I had an audible pulse tone through the speaker. After disconnecting the speaker I noticed I could still hear an electrical click in the power supply. I took it to our local Apple "dealer" and he confirmed that the power supply is bad.

Now my question: the Apple supply is $135 as opposed to the Applied Engineering supply which is $99. Which would you consider, forgetting cost difference, as better for the IIgs system?

Walter Monhof
Hephzibah, Ga.

This sounds like a nightmare. Let's retrace a few steps.

First, the pulse/click with the UniDisk 3.5 installed is normal. Some IIgs programs like the Finder have to monitor the drive to see if a disk has been changed; the UniDisk takes this "polling" rather hard. We've got IIgs systems here that have survived this for years.

Second, the click in the speaker may be normal. My ROM 01 machine at home does this (with only an Apple 3.5 and 5.25 attached) when I turn the sound level up. It apparently is some sort of crosstalk reaching the sound circuitry.

Third, the "click" in the power supply may or may not be normal. The supply is a switching type that emits a high-pitched whine in some people's hearing range (I guess I'm one of the "fortunate" ones). As the supply has to "strain" a bit to send the polling pulse to the disk port, I can hear the frequency of the whine change. My system has done this ever since I bought it.

If you have had no other problems with the system other than the "click" from the power supply and speaker while the drives are polled (the pulsing of the drive lights is perfectly normal) you may want to check a little more into how the dealer determined the supply is "bad"; that is, did he use some sort of diagnostic equipment or just recommend replacing the supply as a cure for the "problems" you've seen. If the latter, I'd suggest asking for a guarantee that the new supply will eliminate the problems. If he's willing to do it, paying the extra dollars may be worth it. If not, and you determine the supply is the problem, then it may be a toss-up. My feeling is that it is worth paying a little more for the Apple logo all other things being equal; for one thing, it eliminates any contention with the dealer about my system configuration. Others are quite welcome to disagree.—DJD