A2-Central™

formerly Open-Apple

May 1991
Vol.7, No.4

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

A journal and exchange of Apple II discoveries

Miscellanea

"Apple CD-ROM acknowledges Apple II products; film at 11." Part of the latest Apple User Group mailing from Apple Computer included a CD-ROM with several product training stacks; these are HyperCard stacks dealers and other interested parties can work-through to learn about new Apple products. Like most propaganda Apple inundates users groups and dealers with, these CD-ROMs have historically been Mac-only with a possible occasional reference to the Apple II.

The differences in this CD-ROM are apparent from the second you open the cover; the descriptive blurb mentions the presence of **HyperCard IIgs** stacks on the disk for two products (the Apple IIe Card for the Mac LC and, of course, HyperCard IIgs itself). It also makes special mention that the IIgs stacks are in COLOR (yes, they put the word in all caps).

We hope this means Apple is prepared to quit dragging its feet on making the Apple II more visible in its marketing strategy. It's good to see some of the word is getting to user groups and dealers, but it may be getting buried in the avalanche of Mac-exclusive material sent out. It also doesn't redress Apple's total neglect of the Apple II in its television campaign of last fall; until Apple gets its products in the *public* eye, any chance of instilling user confidence (and loyalty) is minimal.

Is something (big?) brewing at Apple? With spring comes shirt-sleeve weather, baseball, and a rebirth of Apple II Forever rumors. (Autumn, on the other hand, always brings Death of the Apple II rumors. Is this a pagan ritual?)

The main theme of this spring's rumors seems to be along the line of a less expensive IIgs rather than the wildly enhanced IIgs theme of the last couple of years. Tales of Apple's surprise at exceedingly strong sales of the low-priced Mac Classic abound. Apple recently announced yet another reorganization, this time to accelerate the development of new technologies and new products. Out of this reorganization has sprung a Consumer Division, to which the Apple II has been assigned. Speculation is that the long-awaited "business plan" for the Apple II is emerging; one would normally assume that *consumer* products would be *advertised.*

The most widely rumored Apple II enhancement is support of the Apple SuperDrive on the IIe and IIgs via a peripheral card. This would give both systems support for a higher-density (1.44 megabyte) floppy format. It also could be the basis for supporting the access of Mac and MS-DOS files on the IIgs if the appropriate File System Translators are incorporated into a future version of GS/OS.

The SuperDrive would also solve many woes for single floppy systems; you would actually be able to get a full GS/OS system on a (high-density) 3.5 with room to spare for enhancements like extra desk accessories, fonts, and so on. Or you could get GS/OS, *HyperCard IIgs,* and the Home Stack on one disk. This leads to the possibility of a 2 Meg IIgs that could run *HyperCard IIgs* right out of the box, which the current IIgs can't do because of memory and disk space limitations. We're just thinking out loud here, kids; don't quote us as saying any of this is actually going to happen.

However, we do plan to grab a SuperDrive to see if it works as an 800K drive on the current IIgs disk port (we're confident the current IIgs drive port doesn't support the 1.44 density); if it does, it would make sense to buy the SuperDrive rather than the Apple 3.5, since Apple charges the same price for either drive.

Again, we don't know if the speculation has any basis in reality. But a less expensive IIgs with a SuperDrive sold through consumer channels is a very attractive prospect for IIgs users wanting to see the machine given a fighting chance.

Are you interested in techonology to assist the disabled? A national search to identify applications of computer technology (ideas, systems, devices, and programs) to aid the disabled is being conducted under the auspices of John Hopkins University and a grant from the National Science Foundation. The competition will culminate in an awards ceremony at the Smithsonian Institution; the grand prize (out of over 100 awards) is $10,000. For more information on participating write C.A.P.D., P.O. Box 1200, Laurel, Md. 20723. The entry deadline is August 23, 1991.

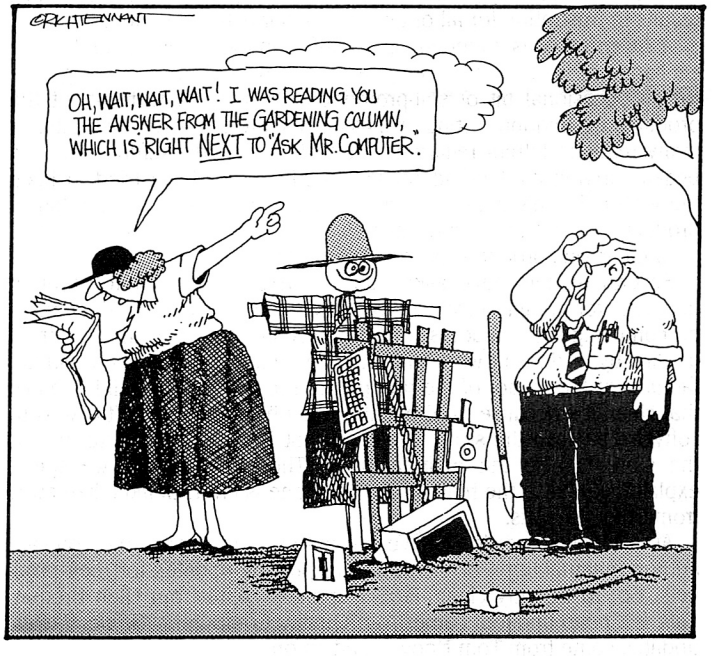
Zip lowers price of IIgs speed. Zip Technology has lowered prices on it's *Zip GS* (formerly the *Zip GSX*) accelerator for the Apple IIgs. The introductory price for the 7 MHz vesion is now $149; upgrades to 8 and 9 MHz are $29.95 and $59.95 respectively.

Cache memory can also be upgraded; 8K is $19.95 and 32K is $49.95.

The fastest *Zip GS* is a 10 MHz with full 64K cache; in mid-April when we checked the complete board current price was $429, or $250 as an upgrade to an earlier *Zip GSX.* Contact Zip Technology, 5601 Slauson Avenue, Suite #190, Culver City, Calif. 90230, 213-337-1313 or 800-937-9737 for orders. Zip's FAX number is 213-337-9337.

A new ROM v2.0 adds capabilities to the *RamFast* SCSI interface. The ROM adds support for removeable media devices such as the Apple CD-ROM and SyQuest cartridge hard disk as well as tape drives.

As with the previous version, the installation software is accessible



through a ROM disk implemented on the *RamFast* itself. The new text-based interface displays a row of selection options in a bar at the top of the screen; you can use the keyboard or mouse to pick and choose. From the menu you can set options for the *RamFast*, format and partition your hard disk, and so on.

A new option is tape backup and restore support; the *RamFast* will work with a "streaming" tape drive (one that reads and writes data in a continuous stream rather than with random access techniques) to backup or restore your drive in a few minutes. This operation can be conducted in the background while you use your system as long as you aren't writing to the volume you are currently using; for example, you can back up or restore one partition of your hard disk while working on another (we haven't had the opportunity to test this ourselves; we don't have a tape unit here, since we prefer to use a SyQuest unit for backups). Non-streaming mechanisms (one from 3M was specifically mentioned) do work but are not as fast. Supported mechanisms include models from 3M, Teac, Weitex, Adaptec, Archive, and Viper.

We checked a SyQuest *SQ555* (42 megabyte cartridge hard disk) with the *RamFast* and an Apple High-Speed SCSI card; the initial (power-up) boot time to the Finder on a 2.8 MHz IIgs was about 24 seconds with the Apple interface compared to a little over 8 seconds with the *RamFast*.

We also attached an AppleCD SC and were able to read the ProDOS partition of Apple's *Gorrillas in the Disc* (Developer CD-ROM Volume6) using the *RamFast* with its driver. Reading other files systems from GS/OS would require adding the appropriate FST, such as the HS.FST for the High-Sierra format. Additional CD-ROMs supported are models from Sony, Toshiba, Hitachi, and NEC.

ROM 2.0 also has a provision for selecting any partition as the boot partition; this is nice if you want to change over the system to boot into a different operating system for a series of sessions. To select the boot partition, you hold down a number key (for the partition number you want to select) as you boot. If your current partition boots into ProDOS 8, you will have to be **quick** to get that key down.

Other added features are the support of more partitions (up to 12 from a previous 8), password protection to "lock" the hard drive, and performance enhancement for drives without track caching. The ROM upgrade is $15 from C. V. Technologies, 1800 East Whipp Road #200, Kettering, Ohio 45440, 513-435-5743.

***GS Numerics* (see "*GS Numerics*: Graphic Math", *A2-Central*, Feb. 1990) has a new lower price of $99.95.** You can order from Spring Branch Software, Inc., R. R. 2, Box 268A, Manchester, Iowa 52057, 319-927-6537, or see this month's catalog.

Update madness. While we're on the subject of product updates...it's nice to see we've instilled enough trust in our users that you feel we can solve any problem. Unfortunately, handling requests to distribute other people's property isn't something we can do. We have to turn you down when you request updates from us, which results in frustration for all of us, so let's clarify the situation.

A2-Central was founded as an informational newsletter for the exchange of ideas and has generally been devoid of advertising save for an occasional bit of self-promotion. We tend to write about the products *we* find interesting or that we ourselves use (we also publish many readers' letters regarding products *you* find useful). Since we began carrying hard-to-find Apple II products, we have tried to keep the editorial content of the newsletter separate from our catalog of products, except to mention ourselves as an additional source when we are carrying products that we also discuss.

Several readers have asked us to make an attempt to mention upgrades for popular products when they are available. This seems to be confusing some readers who began requesting their updates from us directly. It doesn't work that way; *you must go back to the manufacturer or the author of the product for the update.* If we didn't make that clear, it's because we thought it was obvious; you don't take your software back to the store where you got it to get an update (unless the store is also the manufacturer). The person-power we spend explaining this via the mail or on the phone is effort being taken away from other pursuits.

As a reseller, we don't have the authority to bypass the software author, manufacturer, or distributor to provide updates for *their* products. So, to be very clear: *ProSel* updates come from Glen Bredon, *Talk is Cheap* updates come from Quality Computers, *GEnie Master* updates come from Tom Hoover, and so on.

We are in a similar position with Apple updates, such as System Software. We do obtain licenses for distribution of Apple's system software on some of our products. Apple's licenses are very specific in the way we are supposed to handle this distribution. By any interpretation of the license, we aren't allowed to provide "on demand" updates of the raw system software disks via mail.

You should be able to obtain system software updates from one of the following sources: an authorized Apple Dealer (if the dealer says he can't provide the update, give Apple's Customer Assistance Center at 800-776-2333 a call), an online service that has obtained the appropriate electronic licensing agreement from Apple (such as America Online, GEnie, CompuServe, or others), an Apple user group that has obtained the necessary licensing, or as a product from the Apple Programmers and Developers Association (You no longer have to be a member of APDA to buy products there unless the product is still in a beta version, so highlight these APDA numbers: U.S.A.–800-282-2732, Canada–800-637-0029, Elsewhere–408-562-3910). You might also find the system software provided as part of someone's product. In our case, the current version of GS/OS is on our December 1990, January 1991, and March 1991 ***A2-Central*** disks, which we sell (by special request) as a "quarterly set" for $18.

Most of these system software update paths don't include new manuals. We aren't aware of any update to the *user* manuals since the introduction of System Software 5.0 (your Apple dealer should be able to obtain this for you; the Apple part number is A0013LL/A and the suggested retail price is $49). We also don't have access to copies of any Apple user manuals for the IIe, IIc, IIgs, and so on that are provided with the computers; these are only supplied by Apple. If you buy a used Apple II and require the manuals, it would be wise to make sure they are included in your purchase. If you have Apple manuals and feel they are lacking, fill out the Reader Response card Apple encloses with the computer and let them know. Make note of the address for later comments; we've found the problems you notice several months later aren't the same ones you'll notice the first week you have the system.

Some of the tales of woe are heartbreaking; it is irritating to have a product sitting unused because some key component is missing. But unless it's *our* product, we can't do anything except make note of the problem and, if it's one that appears to be of common interest, let others know about it.—DJD

Questioned SANE-ity

"This is a simple game. You throw the ball; you hit the ball; you catch the ball."—Durham Bull's manager in *Bull Durham.*

The problem with being the editor is that every month you have to figure out something to write about. It's usually easy; most of the articles start with a series of letters asking a common question, such as the following example:

> From an Applesoft or 8-bit machine language program, how can I convert the double precision SANE numbers stored in AppleWorks spreadsheet files into some form usable by Applesoft?
>
> John G. Thomas
> Trenton, N.J.

This looks pretty innocent, and reasonable, and it is both of those things. It could be a lead-in for an article about reading spreadsheet files from Applesoft. It turns out that isn't practical for a single issue; Tom Weishaar wrote "Reading AppleWorks databases" back in ***Open-Apple***, Vol. 3, No. 2 (March 1987), but the Apple II File Type Note for AppleWorks database files (type $19) only requires about a page to describe the data record format. Spreadsheet files (type $1B) are much more complicated; about ***5*** pages are dedicated to the data record format.

So it's easy; we just describe the SANE format, right? Maybe even show how to feed the number into an Applesoft variable. It makes a nice, concise letter, right? That's what I thought.

Except that even the "simple" form of the answer takes more space than a letter really should, so it gets answered here as a two-part question. It's been broken into information a mathematically-inclined person can use to interpret AppleWorks constant values (it doesn't presume to teach number systems and so on) and a "practical" section with a program to read and display the values.

The first part is "How do I interpret an AppleWorks spreadsheet numeric value?" "SANE" stands for "Standard Apple Numerics Environment". Apple developed SANE as a mathematics toolkit for demanding applications. SANE consists of a series of function calls for manipulating numbers and conversion utilities for importing and exporting their text representations. SANE is available in the Apple IIgs and Macintosh tools supplied with the system software for the respective computers. It is also available as a set of 8-bit Apple II routines accessible from assembly language or Apple Pascal that can be licensed for commerical use from Apple Software Licensing, Apple Computer, Inc., 20525 Mariani Avenue, M/S 38I, Cupertino, Calif. 95014, 408-974-4667.

Our eventual aim is to convert the powers of 2 representation common to binary computers into a powers of ten representation more commonly used by humans. SANE uses several binary formats for representing its numbers; AppleWorks uses the SANE "double" format as presented in the *Apple Numerics Reference Manual, Second Edition,* page 248, in Figure D-3. This is a 64-bit representation which breaks down like this:

```
                              Byte
   0       1       2       3       4       5       6       7
seeeeeee eeeefffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
```

The bits are assigned as follows: "s" is the sign bit (it's set to "1" if the sign is negative), "e" signifies bits (11 of them) representing the binary exponent, and "f" signifies bits (52 of them) representing a binary floating point value.

When AppleWorks stores the 8 bytes of this value, like most other Apple II numeric values the bytes are stored in *least significant* to *most significant* order. That means the ordering within the AppleWorks file will be reversed to:

```
                              Byte
   7       6       5       4       3       2       1       0
ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff eeeefffff seeeeeee
```

Remember to correct this ordering before trying to evaluate the number.

The use of the sign bit is straightforward, but both the exponent and floating point value are manipulated to generate the final value of the number. Table D-3 (on the same page of the SANE reference) gives the methods of classifying and intepreting these values. In our own words it works out like this (S is the value of the sign bit, E is the value of the exponent, and F is the value of the floating point portion):

• If E is neither zero (all bits clear) nor 2047 (all bits set), then the value is calculated by:

```
(1)^S * 2^(E-1023) * (1.F)
```

This is the "Nomalized" class.

• If E is zero and F is non-zero, then the value is calculated by:

```
(1)^S * 2^(E-1022) * (0.F)
```

This is the "Denormalized" class.

• If both E and F are zero, then the value is calculated by:

```
(1)^S * 0
```

This is the "Zero" class; notice the value is considered "signed".

• If E is 2047 and F is zero, then the value is calculated by:

```
(1)^S * Infinity
```

This is the "Infinity" class; you can have positive or negative Infinity.

• If E is 2047 and F is non-zero, then the value is "not a number", represented by:

```
NaN
```

This is the "NaN" class. You can get a value that is NaN by attempting a calculation on a real number that does not return a real number; for example, the square root of -1 (a real number) does not result in a value within the set of real numbers (there is no real number you can multiply times itself to result in a -1). If you have enough math background to grasp this, then you can get by. Otherwise consider such a result "undefined for our number system", and don't let it keep you up nights.

For both the Normalized and Denormalized classes, notice that the bits representing the floating point portion are interpreted as a binary fraction (that's a *binary* point, not a *decimal* point). For the Normalized class, you add one to the fractional value.

Also notice that the exponent is adjusted by subtracting either 1023 or 1022, depending on the class.

This information and the Apple II File Type Note for type $1B (AppleWorks spreadsheet file) is enough to build a simple routine to read SANE constants from a spreadsheet file. (Apple II File Type Notes are distributed with the Apple II Technical Notes.)

The second part of the question is "How do I convert this to a form usable by Applesoft". The answer is that you don't, completely. Applesoft's floating point math routines are limited to 5-byte values, and you can't stuff all of an 8-byte value into 5 bytes. So you are left with compromises.

Part of the "abbreviation" is the limitation of Applesoft's range to a 8-bit exponent. SANE values that fall within the range from 2^(-128) to 2^127 can be rounded into an approximate Applesoft value (this corresponds roughly to a range of 10^-39 to 10^38).

You can also identify conditions such as positive and negative infinity, zero, and NaN by testing the value of the exponent and the floating point portion before attempting to use them in a conversion.

The following program is bare bones; all it does is scan a spreadsheet file row by row, column by column, looking for and printing constant values. (Fancier implementations are left as an exercise for the reader.)

First, we set up a few constants:

```
1000  REM == brute force decode of SS ==
1010  LOMEM: 24576: REM lock out memory below $6000
1020  DIM B(7): REM array for converting byte to bits
1030 B = 300: REM initial byte offset (past ASP header)
1040 D$ =  CHR$ (4): REM control-D
1050 FI$ = "TEST.SS": REM name of spreadsheet file
```

We use the value of "B" as a pointer to the next byte of the file we want to read. We start with B set to 300, the first byte past the information header for the spreadsheet file and the start of the first row of spreadsheet information.

The AppleWorks spreadsheet file is stored as sequential "rows" corresponding to the rows in the loaded spreadsheet. The definition for a row begins with two bytes representing the length of the row. We BLOAD the two bytes and PEEK them out of memory to find out how long the row is; a row length of $FFFF (65535 decimal) indicates that we have read all the valid rows:

```
1070  REM — read row length —
1080  PRINT D$;"BLOAD ";FI$;",T$1B,A$2000,B";B;",L2": REM load length of row
1090 RL =  PEEK (8192) +  PEEK (8193) * 256: REM and evaluate
1100  PRINT "Row length ";RL
1110  PRINT "==========": REM divider
1120  IF (RL = 2 ^ 16 - 1) THEN  GOTO 1490: REM $FFFF signals end
```

Now that we have the row length, we can move the pointer forward two bytes to the start of data for the row and BLOAD the entire row into memory:

```
1130 B = B + 2: REM point to start of row info
1140  PRINT D$;"BLOAD ";FI$;",T$1B,A$2000,B";B;",L";RL: REM read row
```

The first two bytes of the row data define the row number:

```
1160  REM — start new row —
1170 MP = 8192: REM memory pointer
1180 RN =  PEEK (MP) +  PEEK (MP + 1) * 256: REM row number
```

Rows consist of zero or more columns. The first column begins past the row number, and now we can move to decoding each column.

Each column begins with an identification byte. If the byte value is 128 or less, its magnitude is the column width. A value of $FF (255)

indicates there are no more columns in this row. A value of $81-$FE (129-254) indicates that a number of columns equal to that value minus $80 is to be "skipped":

```
1190 MP = MP + 2: REM point at column data
1200 CN = 1: REM column number
1210 CI =  PEEK (MP): REM column info
1220  IF CI = 255 THEN  PRINT "<End of row>": PRINT : GOTO 1450
1230 CS = 0: IF (CI > 128) THEN  GOTO 1390: REM skip columns
1240  IF CI < 128 THEN CS = CI: REM column size
```

The next byte is a flag byte indicating the type of column data. For a constant value, bits 7 and 5 of the byte are both "1":

```
1250 BYTE =  PEEK (MP + 1): GOSUB 1520: IF  NOT (B(7) AND B(5)) THEN
      GOTO 390: REM not constant
```

If we've found a constant, the next byte is another flag byte (which we ignore here).

The following 8 bytes are the SANE "double" value, which we need to read in reverse order. The following routine reads them, displays the bit pattern for each byte, and also builds a 64-character string representing the SANE value. That string is passed to a subroutine beginning in line 1580 to evaluate and display the number:

```
1270  REM — print 64-bit entry for constant —
1280  PRINT "Constant found in Row ";RN;", column ";CN
1290 B$ = "": REM use this string for 64-bit pattern
1300  FOR I = 7 TO 0 STEP  - 1
1310 BYTE =  PEEK (MP + 3 + I): REM get data byte
1320  GOSUB 1520: REM parse bits
1330  FOR II = 7 TO 0 STEP  - 1:B$ = B$ +  STR$ (B(II)): PRINT B(II);:
      NEXT II: PRINT " ";: REM print byte as bits
1340  NEXT I
1350  PRINT : REM end line
1360  GOSUB 1580
1370  PRINT "————————————————": REM divider
```

Now we move to the next column and point to the next data byte, always looping back to the "start of column" check at line 1210:

```
1390  REM — end of column data —
1400  IF CS <  > 0 THEN CN = CN + 1:MP = MP + CS: REM bump column number
1410  IF CS = 0 THEN CN = CN + (CI - 128): REM  or skip "blank" columns
1420 MP = MP + 1: REM bump pointer
1430  GOTO 1210
```

When we hit the end of a row, we jump to here and point at the next row in our disk file:

```
1450  REM — end of row —
1460 B = B + RL: REM point to next row
1470  GOTO 1070
```

When we've finished the last row, we come here:

```
1490  REM — end of data —
1500  END
```

Now come the subroutines. First is an old standby to break a byte into an array of bits:

```
1520  REM — parse byte into bits —
1530  FOR BIT = 7 TO 0 STEP  - 1
1540 B(BIT) = 0: IF ((BYTE - 2 ^ BIT) >  = 0) THEN B(BIT) = 1:BYTE = BYTE - 2 ^ BIT
1550  NEXT BIT
1560  RETURN
```

Next is the routine to take our 64-bit SANE number and represent it as a BASIC floating point number, or report why we can't. First we take the string and split it into three strings representing the sign, exponent, and floating point values:

```
1580  REM — break string into components —
1590  IF  LEN (B$) <  > 64 THEN  STOP : REM something's wrong!
1600 S$ =  LEFT$ (B$,1)
1610 E$ =  MID$ (B$,2,11)
1620 F$ =  RIGHT$ (B$,52)
```

Now we can evaluate the sign bit with a simple test:

```
1630 S = 1: IF S$ = "1" THEN S =  - 1: REM sign
```

For the exponent and floating-point values, we feed them to a subroutine that returns a value. The value will be exact for the exponent, but usually represents a rounded value for the floating point portion:

```
1640 BB$ = E$: GOSUB 1700:E = V: REM exponent
1650 BB$ = F$: GOSUB 1700:F = V: REM floating point representation
```

Now we can print the values, and feed them to an evaluation routine to display the Applesoft floating point interpretation:

Ask (or tell) Uncle DOS

*Last month I included information on GEM 4.0 updates; although I mentioned the update price of $10, I forgot that updates that require a disk cost an additional $5 (total of $15). How would you get an update without a disk? Well, that's for people who download the **GEM** update (the $10 gets you a program to unlock all of it's features).*

GEM author Tom Hoover also let us know that he's already shipping version 4.1, which will work correctly with new library pages being implemented by GEnie (the older versions will break). Registered owners of 4.0 can download this update free from GEnie.

The simple Backup II patch we gave last month (page 7.20) to allow use with the Universal Drive Controller was a little too simple; it should actually consist of four NOP instructions instead of the one that was given. Let me go back to Steve Cunningham's original patch:

```
POKE 32768,234:POKE 32769,234
POKE 32770,234:POKE 32771,234
BSAVE BACKUPII,A$8000,L4,B$28C4,TSYS
```

We are slowly but steadily catching up with the backlog of mail (this is a pleasant surprise). As we get closer to the end of the long march, many of the items have turned out to be programming questions that have been on the "back burner", so you may see the number of programming examples increase for a couple of months. If you're not a programmer, don't panic; things tend to balance out over the long haul.—DJD

Dissappearing drives

One technical point I would like cleared up concerns Apple's new High Speed SCSI Card. When I installed it into my IIgs, the system would no longer boot or recognize any SCSI device.

Dr. Leigh Rowan-Kelly
Pymble, N.S.W.

*Whenever you use the Apple High-Speed SCSI Interface or the **RamFast**, you need to be confident your system's memory is DMA (direct memory access) compatible. Each card has a switch to disable its use of DMA, and if you have any doubts try the card with the use of DMA disabled. If it begins to work, you probably need to contact the manufacturer of your memory card (this includes any adapter used in the IIgs memory card slot, such as the Applied Engineering **RamKeeper**) to review your options. If you are using an Apple II system other than a IIgs with an accelerator, remember that accelerators for 8-bit systems are not currently DMA compatible. (Drew Vogan at C.V.Tech also let us know that you need version 1.02 of the **Zip GSX** for DMA compatibility with the **RamFast**.)*

*If you have established whether your system is DMA compatible and have the card correctly configured but are still having trouble with the drive, the problem could be in the cable (not an uncommon problem), or it could be the more obscure issue of **termination**.*

*The specified collection of SCSI signals running between different devices is referred to as a **bus**. You can connect up to eight devices on*

```
1660 PRINT "S = ";S;", E = ";E;", F = ";F
1670 GOSUB 1750
1680 RETURN
```

This is the "binary to decimal" converter; it works by scanning the binary string character by character. For each character scanned, it doubles the previous accumulated value in "V" and adds the value of the current bit (0 or 1). "V" contains the decimal result when we RETURN:

```
1700 REM — convert binary to decimal (~exact) —
1710 BL = LEN (BB$):V = 0: IF NOT ( LEN (BB$)) THEN GOTO 1730
1720 FOR BB = 1 TO BL:V = V * 2 + ( MID$ (BB$,BB,1) = "1"): NEXT BB
1730 RETURN
```

(Math hawks may wonder why we didn't use logarithms to calculate the value. The problem is that the limited precision of Applesoft introduced more error than we felt was acceptable.)

Here's the subroutine to determine the SANE class of the result. "Normalized" and "Denormalized" classes are passed to subroutines to determine if they will overflow or underflow as Applesoft floating point numbers. The "Zero", "Infinity", and "NaN" classes are identified by the values of the exponent and the floating point components:

```
1750 REM — display A/S value (if possible) —
1760 REM 2^52 is about 2.220446059E-16 per HP-16c; 2.22044605E-16 per A/S
1770 IF ((E) AND (E < 2047)) THEN GOSUB 1830: REM normalized
1780 IF (( NOT E) AND ( NOT F)) THEN GOSUB 1890: PRINT "Value = ";
       S * (F / (2 ^ - 52)) * (2 ^ (E - 1022)): REM denormalized
1790 IF (( NOT E) AND ( NOT F)) THEN PRINT "Value = Zero (sign = ";S;")"
1800 IF ((E = 2047) AND ( NOT F)) THEN PRINT "Value = Infinity (sign = ";S;")"
1810 IF ((E = 2047) AND F) THEN PRINT "Value = NaN (sign is immaterial)"
1820 RETURN
```

For the Normalized class, first we need to check and see if the value is too large or too small for Applesoft to accurately reflect. We do this by checking the adjusted size of the exponent. The basic equation used is:

```
value = sign * 2^(exponent-1023) * 1.float
```

where "float" is our binary floating point string interpreted as part of a binary decimal. We didn't try to place the "binary point" when evaluating the 52-bit binary floating point component of the SANE value, so our value for "F" is 2^52 times too large. To evaluate "1.float", we multiply "F" by 2^52, add 2^52 (that gives us the equavalent of "1.float * 2^52"), then divide by 2^52.

This subroutine evaluates a normalized value.

```
1830 REM — do normalized checks —
1840 IF ((E - 1023) > = 127) THEN PRINT "Value overflow (normalized)":
       GOTO 1870
1850 IF ((E - 1023) < = - 128) THEN PRINT "Value underflow (normalized)":
       GOTO 1870
1860 PRINT "Value = ";S * (((2 ^ 52) + F) / (2 ^ 52)) * (2 ^ (E - 1023))
1870 RETURN
```

For the denormalized class, we also do the Applesoft range checking with the exponent. The evaluation is slightly different than the normalized version:

```
value = sign * 2^(exponent-1022) * 0.float
```

This subroutine does the work for the denormalized class:

```
1890 REM — do denormalized value —
1900 IF ((E - 1022) > = 127) THEN PRINT "Value overflow (denormalized)":
       GOTO 1930
1910 IF ((E - 1022) < = - 128) THEN PRINT "Value underflow (denormalized)":
       GOTO 1930
1920 PRINT "Value = ";S * (F / (2 ^ - 52)) * (2 ^ (E - 1022))
1930 RETURN
```

On the way to the Applesoft representation, we obtained the basic values for the elements of the SANE "double" format. If you just want to be able to *print* the representation of any valid SANE value as part of a "dump" of the spreadsheet file, you may be able to write a routine to do that from this information. Personally, we're not that enthused about the prospect of writing it, though if someone puzzles it out we'd be interested in seeing your approach. If you want to perform calculations with the actual values, probably the best solution is to license the 8-bit version of SANE. AppleWorks itself contains SANE as part of its code for just this reason (so *that's* how those values got there!).

If you have a IIgs, a SANE toolset for native-mode (16-bit) applications to use is part of the System Software. In fact, I used ORCA/C to help discern the number formats since it can use SANE types directly (I wasn't up to writing a complete set of Applesoft routines to manipulate SANE numbers).—DJD

---

the bus, each with a separate identification (SCSI ID). Each device connected to the bus must have a unique SCSI ID, which is usually selected by switch settings or by a set of "jumpers" (paired electronic terminals which can be connected by small "jumper blocks"). The interface card itself is a device, so it occupies one of the bus positions; the card's SCSI ID on the Apple cards is set using a bank of small switches.

Most pre-assembled hard disks allow you to set the SCSI ID by means of switches on the exterior of the drive. If you are assembling a drive yourself, the drive mechanism usually has a set of jumpers that allow you to set the SCSI ID; you can user jumper blocks to connect these and set the ID, or (if so inclined) you can connect the jumpers to a switch mechanism. The location and use of these jumpers varies from drive to drive, so if your drive mechanism did not include instructions on connecting them you will need to contact either the supplier or the manufacturer of the mechanism to determine how to configure them.

There are two other jumpers you should try to locate. One determines whether parity (a method of checking for transfer errors) is enabled; if you have problems with a drive, you may want to try this jumper in the other position. (We don't have a hard and fast rule for determining whether it is "normally" connected or not.) The other determines whether the drive supplies terminator power, and that could be the culprit in your case.

Signal pulses sent out on the SCSI bus can be "reflected" when they hit the end of the bus. Imagine a long rope tied firmly to a pole; if you give the rope a quick "flip" and send a pulse down the rope, the pulse may bounce back toward you (slightly diminished) when it reaches the anchor point. The ends of electronic bus connections can do something similar with signal pulses.

You can kill the reflection by "damping" (absorbing) the signal energy at the point where it would normally be reflected. For example, if you connected the rope to the wall using a large enough spring, the pulse would go into stretching the spring and (for the most part) be absorbed. Electronic pulses on a bus can be absorbed by adding resistance to a portion of the circuit; the energy of the pulse is dissipated as it hits the resistance. Since this resistance ideally absorbs the energy at the "ends" of the bus where a signal might be reflected, it's referred to as "termination".

The SCSI specification requires termination be present at each end of the chain of devices. Many Apple II users use just a SCSI interface and a hard disk; this requires that termination exist on the SCSI interface and on the hard disk. The SCSI termination resistors require power be supplied for their purpose.

Now here's the problem: The Apple Rev. C. SCSI interface circuitry did not provide either termination or terminator power. Still, the SCSI bus usually worked as long as the drive was terminated and the cable run did not extend over too long a distance. That's partially why suppliers like Chinook encouraged the use of shorter SCSI cables; also, longer cable runs cause signal degradation in any circumstance (the SCSI specification allows for a maximum cable length of 6 meters for the total bus length). You would ideally want the drive to be terminated and to supply terminator power to the bus, but even if it didn't do these things it might work on the Rev. C card.

When Apple designed the High Speed SCSI Card, they wanted it to be fully compatible with the SCSI specification, so termination was added to the card. However, allowing the card to supply terminator power to the bus would have caused it to exceed the allowed power draw it could take from its slot in the Apple II. So the High Speed Card does not supply the

power needed to stabilize its terminator resistors; it expects a device attached to the bus to supply terminator power (only one device on the bus needs to do this, although there should be no problem if more than one device supplies power).

A SCSI device you add may or may not supply terminator power. Many Mac models in use do supply power, so if you buy a Mac drive it may not be configured to supply terminator power. You may find it doesn't work on the High Speed SCSI Card unless the drive can be altered to supply terminator power. (You should be able to get the information from the supplier.)

Drive termination is also an issue. Chris Adams at Chinook Technology says that, to his knowledge, all drives are shipped with internal termination resistors installed and the company integrating the drives for sale must make the decision about using internal or external termination. Chinook is now supplying its drives configured with external terminators to reduce confusion when daisy-chaining drives. With this method, if you don't ***see*** *the terminator attached to the drive, you know it isn't terminated. You also don't have to worry about multiple drives being (internally) terminated.*

The ***RamFast*** *card does have a switch that causes the interface to supply terminator power. If you have a drive that isn't appearing with the* ***RamFast****, try flipping the switch to supply power. This may also cause the* ***RamFast*** *to exceed the recommended power draw for it's slot, but we haven't heard of such a problem related to its use. (If you do use this option and have problems, we'd like to hear from you.)*

The issue of power draw from a particular slot is not as much at issue with the reliable operation of the computer as the total power draw of all devices. We surmise Apple's position to be that each card must draw no more than the recommended power because the type and number of cards in a particular system cannot be known in advance, and establishing a per-slot limit insures that no combination can exceed the capacity of the power supply.—DJD

Control-Show'ed up

How quickly we forget old friends... for the display of control characters, the *Global Program Line Editor* (*GPLE*) from Beagle Bros requires only the "ESCape H" command to do this. Normal function is restored by "&" or "CALL 1016". This works in both DOS 3.3 and ProDOS. This, of course, applies to program material, not to routine screen output.

For viewing control characters in text files, the communications program *Modem MGR* permits entering or viewing control characters in its editor mode. A "Page" command, for instance, shows up as "^L". *GPLE* shows all carriage returns as inverse "M"'s; *Modem MGR* does not show carriage returns.

Charles A. Jarvis
Leavenworth, Ks.

You're totally right; we tried ***GPLE****'s "control-show" mode and embedded control characters show up during a catalog.*

We primarily use ***Program Writer*** *for Applesoft programming these days, but GPLE provided a lot of resident features for Applesoft that* ***Program Writer*** *(which is called up only when actually editing Applesoft programs) doesn't. With GPLE installed, you can use its commands and macros even while running a BASIC program. GPLE primarily gave way for us since it didn't have a full-screen mode and sometimes (though only very occasionally) collided with other programs.*

There are other utilities that will allow viewing control characters in files. ***AppleWriter*** *did this; though locating* ***AppleWriter*** *(which has been out of print for years) is geting more difficult,* ***AppleWriter*** *author Paul Lutus's freeware* ***FreeWriter*** *program also has the capability.—DJD*

AppleWorks conversions

Does anyone know of a way to convert AppleWorks or DIF files into *Managing Your Money* format? They do not have an import function. I would like to use that program but I have lots of files which I'm not willing to re-enter.

Karl Hess
Cleveland, Ohio

Two pages or not two pages

Could you please write an article on how to use text page 2? For writing on page 2, while displaying page 1, and vice versa (in Applesoft).

Donald Burritt
Riverside, Calif.

Sorry, but the Apple II video firmware only supports using 40-column text page 1. Applesoft commands such as PRINT and the underlying monitor routines are only designed to drive the primary text page.

If you know ahead of time what you want to appear on Page 2 (such as a page containing a "help" screen) you can design "static" screens on text page 1, then BSAVE the screens to a file and BLOAD them later into page 2. The following program will illustrate:

```
1000  REM == make/show page 2 screens ==
1010  IF  PEEK (104) < 12 THEN  POKE 103,1:
      POKE 104,12: POKE 12 * 256,0: PRINT  CHR$ (4);
      "RUN P2.SCREENS": REM relocate to $C01 if in
      text page 2 memory
1020 P1 = 49236:P2 = 49237: REM page 1/2 select
1030  VTAB 12
1040  REM — make pages —
1050  FOR I = 1 TO 4
1060  HOME : VTAB 12: HTAB 14
1070  REM the following can print anything you want
1080  PRINT "Sample page ";I
1090  PRINT  CHR$ (4);"BSAVE PAGE,A$400,L$400,B";
      I * 1024
1100  NEXT I
1110  REM — display page on cue —
1120  HOME
1130  VTAB 12
1140  PRINT "Choose 1-4 or '0' to quit: ";
1150  GET A$:A =  VAL (A$)
1160  IF A > 4 THEN 1150
1170  IF  NOT (A) THEN 1240
1180  PRINT
1190  PRINT  CHR$ (4);"BLOAD PAGE,A$800,L$400,B";
      A * 1024
1200  POKE P2,0: REM display page 2
1210  POKE  - 16368,0: WAIT  - 16384,128:
      POKE  - 16368,0
1220  POKE P1,0: REM display page 1
1230  GOTO 1120
1240  HOME
1250  NEW : REM restore pointers to $801
1260  END
```

This program relocates itself above the memory used by the second text page, creates a series of four screens on text page 1, and saves each page to the same file (using the BASIC.System "B" parameter to offset the location of each page within the file by its number; we left page "0" undefined). A specific page can be BLOADed back into text page 2 for display; we select the visible text page via softswitches in the variables "P1" and "P2".

You should ***not*** *BLOAD screens directly back into page 1 memory. Within the $400-$7FF range of memory assigned usually to the text screen there are a set of 64 reserved memory bytes used to store certain system information. These are usually referred to as "screenholes" since they are non-displayed locations interspersed with the display memory. When you do a simple BSAVE SCREEN,A$400,L$400 these screenholes are saved with the screen memory; if you re-load the screen you will overwrite any* ***new*** *information in these 64 bytes with the* ***old*** *information; the results are unpredictable. ProDOS marks the text screen memory as "in use" for this reason, and you'll see a "NO BUFFERS AVAILABLE" error (indicating the memory is not available for ProDOS's use to load a file) from BASIC.System if you try to load a screen there.*

Page 2 text also reveals one of the few compatibility quirks of the IIgs. Due to the way the IIgs video screens are implemented and a design decision the display of text page 2 was not directly supported by default. It turned out that the use of text page 2 was important enough that the IIgs engineers later added a way to enable it. If you type control-open-apple-escape on the IIgs, you'll see the option for Alternate Display Mode. To see page 2, you'll need to enable the alternate display mode. If you plan to use page 2 in your program, you should add a note for IIgs users mentioning the need to do this.—DJD

Hyper laser

Apple II forever department: My school recently bought a laser disc player and a Mac to control it, and a chemistry disc which didn't have any software with it. I will be working on writing something for the Mac once the laser disc toolkit for the Mac comes from APDA, but in the meantime I demoed to the other teachers how *HyperStudio* can run the laser disc just fine without all the extra fuss. Meanwhile, on my own IIgs, on one *ProSel* screen I have both the first long Applesoft program I wrote 10 years ago for my II+ (it averages my grades), and the most up-to-date medium in *HyperStudio*, both accessible with a touch of a button on the same machine. The IIgs is a really great bridge machine as the computer evolves. Thank you for your continuing support of the Apple II.

Martin F. Schmidt
Finksburg, Md.

Manual saves user (again)

I discovered a neat ImageWriter II (with *TimeOut SuperFonts*) problem and solution.

I have had *SuperFonts* for a while and was not really happy with its hang-ups, inability to print one-inch labels, and so on. I figured it was just a quirk of the program since 1" labels printing was not addressed in the book.

Suddenly, after inserting a font I had never

tried before, the whole printing process "hung up" somewhere in the first two or three lines of printing and there was no escape. Only rebooting AppleWorks would work.

All printing in AppleWorks without *SuperFonts* selected worked fine and other programs printed out correctly as well.

Until...I picked up my *ImageWriter Owner's Guide* just for grins and scanned through it. I cam across the DIP switch settings for S-1 and S-2 (on page 108) and, curious as I was, I compared them to the settings on my printer.

All was super...until I looked at S-2, switches 5 and 6. They were the opposite of the illustration. Switch 5 was down and switch 6 was up.

I never really looked at these closely because the book says "Do not touch switches SW2-5 and SW2-6. These are reserved for authorized Apple technicians to adjust hammer-fire timing."

I grabbed my trusty ball pen, retracted the point and used the pen barrel tip to "flip" the switches around to match the book.

I loaded AppleWorks, loaded a file, selected *SuperFonts* and printed. I have printed hundreds of documents and labels since and never once has it failed to print fantastically. The ImageWriter and *SuperFonts* combination is superb. On some fonts, I have to look close to believe that this is "just" a 9-pin printer, and can now "quality out-print" my Panasonic 1091 printer (only by using *SuperFonts*, though).

John J. McMahon
FPO New York, N.Y.

*Manuals are sometimes hard to read, but often they **do help**. The effort is well spent.—DJD*

Apple Pascal User Groups?

Do you know of any Apple Pascal user groups?

Rich Wilson
Lincoln Cons. School District No. IV
Sixth and Elm
Winfield, Mo. 63389

Not that we know of, though some user groups may have Apple Pascal Special Interest Groups (SIGs). We've included your address so interested groups can contact you.

If you're curious about the dialect of Pascal used in Apple Pascal, you might broaden your search to SIGs interested in UCSD (University of California-San Diego) Pascal, which Apple Pascal is based on.—DJD

Learn to 'share

I am using AppleWorks 3.0 (network version) on AppleShare. Can you tell me if there is a way to keep the students from using the open-apple-< to back out of the directories to the root directory? When students choose "get files" for the first time there are two directories already there, "SP" and "SETUP". I'd like these not to show. They confuse the kids.

John M. Perry
St. Petersburg, Fla.

*You can assign the student a subdirectory that they will be **forced** to use by manipulating the access privileges on the AppleShare folders, including the volume itself, so that they can only "Modify files" for their own folder. You can also elect not to allow the user to see (all) folders or files within the folder they are currently viewing.*

There is no way that I know of to prevent a user from seeing selected files or folders. Therefore you can't hide folders in the root directory, because all files involved in the path to their "user" folder must be visible to them (disabling "See folders" in the root directory of the server volume will make all subdirectories on the volume inaccessible to the user!). The students are going to have to learn enough about the computer to conquer their confusion; there's a limit to how much hand-holding can be expected.

You could disable access to the two subdirectories you mention by disabling "See files", "See folders", and "Make changes" options for each of the two folders. The best the user could do would be to open the folder, but they would be unable to see any files it contains or make any changes (including saving their file to the folder). They would be forced to back out and try another folder.

This might encourage them to save files to the root directory; you can stop this by allowing them to "See files" and "See folders" for the root directory, but disabling "Make Changes". This will force them to move into a subdirectory.

We have a problem here with users saving everything to the root directory on the server, which really slows down performance in some cases (Try opening a Finder window on the server with a couple of hundred files in the root directory!), and I've jokingly threatened to do this on the server at the office.

Since the Administrator probably is the owner of the server volume, you'll need to log on as the Administrator to change the server volume's access. (You can either do this from the server or from a workstation; to access the server privileges from a workstation, highlight the server volume itself from Finder and open the information window for it.)—DJD

GS/OS awe

I like the Apple II and the Mac, but my investment in Apple II hardware and software is so great that it doesn't make sense to change just to have what Apple wants me to have. Even with the new "low cost" Macs.

I'll change systems when the Apple IIgs no longer does what I need for it to do. Just using GS/OS gives me the most "awed" feeling because of the way it works. I can't really describe it, but it does feel real good!

Robert Green
Cedar Hill, Texas

Orphans abroad (and here)

Do you know when we will have the new Apple IIgs?

Gilles Mangin
Terville, France

Do you know how to get the IIe enhancement kit and the IIc ROM upgrade (for the UniDisk 3.5)?

Marcus Staender
Unna, Germany

I uploaded my UniDisk RAM-based driver for the old original Apple IIc to GEnie; it is file #15122 ("OLD.IIC.UNI.BXY") and is 14K bytes in size.

The driver should work with any Smartport device but I've only tried it with the UniDisk 3.5. My supplied utilities all expect the UniDisk so beware of the format and eject commands!

I don't reserve RAM in the auxiliary memory; I use the main language card $D0 bank 2 space (see "New MLI command", ***Open-Apple*** Oct. 1988, p. 4.64).

Why this instead of an upgrade? If you think Apple support is bad in the USA, well it's much worse in Canada. Our local Mac dealer offered to fix me up with an Apple and a 3.5 drive for over $800 when I went in asking about UniDisks. I didn't listen to the details at that price. I finally found a used UniDisk for about $200 Canadian. At that price, I took a chance on writing a RAM-based driver.

Doug Mitton
Brockville, Ont.

*Assuming compatibility problems don't arise, that driver is likely to be a popular item. Be aware that the area you're stomping on is reserved for AppleTalk (see "New MLI command verboten", **Open-Apple,** Nov. 1988, p. 4.80); this shouldn't be an issue on a IIc since the AppleTalk code won't be loaded (all you have to worry about is someone else using the same area). Also, since the driver is RAM based it has to be loaded before it is used; that means you'll still have to start up from a 5.25.*

We continue to be overwhelmed by requests from Apple II users that have been orphaned overseas. We see many requests from Apple IIc owners wanting UniDisk 3.5 upgrades and Apple IIgs owners wanting ROM 01 and VGC upgrades for their ROM 00 systems, or the newer ROM 03 IIgs.

*We have pressed Apple many times trying to find out how users can do this when their local Apple dealer denies support of Apple II products, and have gotten **no** definitive answer (at least USA customers can now call the Customer Assistance Center!). Which means you may have to locate and replace the part (or even motherboard) yourself, at your own expense and risk. I think international customers of Apple Computer are learning (and teaching us in the US) a valuable lesson about the company's real committment to user support; indeed, we hear similar complaints from rural customers in the U.S. who can't seem to find an Apple dealer to solve their problems within easy reach.*

*If you need a replacement part or upgrade, possibly the only recourse is to haunt the "for sale" sections of various publications until you locate what you need. One of the largest sources of mail order items is **Computer Shopper** (published by Coastal Associates Publishing L.P., One Park Avanue, New York, N.Y. 10016); it lists some companies such as:*

ElectroValue Industrial, Inc.
P.O. Box 376-CBS
Morris Plains, N.J. 07950
201-267-1117

Pre-Owned Electronics, Inc.
30 Clematis Avenue
Waltham, MA 02154
617-891-6851
800-274-5343
617-891-3556 FAX

Shreve Systems
2421 Malcolm Street
Shreveport, La. 71108
318-635-1121

These companies often buy and sell used Apple II systems and components; some even offer repair exchanges. These companies also may carry rare software and hardware products, and if you are looking for something

obscure they are worth checking. We've also listed other companies in the past as sources of Apple products such as:

Sun Remarketing
P.O. Box 4059
Logan, Utah 84321
801-752-7631
800-821-3221

We are aware that there are situations where US companies won't sell or supply upgrades to non-US markets; we ran into this head on when we offered the Claris AppleWorks 3.0 upgrade late in 1989. As best we understand it, in addition to normal import and export restrictions, the problem is that companies that negotiate for overseas distribution may sign contractual agreements with their authorized agents that prevent them from selling directly into their distributor's market (and undermining the value of the agreement). In some cases, companies may also not enter a market simply because they don't feel they are in a position to deal with support issues in that area of the world.

Since Apple already has dealers in markets that generate many of our reader's complaints, we're not sure that they can hide behind these excuses. Either their agents should support the systems, or Apple should offer support directly since they obviously can't "interfere" with inept support. But other than expressing displeasure on the behalf of our readers, we don't have control over these decisions.—DJD

A2-Central™

[redacted]

Publisher:
Tom Weishaar

Editor:
Dennis Doms

with help from:

Sally Dwyer	**Dean Esmay**	**Joyce Hammond**
Betty Harding	**Jay Jennings**	**Jeff Neuer**
Denise Shaffer	**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

Printed in the U.S.A.

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

GS/OS versions

How can you find out what version of GS/OS you have, other than looking at the disk label? Pressing the space bar during the boot process shows version 3.03 on a System 5.0.4 disk.

Thomas Kepka
Wheaton, Ill.

*After the advent of both versions 5.0.3 and 5.0.4, one of the more frequest questions we received was "how do I tell which version is on my hard disk?". To which the totally honest answer has to be "if you don't know, you should re-install". **And you need to keep track of the last System Software update you performed** for the reasons that follow.*

We can think of a few legitimate reasons you may need to discern the System Software vintage. Unfortunately, it isn't very easy to divine the version of the System Software from that of its component files. We've tried to emphasize that "System Software" indicates a collection of individual files all with individual version numbers. The "3.03" you see on the diagnostic boot screen is the version of the GS/OS kernel for System 5.0.4 and System 5.0.3, so by itself it doesn't differentiate between the two versions. As the System Software stabilizes, changes between files in minor revisions may become fewer.

*Determining the System Software version from the individual files assumes that you **always** install the complete new version, and that the files that can be used to differentiate the versions are part of the set you installed. If you manipulate the files independently, then all bets are off (that's one reason we emphasize use of the Installer, to prevent mixing file versions). If you alter the files so that the creation or modification dates change, then you've eliminated that method of detection.*

All that having been said, these are suggestions for trying to determine the System Software versions for 5.0, 5.0.2, 5.0.3, and 5.0.4. Start off by hitting the spacebar or "Esc" key during a boot, and checking the GS/OS version number that appears on the text screen:

SS Version	5.0	5.0.2	5.0.3	5.0.4
GS/OS version	3.00	3.01	3.03	3.03

*So now you see the problem; v5.0.3 and 5.0.4 have the **same GS/OS version number**. As a matter of fact, only 6 files changed between 5.0.3 and 5.0.4. One of these was TOOL018 (QuickDraw Auxiliary) which should be installed on your system. If you have a utility that will display tool set versions, check and see if you show version 3.0.**3**, indicating System 5.0.**4**. Or, for lack of a better way, check the file dates; for v.5.0.3, the creation (and modification) date and time is Mon, Aug 20, 1990, 10:24 AM; for v.5.0.4, this became Tue, Dec 4, 1990, 1:54 PM.*

*Some users ask for us to print a complete list of revisions for each IIgs System Software revisions. Given that the release notes are often several times the bulk of our newsletter, this isn't going to happen. We will print an **overview** of new features, but if you want to know the minute details you'll have to get your own copy of the release notes (they're supplied as part of the **Apple IIgs System Software Update** package from APDA, $30, APDA part #A2Z1002/C; phone: U.S.A.–800-282-2732, Canada–800-637-0029, Elsewhere–408-562-3910). Otherwise, we'd become solely a "IIgs update list" newsletter.*

Just as you have to put gas and oil in your car, there is a certain minimal amount of work that has to be done to maintain a computer. Asking someone miles away to divine your system configuration is not really rational, and as you can see from above there's no way we can be totally sure without spending more time diagnosing the problem than you would spend doing another installation. Its like calling up a service station and asking what brand and weight of oil you put in your car last year.—DJD

PEEKing at versions

Is there a location in ProDOS and the BASIC.System interpreter that a program can PEEK to identify the version that has been installed? If so, what are the values for the various versions?

Jimmy Isaac
Compton, Calif.

*Per the **ProDOS 8 Technical Reference Manual** (page 97): the version numbers are placed in specified locations in ProDOS 8's global page; IVERSION ($BFFD) contains the version of the resident system interpreter, KVERSION ($BFFF) contains the version of the currently loaded ProDOS 8 kernel. These two program lines read and print the contents:*

```
1000 PRINT "IVERSION = ";PEEK (49149)
1010 PRINT "KVERSION = ";PEEK (49151)
```

The KVERSION value is incremented every time there is a change in the ProDOS 8 kernel, so it may not reflect a change in the version number displayed on the ProDOS startup-screen (for example, the KVERSION value for both ProDOS 1.8 and 1.9 is $08 since the kernel didn't change). BASIC.System v1.4 and 1.4.1 both have IVERSION values of $04.

We built a table with a series of values for ProDOS 8 and BASIC.System files we happen to still have languishing about. It turns out the KVERSION value for ProDOS 8 matches the "x" value in the version number "1.x" up until v1.9 (as mentioned). For example, the KVERSION value for ProDOS 1.6 is 6. This works nearly the same for BASIC.System; the anomally is version 1.2 where the IVERSION value was left at 1. Given the precedent, we're not sure the KVERSION and IVERSION values will reflect the version number as displayed on the startup screen.—DJD

Apple HD 20 and IIc

I have access to an old Apple HD 20 from a Mac which hooks into a 3.5 inch drive port. Has anyone sucessfully got one to work with a IIc? (I haven't tried to hook it up—visions of smoke curling up from the chassis.)

W. E. Boardman
Lynnwood, Wash.

We haven't heard of such a thing, but maybe someone out there has and will tell us.

*Look **carefully** at the hard disk connector and make sure it isn't a SCSI connector. Both the DB-19 (**19** pins) connector used by the floppy disk and the DB-25 (**25** pin) connector Apple uses for the SCSI port are shaped similarly, but they are electronically very different. We've had users confused by the general similarity in appearance.—DJD*