A2-Central™

formerly Open-Apple

February 1989
Vol. 5, No. 1

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

A journal and exchange of Apple II discoveries

Fonts: on screen and in print

If you're going to be a citizen of the Apple II kingdom in the coming decade, it's becoming clear that you're going to have to learn about fonts. Once upon a time, only printers (Gutenberg's offspring, not the ImageWriter's ancestors) had to know all about fonts, but those days are now only a memory, like manual typewriter returns.

Although handwriting did once capture our written personalities, this is 1989 and third-graders are desktop-publishing book reports. Today you may feel that the typefaces built into your ImageWriter can capture as much of your personality as you want captured, but tomorrow's graphics-based personal printing will capture your imagination as well as your personality. The future will offer you multiple opportunities to express yourself—and to look like a clown if you don't know all about fonts—so pay attention.

Fonts in printing. Traditionally, the word "font" has meant a complete set of type (all the upper- and lower-case letters, numbers, and punctuation marks) in one face, style, and size. This is exactly what the word means in today's context as well. However, instead of consisting of wooden boxes filled with heavy lead castings, today's fonts consist of files you can store on your disks filled with ethereal digital images.

The "face" or "typeface" of a font refers to its basic design. In desktop publishing, this quality is also referred to as the font's "family". Some important elements of a family design include serifs (fine lines that finish off the main strokes of a character, such as the little feet and hats you can see if you take a close look at this M) or the lack of them (called a "sans serif" family) and the relative height of capitals, lower-case letters (called the "x-height"), ascenders (the uprights on "d", "h", and friends), and descenders (the legs dangling under "g", "j", "p", "q", and "y").

Traditionally, "style" refers to whether a font is plain, **bold**, *italic*, or ***bold-italic***. Here in the age of desktop publishing, underline, outline, shadow, and SMALL CAPS are other styles you'll occasionally see (very rarely, let's hope). In the days of lead castings, each style (and each combination of styles, such as bold-italic) required a separate font. On the Apple II, styling is typically done by electronic manipulation of a plain font.

Font "size" is traditionally expressed in a unit of measure called the "point". There are 72 points to the inch. A font's size is usually considered to be the distance from the lowest descender to the highest ascender. In print shops, the size doesn't include any of the white space between lines. This white space is called "leading" (rhymes with "red", not "reed"). Thus, "10 on 12" describes a 10-point font on a 12-point line—2 points of leading are inserted below each line of type. Nowadays, the size of a font, like its style, can be manipulated electronically, however, the results are often unsatisfactory, particularly when a small font is used to make a larger one.

Some typefaces or font families are designed primarily to be easy to read. Other font families are designed primarily to attract attention. When designing documents, please do the universe a favor and save the attention-grabbing fonts for grabbing attention. If you set an entire block of text in an attention-grabbing font, you lower both its readability and the probability that anyone will read it.

Another easy document-design rule pertaining to typefaces is to limit each of your documents to just a couple of font families. Most printed material uses a serif typeface for text and a sans serif typeface for headlines, or perhaps the reverse. Well-designed documents create emphasis and flow by varying the size and style of the fonts, not by changing font families.

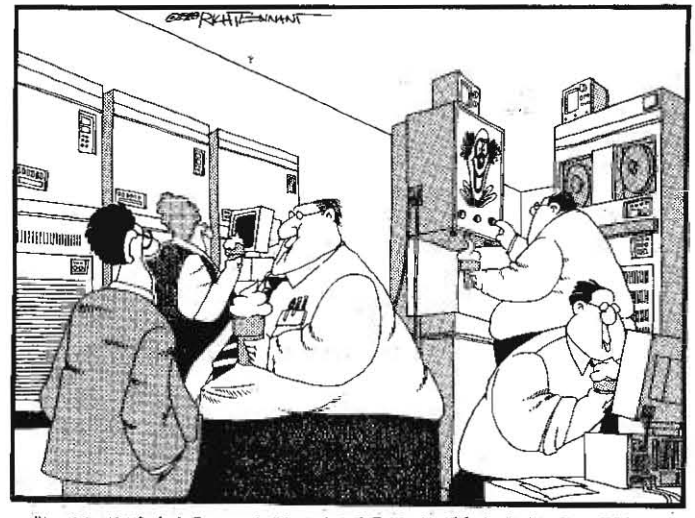
While ***A2-Central*** isn't about to win any awards for graphic design (too many large blocks of small type) we do stick to the same three font families for everything we do—from the newsletter itself to promotional materials to advertising to disk labels. The large "A2-Central" at the top of our front page and the text you are reading now, as well as our italicized answers to readers' letters, are all various sizes and styles of a serif typeface called Benguiat. Our headlines and the small type used in figures and in the copyright/warranty box on the back page are all various sizes and styles of a sans serif typeface called Helvetica.

We also use a third typeface, Courier, when we want to show something as it would appear on your computer screen. Unlike most font families used in printing, Courier is a non-proportional, or monospaced, font. All its characters are exactly the same width. This means that the letters in a block of text set in Courier will line up in columns, just as they used to do on your typewriter and still do on your 80-column computer screen.

While a single document should limit font usage to just a couple of families, deciding exactly which families will create the look and feel you want can be a big job. There are over 500 families available right now for use with software such as *AppleWorks GS* and *SuperFonts*.

Font use on personal computers. The Apple II is the first personal computer alphabetically, chronologically, and in the use of fonts. Early font usage on the Apple II, however, was limited to screen displays. Later, software was developed that popularized dot-matrix printing with fonts. Three early software packages that allowed the use of fonts on the screen were Apple's *DOS Toolkit*, *Higher Text II* from Synergistic Software, and *Apple Mechanic* from Beagle Bros. *Apple Mechanic* was the first program to offer proportional fonts (characters of various widths) on the Apple II.

Each of these three programs had its own special format for fonts.



"WE'RE USING A 4-TIERED SYSTEM-APPLE II, TO APPLETALK, TO MR. SMOOTHY, TO MAINFRAME."

This lack of standardization quickly became the standard. Publishers of later programs that allowed printing with fonts, such as Data Transforms with *Printrix* and *Fontrix* and Springboard with *Newsroom*, each designed its own unique font format. Broderbund, the exemplar among publishers who make users' lives miserable through copy protection and non-standardization, again led the pack by developing *two* unique font formats, one for *Print Shop* and another for *Print Shop GS*.

Fortunately, with the advent of the Apple IIgs, Apple created and documented a standard font format. The format is based on the format of Macintosh fonts, only it's better. Not only all non-Broderbund IIgs programs, such as *AppleWorks GS*, but also IIe/IIc/IIgs programs such as Beagle Bros *SuperFonts* and TimeWork's *Publish It!*, use the new format (or a slight variation).

A font clearinghouse. Because of the movement in the Apple II community to a single font format, and because of the extreme confusion in the Macintosh community about fonts, we recruited a subscriber and GEnie user with a large collection of and interest in fonts, Mark T. Collins from Waukesha, Wisc., to create an ***A2-Central Font Clearinghouse*** on GEnie. What Collins is trying to do is bring together a large library of public domain IIgs fonts that are bug-free and compatible with each other.

We've put a lot of Collins' work on this month's ***A2-Central*** disk, including his shareware font editor, called *Font Doctor*. We've also included a nice selection of public domain fonts, samples of fonts, programs related to fonts, and two AppleWorks data base files that list the specifications of the fonts currently in the GEnie clearinghouse collection.

While I'm mighty proud of the work Collins has done for us on GEnie, there are other sources of public domain IIgs fonts you should be aware of. The other national online services, of course, include many font files in their libraries, as do most user groups. TechAlliance (formerly Apple Co-op, 290 SW 43rd St, Renton, WA 98055) has several public domain font disks available for $3.00 + .50 each. Beverly Cadieux, 3103 Lake Stream Dr., Kingwood, Texas 77339 has put together a set of six 3.5 disks with about 500 font files that she will send you for $30.00 + $3.50.

In addition to public domain fonts, shareware and commercial fonts have been developed for the Macintosh and some of these are in the process of being converted to Apple II format. The only commercial fonts originally developed under the IIgs-standard that I'm aware of were published by StyleWare. However, since Claris purchased StyleWare that product has been in limbo.

TimeWorks also sells a font disk for *Publish It!* These fonts follow the IIgs standard except for one small detail—they have a file type of $F7 rather than the standard $C8. To use standard IIgs fonts with *Publish It!* you have to change the filetype to $F7. To use *Publish It!* fonts with software that follows the standard, you have to change the filetype to $C8. You can do this using disk zap software if you know what you're doing. Or you can do it with a public domain program sent to us by William Olsen of Riverside, Calif. The program is in our library on GEnie in a file called CHANGE.FONT.FILETYPE.BQY and it's on this month's ***A2-Central*** disk in a file called CHG.FNT.FILETYP.

At one time StyleWare also advertised that it would publish a font editor for IIgs-style fonts so that each of us could create new fonts or modify existing ones. However, that product didn't make it to market. Beagle Bros is working on a font editor, but it isn't expected to be ready until later in 1989. At this time, the only IIgs font editor we're aware of is Collins' *Font Doctor*.

Aspects of fonts. Neither computer monitors nor dot-matrix printers are able to display fully-formed characters. What they do display is very small dots, or "pixels". The essence of a font file is a description of which pixels are foreground (or "on") and which background (or "off") for each character in the font. If pixels, monitors, and sheets of paper were always perfectly square, life would be fair. But they're not and it isn't.

While traditional printing-press printing uses what we consider fully-formed characters, pixels aren't alien to the print shop. Photographs are reproduced by "screening" them, which converts each photograph into a matrix of pixels. The term "resolution" refers to the density of the pixel matrix. The smaller each pixel is, the more of them a printer can cram into a unit of space. And the more of them that get packed in, the more "information" there is and the better looking the graphic.

A typical printed photograph has between 120 and 150 multi-shade pixels per inch both horizontally and vertically. An ImageWriter, on the other hand, has 72, 80, 96, 107, 120, 136, 144, or 160 black/white dots per inch horizontally and either 72 or 144 dots per inch vertically. Even though the ImageWriter, at 144 x 144, can match the resolution of a printed photograph, its lack of multi-shade dots means it can't match the look of printed output.

While the Apple II was first with fonts, the Macintosh was the first computer to use the IIgs-standard font format. It will be helpful —believe me—if we look at the Macintosh a few minutes to understand how we got where we are today.

For years the Macintosh was sold only with a built-in monitor and all Macintosh monitors were exactly the same size (a little small). When a Macintosh is properly adjusted, its active screen area is 18 centimeters wide by 12 centimeters high—a ratio of 1.5. Meanwhile, the pixel resolution of the monitor is 512 wide by 342 high. Manipulate these numbers mathematically for awhile and you'll find that the Macintosh screen has a resolution of 72 pixels-per-inch both vertically and horizontally. Macintosh screen pixels are exactly one printer's point wide by one printer's point high.

One of the design goals of the Macintosh was to create a computer that could display on its screen what it would print on its printer—what you see is what you get, or WYSIWYG (wizzy-wig). The goal was met not only in terms of fidelity to the proportions of the material on the screen, but in terms of fidelity to its exact size. The Macintosh screen displays a 72 x 72 pixel-per-inch matrix and the ImageWriter duplicates that matrix exactly.

(Well, sort of. I own four "consistent-interface" Macintosh programs. Most of them have a standard item under the "File" menu called "Page Setup". *MacPaint*, which prints at either 72 x 72 ("Print Draft") or 144 x 144 ("Print Final") lacks the standard Page Setup choice. *AppleLink Industrial Edition* prints using the ImageWriter's text characters and although Page Setup appears in its File menu, it's never selectable. *Quark XPress*, my desktop publishing program, and *MacWrite* both have a selectable Page Setup. Among the items in the Page Setup dialog box for the ImageWriter (Page Setup dialog box contents vary depending on what kind of printer you are using) is a goodie called "Tall Adjusted". When Tall Adjusted is selected (the default in *Quark XPress*), you get 72 x 72 (Quality=Faster in the Print dialog box) or 144 x 144 (Quality=Best in the Print dialog box) square ImageWriter pixels per inch. When Tall Adjusted is *not* selected (the default in *MacWrite*), you get 80 x 72 (Faster) or 160 x 144 (Best) slightly-tall pixels per inch. If you import a perfect circle from *MacPaint* into a document and print the document with Tall Adjusted off, you get an ellipse. I suspect *MacWrite* has a Tall-Adjusted-off default so that it naturally takes advantage of the ImageWriter's maximum resolution (160 x 144) but I'm not at all sure this is reason enough to contaminate the Mac's otherwise perfect wizzy-wig design (notice that when you flip the Page Setup to Tall Adjusted in *MacWrite* the ruler's inch measurements are exact (if your active screen image is the specified 18 centimeters horizontally), while with Tall Adjusted off they're not). While we're on the subject, "Best" quality printing on the Macintosh gives you four times more resolution than "Faster". When Best is selected, the Macintosh automatically looks around for a font twice as big as the one being printed. If it finds one, it uses it to increase the resolution of the printed output. This trick increases sharpness without changing the printed font size. Beagle Bros' *SuperFonts* does the same thing and can match Macintosh output exactly on an Apple II (it even has a Tall Adjusted toggle), but you don't get to see what your printed page will look like while you're editing it.)

When we talk about printing fonts on paper or displaying fonts on the Macintosh monitor, we can talk about pixels-*per-inch*. When we move to talking about *Apple II* monitors, however, the per-inch concept drifts into the rasters. That's because Apple II monitors come in a wide variety of sizes, yet they all display the same number of pixels horizontally and vertically. A "9-inch" monitor has smaller, denser pixels than a "19-inch" monitor, but they both have exactly the same number of pixels in total. The number of pixels *per-inch* depends on the size of the monitor and becomes fairly meaningless. The variety of monitor sizes also makes it very difficult to have any kind of fidelity between the size of an image on the screen and the size of that image when it is printed. To accomplish that kind of fidelity you'd have to

make sure that everyone had the same size screen (as Apple did with the Macintosh).

Furthermore, the ratio between the height and width of the screen image depends on the position of those small adjustment knobs hidden on the back or inside most monitors. The standard aspect-ratio for Apple II monitors (and for television screens) is 4 units wide by 3 units high, a ratio of 1.333 (not the 3 x 2, 1.5 ratio of the Macintosh). However, by turning the adjusting knobs, most monitors can be set across a relatively wide aspect-ratio range. I suspect very few of you have monitors that have been adjusted so that the aspect ratio of the active screen area is exactly 1.333. This lack of standard adjustment makes it virtually impossible to have any kind of fidelity between the vertical-horizontal proportions of an image on the screen and the proportions of the image when it's printed. Every day, Apple II users draw perfect circles on their monitors and then print eggs. Now you know why.

The problems created by the variety of monitor sizes and aspect ratios in the Apple II kingdom, however, pale compared to the problems created by the fact that the Apple II has more horizontal resolution than the Macintosh, but far less vertical resolution. The two graphics modes usually used when IIgs-standard fonts are displayed on the Apple II screen are double-high-resolution, which has 560 pixels across and 192 up and down, and 640-mode super-high-resolution, which has 640 pixels across and 200 vertically. Notice that both double-high-res and 640-mode super-high-res have more horizontal resolution than the Macintosh (560 and 640 compared to 512). The limitation of Apple II graphics is the vertical resolution—just a little more than half that of the Macintosh (192 and 200 compared to 342).

(The big limitation of Macintosh graphics, on the other hand, has been the lack of color. A monitor's ability to do color, its vertical resolution, and its cost are all related to one another. In the Apple II we have a computer that can do color work on relatively inexpensive monitors, but with limited vertical resolution. The original Macintosh could squeeze a lot of vertical resolution onto a relatively inexpensive monitor, but it couldn't do color. Apple added color capability to its Macintosh II without giving up any vertical resolution, but for the price of that machine's color monitor and video card you can get Apple's limited-resolution IIgs color monitor and have the IIgs itself thrown in for free.)

Ah, but what does all this screen-resolution mumbo-jumbo have to do with fonts? The simple truth is that almost all of the IIgs fonts available today were originally designed using the square pixels of the Macintosh. By fiddling with the vertical- and horizontal-size knobs on the back of your Apple II's monitor (Apple's own monitors don't have horizontal-size knobs, but some others do), you can also get square pixels on an Apple II. However, square pixels are only possible in standard-high-res (aim for an aspect ratio of 1.46). In the more commonly-used double-high-res and 640-super-high-res modes, pixels are tall and skinny—usually a little more than twice as high as wide—no matter how much you fiddle with the adjustment knobs.

When fonts designed for the square pixels of the Macintosh are displayed using the tall, skinny pixels on an Apple II, the once well-proportioned characters become tall and skinny, too. (Characters that originally looked tall and skinny on the Macintosh nearly disappear on the IIgs.) With any given font you can actually get a few more characters across the screen on a Apple II than on a Macintosh because of the additional horizontal resolution. However, you can only get about half as many lines on the screen. And those half-as-many lines get stretched vertically so that they touch both the top and bottom of the active screen area. That's why we have those tall, skinny characters and pixels.

And it all boils down to this: you are sitting in a warm computer room. You have a document on the screen in front of you. The document uses a font originally designed for the Macintosh. You see tall, skinny characters. You want to print this document. But do you want your printed output to duplicate *exactly what you see on your screen* (like the Macintosh) or do you want your printed output to have *well-proportioned characters* (like the Macintosh)? What a dilemma.

If you own the standard IIgs-issue AppleColor RGB Monitor and want your printed output to look *exactly* like what you see on your screen (a typical wish when using drawing programs, for example), start by turning the knobs on the back of your monitor until your aspect ratio is about 1.44. According to the manual that comes with that monitor, the active display area is adjusted at the factory to 20 centimeters wide by 15 centimeters high (the expected ratio of 1.33). My own monitor's unadjustable width is only about 19.5 centimeters, so I've set the height to 19.5/1.44 or about 13.5 centimeters. Perfection would be an active display area 20.3 centimeters wide by 14.1 centimeters tall. Mathematically this works out to 80 pixels per inch horizontally and 36 pixels per inch vertically—exactly the resolution the IIgs uses when printing on the ImageWriter. Thus, if you can adjust your monitor's active display area to these dimensions, what you get on paper can exactly match what you see on the screen.

However, to turn "can" into "does" you have to understand three options you get in the IIgs Page Setup and Print dialog boxes. Like righteous Macintosh software, desktop software on the IIgs includes Page Setup and Print choices in the File menu. The three critical options are:

Orientation. Page Setup gives you an Orientation choice of Portrait or Landscape. Portrait is what we'd normally call "normal"; Landscape is what we'd normally call "sideways". To get exactly 80 x 36 pixels per inch on an ImageWriter you need to select Portrait. Because of the limitations of the ImageWriter when you turn it on its side, Landscape printing is done at 72 x 40 pixels per inch, which creates slightly shorter, fatter characters.

Aspect ratio. Also hidden away in the Page Setup dialog box is an essential item called "Vertical Sizing". *To create an exact duplicate of what you see on your screen on your printer, you must set Vertical Sizing to* ***normal.*** But if you are printing clever documents using fonts, you probably don't really want to print those tall, skinny characters you see on your screen, do you Bobo? *To print text with fonts in their "correct" (**MacWrite's** 80 x 72) proportions, you must set Vertical Sizing to* ***condensed.*** Like the Macintosh's Tall Adjusted toggle, Vertical Sizing flips between exactly what you see on your screen (normal) and better-looking text (condensed). There is no Tall Adjusted toggle on the IIgs, just as there is no Vertical Sizing toggle on the Macintosh.

Resolution. Finally, the IIgs Print dialog box lets you select between "Better Text" or "Better Color". If you select "Better Text", the IIgs will look around for a double-sized font and print 160 x 72 pixels per inch (normal) or 160 x 144 (condensed)—the equivalent of the Best Quality selection on the Macintosh. (If the selected font isn't exactly double the height and width of the original, wave good-bye to wizzy-wig.)

The following table summarizes all this and a bit more:

IIgs printing resolutions in pixels per inch

Screen in 640-mode

	portrait orientation		landscape orientation	
	better color	better text	better color	better text
normal	80 x 36	160 x 72	72 x 40	144 x 80
condensed	80 x 72	160 x 144	72 x 80	144 x 160

Screen in 320-mode

	portrait orientation		landscape orientation	
	better color	better text	better color	better text
normal	40 x 36	80 x 72	36 x 40	72 x 80
condensed	40 x 72	80 x 144	36 x 80	72 x 160

Many subscribers have mentioned how slowly GS/OS programs print. It appears to me that the reason is that the GS/OS ImageWriter printer driver *always* makes the ImageWriter print at its maximum horizontal resolution of 160 dots per inch. If the size of the pixel to be printed is larger than that, GS/OS will print several 160 dpi dots per pixel. For example, you can see in the above table that 640-mode portrait orientation in Better Color has a pixel resolution of 40 x 36. However, GS/OS actually has the ImageWriter print 8 dots (160 x 72) of various cyan, magenta, and yellow combinations to create that pixel. While "Better Text" on the IIgs is equivalent to "Best Quality" on the Macintosh, "Better Color" is *not* equivalent to the Mac's "Faster Quality". Once again, we find ourselves sacrificing something in comparison to the Macintosh (this time it's speed) in order to gain color.

The IIgs Print Manager uses multi-dot pixels with Better Color even if you don't have a color ribbon. If you *do* have a color ribbon, be sure to check the Color square in the lower-left corner of the Print dialog box. If you print in Better Color but don't have Color selected, the

Print Manager tries to do gray-scaling with black dots rather than full-color magenta-cyan-yellow mixes.

Font anatomy. Now that you understand all the various ways that a font can be printed, let's look at an actual font file.

A IIgs font file has four major parts. The first is the *header*, which contains all kinds of interesting information about the font. The second is the *strike*, which contains the pixel images of the characters in the font. The third is the *location table*, which has an entry for each character in the font. The entry points to the character's pixel position in the strike. The fourth is the *offset/width table*, which tells how each character is to be placed relative to the current "character origin" or "pen position" and how wide each character is. First we'll look at the strike and at the tables, then we'll come back and investigate the header.

The font strike. Two of the details we'll find in the header are a pointer to the start of the font strike and the strike's row width. The strike is just a very long sequence of bits in bytes, but conceptually the first bunch of bits in the strike represents the top row of pixels for

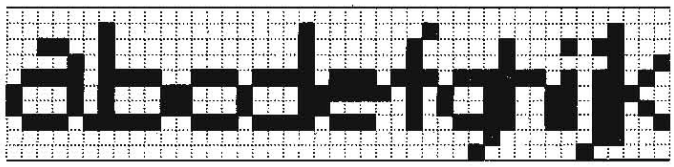


Figure 16-29
Part of a font strike

all the characters, the second bunch the second row, and so on. The number of bits in a bunch for any particular font is the row width. The characters in the strike are all pushed together with no spaces between them, so once you get the rows lined up, the strike looks like Figure 16-29, which is unabashedly stolen from the QuickDraw II chapter of Apple's *Apple IIgs Toolbox Reference, Volume 2*.

Each row of the strike is padded on the right end with enough extra bits to make the row width end on a "word" boundary (a "word" is two bytes). Not every possible ASCII code has to have an image in the strike. The header includes the ASCII code of the first (*firstChar*) and last (*lastChar*) characters in the strike; characters outside this range are defined as "missing". (Character codes can range from 0 through 255, however, the ASCII standard only defines the codes from 0 thorough 127 and Apple has no recommendation other than to look at existing fonts for 128-255.) Characters within the *firstChar-lastChar* range can also be "missing," as we'll see in a moment. Immediately after the pixel image of *lastChar* in the font strike (at *lastChar* + 1) is an extra character that is used in the place of all missing characters. Apple calls this character the "missing symbol" and has made it a required element in the font strike (even if there aren't any missing characters). Usually it's either an empty box or an inverse question mark, but sometimes a designer's copyright is crammed in here.

The location table. The location table is an array of bytes with a two-byte (one-word) entry for each character code from *firstChar* to *lastChar* + 2. The entry holds the distance, in pixels, from the beginning of each row of the strike to the left edge of the character's image.

To find the width of a character's image, subtract the location-table entry for the character in question from the location-table entry for the following character. The difference is called the "image width". If the character has no pixels at all, such as the space character or a control code, the image width can be zero (but it can't be negative). For this scheme to work, the location table has to have an entry for *lastChar* +2. This entry must point to the pixel just to the right of the "missing symbol", which is the same thing as holding the total length of the font strike in pixels (ignoring any word-boundary padding).

The offset/width table. Like the location table, the offset/width table is an array of bytes with a two-byte entry for each character code from *firstChar* to *lastChar* + 2. If the two bytes for a character hold $FFFF, that character is missing from the font. Otherwise the first byte of each entry holds the "character width," in pixels; the second byte holds the offset. Both the width and offset can range from 0 to 254 pixels (not 255, because of the need for $FFFF).

The *width* is the distance, in pixels, that the "character origin" should be moved (to the right only) after the character is drawn. Some characters, such as SPACE, will have a positive character-width (the character origin moves to the right when the character is typed) but a zero image-width (no pixels in the character image). Other characters, such as accent marks, may have a positive image-width (a pixel image) but a zero character-width (the accent mark is drawn but the origin isn't moved so that the next character typed will appear under the accent). Other characters, such as control characters, will have both image- and character-widths set to zero.

The *offset* tells QuickDraw how to position the character with respect to the character origin. Characters can start either to the right or to the left of the character origin. Since the offset is always a one-byte positive number, however, we need an imaginary point somewhere to the left of the character origin to start the offset from. This is provided in the header by an entry called *kernMax*, which is the maximum distance, in pixels, from the left-most edge of the left-most character in the font to the character origin. If *kernMax* is negative, that left-most edge lies to the left of the character origin; if zero, on the character origin; if positive (unlikely), to the right of the character origin. To figure out where to position a character image, QuickDraw adds the character's offset to *KernMax*. If the resulting number is negative, the left edge of the character lies that many pixels to the left of the origin; if zero on the origin; if positive that many pixels to the right of the origin.

Figures 16-26 and 16-27 show a couple of sample characters from an imaginary font. Notice that all the lines and points that things are measured from lie *between* pixels. The image width has to do with the number of pixels in the character, 5 for the "y", 7 for the "f"; the character width has to do with how far the character origin is moved, 7 for both the "y" and the "f". If, in this imaginary font, *kernMax* was -2, the offset of the "y" would be 2 and of the "f", 1.

The header. The header of a IIgs font consists of three distinct parts. The first part holds the name of the font. The second part is the *IIgs header*. The third part is the *Macintosh header*.

The very first byte of a font file tells how many characters are in the font name. The following bytes spell out that name. The IIgs header begins immediately after that. The first two bytes of the IIgs header tell how far it is from there to the Macintosh header.

The Macintosh part of the header, as well as the strike, location table, and offset/width table, are exactly like those in a Macintosh font except for one detail: the high- and low-order bytes of two-byte

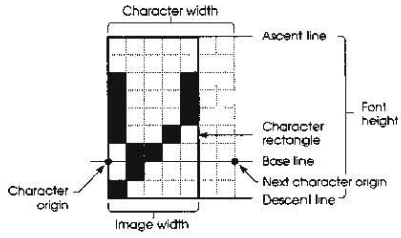


Figure 16-26
Character with no kerning

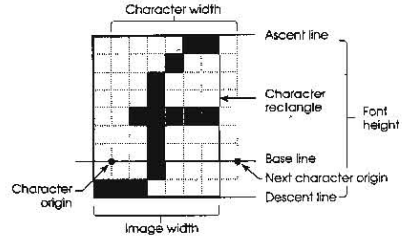


Figure 16-27
Character kerning left

integers are reversed. This doesn't apply to the font strike, but the two bytes in every other word in the Macintosh font need to be flip-flopped. The origin of this requirement isn't perversity, but the forward-thinking architecture of the Apple II's microprocessor.

The first two bytes of the Macintosh section of the font are the Macintosh "font type," which is ignored in the Apple II kingdom. The next two bytes are the ASCII code of the first defined character, *firstChar*. The next two are the ASCII code of *lastChar*. The next two are *widMax*, the character width (in terms of origin movement) of the

widest character in the font. The next two are *kernMax*. The next two are the "negative of descent", which I'll talk about more in a moment.

If you took all the characters in a font and typed them one on top of another with all their character origins in the same place, you'd have a black mess. The smallest rectangle completely enclosing this mess is called the *font rectangle*, according to Apple's toolbox manual. The next two bytes give the *width* of the font rectangle, in pixels. The next two give the *height* of the font rectangle.

The next two bytes give the distance, in words, from this field in the header to the beginning of the offset/width table. To get the distance from here to the beginning of the location table, you must subtract 2 * (*lastChar* - *firstChar* + 3) from the offset/width table distance.

The next two bytes hold the font's *ascent*, the number of pixels from the font's base line to the top of the font rectangle. The next two bytes hold the font's *descent*, the number of pixels from the font's base line to the bottom of the font rectangle. And yes, *ascent* + *descent* must always be equal to the height of the font rectangle. The next two bytes hold the recommended leading (number of blank pixel rows between the lowest row of one line of characters and the highest row of the following line) for this font. Applications may either use or ignore this value. The "negative of descent", disregarded earlier, seems to be disregardable. There's little relationship from font to font in the number stored here. Some have *-descent*, some *-ascent*, and some some other number.

The next two bytes give the width of each row in the font strike, in words. The next byte is the start of the strike. The length of the strike is the row-width multiplied by the height of the font rectangle. The location table follows the strike, and the offset/width table is at the end of the font file.

The IIgs part of the header adds several pieces of information that were left out of the Macintosh header. As mentioned earlier, after the embedded font name comes a two-byte offset, in words, from this field to the Macintosh part of the header. In the current version of the IIgs font format (version 1.1), this field holds a 6. Apple may expand the IIgs font header in the future to add additional information. Software that uses this field to find the Macintosh part of the header, however, will be totally compatible with any future versions of the IIgs font format.

The next two bytes of the IIgs header hold the font's family number. All fonts in the same family (i.e., all Geneva fonts or all Helvetica fonts) should have the same number here. On the Macintosh, one byte of the "font type" is used for the family number and the other is used for font style (plain, bold, italic, etc.). This means only 256 family IDs are available on the Macintosh, but there have been far more font families than that for years already. Overlapping font IDs are a real problem in the Macintosh world. Here in the Apple II kingdom, on the other hand, we have 65,536 available IDs. It's important that all the fonts you use have matching IDs if they are in the same family and unique IDs if they are in different families. If they don't, the print manager can get mixed up and use the wrong font.

One of the major reasons we started the ***A2-Central Font Clearinghouse*** on GEnie was to keep our kingdom's font IDs organized. Apple Developer Technical Support doesn't assign IDs to individual fonts. Instead, it assigns ranges of numbers to companies or groups who distribute fonts. The groups, in turn, can assign numbers within their range to individual font families as they choose. Table 1 is a list of ID assignments as of mid-January 1989. The ***A2-Central Font Clearinghouse,*** for example, has been assigned 2,048 font-family ID numbers between $7C00 and $83FF.

While most of the assigned ranges have gone to commercial companies, we intend to reassign the numbers in our range to fonts designed by individuals. You can get your own font ID number from us simply by designing a new font family and allowing us to put it in our library. We accept either public domain or shareware fonts.

One interesting bit of information about a font family is whether it is designed to be displayed at the Macintosh square pixel 1:1 ratio or at the IIgs tall-skinny pixel 11:5 ratio. While condensed printing solves the tall-skinny character problem when you're *printing* 1:1 fonts, programs that are screen-oriented rather than paper-oriented need some fonts that are designed to look fat and happy on the *screen*. We're trying to give new 1:1 fonts family numbers below $8000 (32767) and new 11:5 fonts family numbers of $8000 and above. Of existing fonts, most of *Publish It!*'s fonts and a few of Styleware's are 11:5.

Table 1: Apple IIgs Font Family Number Assignments, January 1989

DECIMAL	HEX	
	– 1:1 Ratio (Macintosh format fonts) –	
0- 127	00- 7F	Apple's Macintosh fonts
128- 255	80- EF	vendor Macintosh fonts
240- 255	F0- FF	also used by Styleware (Claris)
256- 991	100- 3DF	–reserved for future assignment–
992- 1055	3E0- 420	-assigned to Styleware (Claris)
1056- 8191	421-1FFF	–reserved for future assignment–
8192- 8447	2000-20FF	-assigned to unknown vendor
8448-27903	2100-6CFF	–reserved for future assignment–
27904-28159	6D00-6DFF	-assigned to unknown vendor
28160-31743	6E00-7BFF	–reserved for future assignment–
31744-32255	7C00-7DFF	GEnie A2-Central-New IIgs Families-1:1 Ratio
32256-32767	7E00-7FFF	GEnie A2-Central-Macintosh Conversions
	– 11:5 Ratio (640 SHR format fonts) –	
32768-33791	8000-83FF	GEnie A2-Central-New IIgs Families-11:5 Ratio
33792-65279	8400-FEFF	–reserved for future assignment–
65280-65535	FF00-FFFF	Apple's IIgs Fonts

The third two-byte word in the IIgs section of the font header denotes the font's style. The algorithms used to electronically change plain fonts into bold, italic, and so on don't work well with some fonts. The style byte allows the IIgs to have prestyled fonts. Fonts that are prestyled are not styled further—for example, if you have a prestyled italic font and italic checked in your Style menu, the italic font isn't italicized a second time electronically. It could still have bold or other styles added, however. The meaning of the bits is shown in the following table:

Meaning of bits in the font style byte:

```
 7  6  5  4     3  2  1  0
 0  0  0  |     |  |  |  Bold     (Plain=all bits 0)
reserved  |     |  |  Italic
          |     |  Underline
          |     Outline
          Shadow
```

Only the low-order byte of the style word has any meaning. A "1" bit means the style has been applied. A byte value of zero means plain style. The bits are cumulative, for example, $0011 means bold-shadow. On the IIgs, underline requires a descent of a least three pixels. That's why the IIgs system font, Shaston 8, won't underline.

The fourth two-byte word in the IIgs section of the header gives the point size. On the Macintosh, point size is given as part of the file name. Since Macintosh screen pixels are, in fact, exactly one printer's point high, one would expect Macintosh fonts to be given "sizes" exactly equal to their font height in pixels. But when you examine the headers of a number of fonts, you find that this is the exception rather than the rule. In some cases, it appears that the font size was set equal to the *ascent*, which, while non-standard, does have at least some meaning.

In the majority of fonts, however, there is little relationship between "point size" and the actual pixel height of the font. For example, our library's collection of fonts claiming to be 9 point actually have pixel heights ranging from 5 to 14, with stops at all pixel heights in between. Fonts claiming to be 24 point have actual pixel heights ranging from 13 to 40 and beyond. While there is some point-size meaning *within* families (a 24 point Shmoe is almost always about twice as big as a 12 point Shmoe), *between* families there is almost none.

Even within families, fonts that are billed as exactly twice the size of smaller fonts are often a few pixels off. If you select a high-quality printing mode that looks around for a double-point-size font, your entire document can be reformatted when the double-size font isn't really double size. Because of the mish-mash of point sizes we've inherited from the Macintosh, what you see may not be what you get at all. The lack of relationship between stated point sizes and the actual pixel height of fonts is detrimental to the mental health of our community.

The fifth two-byte word in the IIgs section of the header gives the version number of the font format. All fonts that have the 6-word IIgs

header discussed here should have $0101 (v1.1) stored in this word.

The sixth and last two-byte word in the IIgs section of the header gives the "font bounds rectangle extent". This is the farthest possible horizontal distance (right or left) from the character origin to any pixel that can be altered by drawing in any character in the font. If this value is smaller than it should be, QuickDraw II can overwrite memory locations that don't belong to it and create all kinds of havoc. This value must always be equal to or greater than both *kernMax* and *widMax* (the largest character width in terms of origin movement). If any character in a font kerns to the right of the most distant new character origin, this value will be larger than *widMax*.

In addition to the problems with point size and extent we've just discussed, there are a number of other bugs lurking in font files. For example, some fonts have entirely blank rows of pixels at the top or bottom of the font. This "forced leading" is against the rules. Others have impossibly large *kernMax* and *widMax* values.

One of the primary tools I've used to learn all this is the AppleWorks data base file Mark Collins constructed that contains all the header information from the fonts in our clearinghouse. First he wrote a program that dug the values out of the headers and put them in a text file, then he converted that file into an AppleWorks data base file. This file is available on this month's disk and in our library on GEnie. You can use it to find out, for example, the true pixel height of a font. You can use it to find out how many font sizes are available in a family and whether the larger fonts are really exactly double the size of smaller fonts.

That's all I know about fonts so far. I expect to learn more as the months go by and our font clearinghouse grows. Meanwhile, if you're looking for exactly the right font to catch your personality on paper, you know where to look.

Miscellanea

Computer Systems News, "The Newsweekly for the Computer and Systems Integration Business", lead off its year-end issue with a list of things that didn't change at all in 1988. At the top of the list was:

THE APPLE II: No matter how many new Apple IIIs, Lisas, and Macintoshes the company introduces, no matter how many times experts predict its demise, Apple's classic Apple II continues to flourish. It's worse than crabgrass!

From the *Wall Street Journal*'s "Business Bulletin", December 15, 1988:

COMPUTER SALES stay strong despite a predicted slowdown, retailers say. The big buyers: small and medium businesses, as well as people who want computers for the home....

And from Apple's 1988 Annual Report:

Our Apple II product family continues to be the most popular choice in the primary and middle schools, with an installed base over two million strong....Though we think of Apple II computers primarily in relation to schools, we found in 1988 that many families and small businesses are also buying the Apple IIgs, because of its all-around strengths as a personal computer. Not only can the IIgs run more educational software than any other personal computer, it can also run general productivity software—everything from word processing and home finance to small-business accounting.

Apple's December Dealer Service Notice reminded dealers that

Ask (or tell) Uncle DOS

Subscriber Tom Weishaar of Overland Park, Kansas called in a correction to my statement in the January issue, last paragraph on page 4.94, that AppleWorks 2.1 will "use as much memory as you have" on standard-slot memory cards. In fact, Weishaar said at the end of a long day of troubleshooting that followed a system reconfiguration, AppleWorks 2.1 immediately crashes if you have a standard-slot card with more than two-megabytes on it—even if there's less than 2 megabytes of free space. As soon as AppleWorks sees all that memory it hangs with its tongue drooping out.

GS/OS parallel driver

I'd given up hope of ever being able to print anything from a SYS16 program on my parallel Epson FX-85 connected to a SLOTBUSTER II card in my IIgs. Even the new parallel driver that comes with GS/OS bit the dust.

It turns out that the driver does an ID byte check of the parallel card. Randy Carlstrum at RC Systems, the company that makes the SlotBuster, solved the problem. Bload PARALLEL.CARD,A$2000,T$BB and check $22FD. The original value is $14. Change it to $31 for the SlotBuster II. Since the SlotBuster emulates the Super Serial card, this may work for other Super Serial card look-a-likes too.

Tuckerman Moss
Orinda, Calif.

When I got my copy of GS/OS I was very excited about the parallel printer driver. I thought my two-year wait was over and I could finally use my Apple Dot Matrix Printer and Apple Parallel Card with these great new IIgs programs. However, all I got was a message saying that the driver didn't recognize Apple's own parallel card. I paid over $1,300 Canadian in 1983 for this printer and card and believe there must be an alternative other than selling it off for $200 and buying an ImageWriter.

Lee Luker
Brandon, Man.

I don't know for sure, but I suspect Apple's driver can be made to work with any parallel card that follows the Advanced Firmware Protocol (also known as "Pascal 1.1"; see "Control-I(nterface) S(tandards)," October 1987, page 3.65 and "Using the Advanced Interface," January 1988, page 3.92) by patching the ID byte routine.

Apple's own parallel card doesn't follow this protocol, however. If I was positive that the Apple Dot Matrix printer could emulate all the necessary ImageWriter functions (I think it can) I'd recommend changing parallel cards and keeping your DMP. Remember, however, that it prints more slowly than an ImageWriter and people are already aghast at how long it takes to print on the ImageWriter using the GS/OS drivers.

Some stuff about Siders

I just had to replace the power supply in my 10 meg Sider I. I found the PS-ASTEC from JDR Micro Devices was a perfect fit electrically and physically and it only cost $25 as compared to $150 from First Class Peripherals.

Craig Peterson
Santa Monica, Calif.

I have an older 10 meg Sider that first boots DOS 3.3 and then loads and runs ProDOS 16. Now I want to use GS/OS. What options do I have other than buying a new hard drive?

Rich Katz
Los Angeles, Calif.

You can get an all-ProDOS ROM for your Sider Controller card for $59.95 plus shipping from Advanced Technical Services, P.O. Box 920413, Norcross, GA 30092 404-441-3322 (GEnie email address is STEVE.PARK). You'll have to reformat the Sider after installing the ROM. You'll get one large ProDOS volume that GS/OS is comfortable with.

Some stuff about SCSI

The Apple SCSI card seems to be incompatible with hard drives that use the Adaptec SCSI controller. The card has no problem talking to the drive. I can read and write blocks to any partition. But if you ask the card for the status of the partitions, even numbered partitions are ok while odd numbered partitions return status byte $00. I also have a SCSI card made by C.M.S.. It has no problem talking to the drive, but my drive is larger than 64 meg, which is the limit of the C.M.S. card, so I am interested in using the Apple card to get the full capacity of the drive.

Jock Cooper
Gallatin, Tenn.

After reading "An introduction to SCSI" in your December issue (page 4.85), I was still confused as to whether the Apple SCSI card would work properly with my drive. I have a C.M.S. SD20/A2S (stackable). So I decided to get one and try it. I'm pleased to say that it works flawlessly. The GS/OS thermometer works correctly. The Get Info button in

part 341-0437, Rev A is the current ROM chip for the Apple II SCSI card, that it's required for GS/OS, and that customers should get free upgrades. Apple's ProFile interface card also needs a new ROM to run GS/OS. This one is part 341-0299, Rev B, and is also part of a free upgrade program. Apple has also provided dealers with a new system diagnostic disk for the Apple IIgs (v3.1) that can identify RAM chips that aren't "CAS before RAS," as required by that computer. Apple's previous diagnostic programs didn't test for this and failed to identify this not infrequent cause of ghostly problems.

Timeworks has released *Publish It 2,* a more advanced version of its original desktop publishing program for the Apple IIe, IIc, and IIgs. The advanced version allows larger documents by including support for extended memory cards, can print on all PostScript printers, can use *Print Shop*-compatible graphics, and has several additional features not found in the original program. The original version is still being sold for $99.95, however. The new version is $129.95. Upgrades for registered users of the original *Publish It!* are available for $30 from Timeworks, 444 Lake Cook Road, Deerfield, IL 60015, 312-948-9200.

GS/OS tips: you can't boot GS/OS by executing the PRODOS file on a GS/OS system disk. It has to be a power-on or three-finger salute reboot. If the *Finder's* constant clicking on 5.25 and UniDisk 3.5 drives bothers you (its looking to see if you've switched disks), open up your SYSTEM folder, then your DRIVER folder, and click on the driver for the disk that's bothering you. Then press open-apple-I (Get Info). In the lower-right corner of the information dialog box is an "inactivate" selection. An "inactivate box" appears for anything in your DRIVERS, DESK.ACCS, or SYSTEM.SETUP subdirectories, but you have to reboot GS/OS to get your new selection to take effect.

In the envelope with this month's newsletter you'll find our annual index rather than a product catalog, so I'm going to squeeze in what's new at ***A2-Central*** this month right here.

Two new Apple II-related books have just been published. Brady Books has *Programming the Apple IIgs in Assembly Language* by David Eyes and Ron Lichty. Eyes and Lichty also wrote what many consider to be the Bible of the microprocessor in the IIgs, *Programming the 65816.* Their new book is $29.95; if you order from us our item code is SS-005.

Addison-Wesley is about to release *Using AppleWorks GS,* by Doug Brown. Brown was a member of the *AppleWorks GS* documentation team. His book's discussion of each module of *AppleWorks GS* includes a description of basic features, shortcuts, and sample applications. This book is $19.95; if you order from us our item code is AW-027.

Cirtech has raised the prices of its memory cards that use 256K RAM chips, which continue to be difficult to find. We'll sell our current stock of cards and stick by the prices in last month's catalog till February 28, but expect to see some price increases after that.

We're taking over Kansas City's Avila College campus on the weekend of July 21-22-23 for an ***A2-Central Developer Conference.*** For one slightly extravagant fee you'll get a conference teeming with Apple II hardware and software developers, all your meals, and—if you're at the head of the line—free dormitory accommodations. Apple has already committed to send us some engineers, evangelists, and support types and to throw a party, so plan now on coming to KC this summer. More details next month.

Advanced Disk Utilities returns "SCSI.1" instead of "DEV.1" as before. And the partition button is no longer dimmed.

Ron Kunkel
Machesney Park, Ill

I'm writing to add to December's discussion of generic hard drives for the Apple II. I built my own SCSI drive and have pertinent information on my BBS.

I ordered a Seagate ST277N 65 meg drive mechanism from Hard Drives International (1208 E Broadway Rd, #110, Tempe, AZ 85282, 602-784-1038, 800-234-DISK) for $449, a beautiful case with 30 watt power supply, fan, and all cables from Tulin Corp (2393 Qume Drive, San Jose, CA 95131 408-432-9025) for $119 and a C.M.S. SCSI card from CDA Computer (1 CDA Plaza, P.O. Box 533, Califon, NJ 07830, 201-832-9004, 800-526-5313) for $115. Hard Drives International also offers smaller capacity SCSI Seagate drives for $319 (20 meg) and $419 (40 meg).

The Seagate ST277N consumes approximately 13 watts of power. The Tulin case can hold two drives and its 30 watt power supply is more than sufficient to power two drives. Although cheaper cases can be had, then usually don't come with all the necessary hardware, such as cables, which may be difficult to find otherwise.

The Seagate ST277N mechanism is what C.M.S. used in my C.M.S. SD60 A2S drive. I selected their card rather than Apple's because I was concerned with compatibility with my existing C.M.S. drive.

Anyone wishing to enter the world of SCSI hard drives would do well to build their own at considerable savings. Construction is quite simple (if I can do it, anyone can). The drive mechanism attaches to the bottom plate of the case with four screws. One of the two plugs from the power supply connects to the back of the drive mechanism, as does the cable from the DB-25 connectors on the back of the case. A cable coming from the drive plugs into the power-on light on the case. And the top of the case is reattached using three screws. That's all there is to building it. Total construction time was about 10 minutes for the first one I built, five minutes for the second...and that included a two minute coffee break.

If anyone needs more help, they can contact me on my bulletin board, The Washington Towne Crier (201-689-3649, 300/1200/2400) and I'd be glad to assist.

Dr. Kenneth Buchholz
Washington, N.J.

Back in December, page 4.86, last paragraph in the first column, I said that C.M.S. SCSI cards don't work with certain other cards in the computer. C.M.S. has discovered that the source of the problem was a batch of defective Mitsubishi chips. The bad chip is the biggest chip on the card—C.M.S. will give you a free upgrade if you have a chip from the defective batch.

If you add a generic SCSI drive to your system using an Apple SCSI card and you run GS/OS, remember to use the Installer to add the SCSI driver to your System Disk. Until you do that, GS/OS won't recognize the drive.

Control Panel straightener

My wife teaches biology at a school that uses many Apple IIs. One problem they are having with the IIgs is that they waste a lot of time trouble shooting and correcting altered control panel settings. Is there a way to lock the control panel so that students can't change the settings?

Allan Seidel
Olivetter, Mo.

You could play games with configuration programs that might slow the students down, but any student worth his or her own computer will quickly figure out how to work around it.

The easiest solution is to boot the IIgs with ***option****/control/reset (this is the standard three-finger salute, but hold down the option key instead of open-apple.) This brings up a menu with four choices. Choose the second one, "Set system standards and 60 hertz." This will reset the entire IIgs control panel to normal (checkmarked) settings.*

Rock around the clock

If you use Apple's System Disk 3.2, keep the ALARM.CLOCK NDA out of your SYSTEM/DESK.ACCS folder. With this NDA installed, your SYS16 software will hang when you try to print.

If you use GS/OS, keep the MENU.CLOCK file out of your SYSTEM/SYSTEM.SETUP folder. This file, which inserts a clock into your menu bar, won't let you run your ProDOS 8 software but will drop you into the monitor or hang your computer each time ProDOS 8 starts.

Luigi Bruno
Rome, Italy

Cassette in, sound out

This machine language program will allow II-Plus and IIe owners to use their Apples as tape players (for reasons unknown to me). Just connect the recorder to the cassette connector on the computer, put in a nice cassette, press play, and run the program. Adjust the volume for best results. You will hear the tape through your Apple speaker. It works best with simple melodies. Here's the program:

```
6000: AD 60 C0   LDA $C060
6003: 30 FB      BMI $6000 (or BPL--no difference)
6005: AD 30 C0   LDA $C030 (any even number of
6008: AD 30 C0   LDA $C030       LDA $C030s)
600B: 4C 00 60   JMP $6000
```

Ron Maimon
Syracuse, N.Y.

What your program does is monitor the "state" of the cassette input ($C060—it can be "on" or "off"). In one of those states it toggles

the speaker output ($C030). Here's another program you should try:

```
6000: A0 1E     LDY #$1E   Create 150 microsecond
6002: 88        DEY          delay by counting
6003: D0 FD     BNE $6002    down Y
6005: AD 60 C0  LDA $C060  Get cassette input
6008: 45 2F     EOR $2F      compare it to LAST
600A: 10 F4     BPL $6000    if no change, restart
600C: 45 2F     EOR $2F    Input changed. Fix A
600E: 85 2F     STA $2F      store for LAST compare
6010: AD 30 C0  LDA $C030  Tickle speaker
6013: 4C 00 60  JMP $6000  Start over
```

This code segment comes from an article called "Your Apple Can Talk", by Bob Sander-Cederlof, which was published in the November 1982 ***Apple Assembly Line.*** *This code segment only toggles the speaker output when the state of the cassette input changes. In theory, it should provide a cleaner sound than your program, but I haven't actually tried either one.*

You can use a microphone as well as a cassette player as your input device. Using the rest of the program from ***Apple Assembly Line,*** *you can "record" and "playback" about 10 seconds worth of sound samples. The quality is scratchy and low-fidelity compared to what can be done with sound on the IIgs, but it's an interesting place to begin a study of computer-generated sounds.*

Option-key watchdog

In your answer to January's "Foreign Accents, cont" you mention the possibility of a desk accessory being an 'option-key watchdog' for programs that don't do it themselves. Our *MECC Key Caps* disk accessory, which you mention earlier in the same paragraph, will also function this way. It allows using option, option-shift, and control-key combinations to access any character in a font from within an application. As far as we know, it works with all applications, including *AppleWorks GS*, that doesn't use the option key for other purposes. For those that are already using the option key, you can use the *MECC Key Caps* desk accessory to type the character and then cut or copy it to the

clipboard and paste it into the application.

As you noted, *MECC Key Caps* is included with the purchase of our *Calendar Crafter* program and will be included with our other IIgs products in the future.

Paul R. Wenker
MECC Technical Services
St Paul, Minn.

65802 & No-Slot Clock

I recently put a 65802 in my IIe and found only one incompatibility problem. The software that came with the No-Slot Clock would no longer recognize the clock or even confirm that it exists.

Has anyone else confronted the problem? Is there a solution?

Frank Eddy
Ogden, Utah

Someone out there knows.

RAM chip choices

I have an Apple IIgs with an Apple memory expansion card and am considering expanding it to accomodate *AppleWorks GS* and other programs that require at least 1.25 megs of RAM.

In making a choice of which card to buy, one of the most difficult choices is whether to purchase a card that uses 256K or 1 meg chips. I'd appreciate your thoughts on which is the better choice based both on price and continued availability. I'd also appreciate your thoughts on when the best time might be to buy additional RAM given current trends.

Scott Lorigan
Elk Grove, Calif.

My opinion is that you should be buying cards that use 1-meg chips. On a per-megabyte basis, these chips are already cheaper than 256K chips, and I expect the gap to widen from now on. The 256K chips are getting harder and harder to find. The 1-meg cards are far more expandable, are cooler, use less power and should be more reliable.

The "best time" to buy is when you need them. I expect the price of 1-meg chips to start going down within a few months, but the drop will be gradual . They'll ***always*** *be cheaper tomorrow, but if you always wait till then to buy, you'll never get any.*

Pascal and 3.5 disks

I am unable to transfer/copy Pascal files from 5.25 disks to 3.5s. After copying, it gives me the message UNBOOTABLE DISK when the copy is booted.

R. Naidu
Te Kauwhata, New Zealand

Assuming you are using Apple Pascal 1.3 (earlier versions don't support 3.5 disks), first format your 3.5 with the Pascal 1.3 ***Formatter*** *utility. If you format it with Apple's System Utilities or most other formatting programs, the disk won't boot. After formatting, copy your Pascal files to the 3.5 using the Pascal* ***Filer.***

Source for Curriculum Guides

Please send more information on Apple's *Curriculum Software Guides*, which you mentioned in December's ***Open-Apple*** (page 4.85). I am particularly interested in the Foreign Language K-12 volume.

Richard Melpignano
Bellingham, Mass.

Those books are published by Apple itself and are consequently available only from Apple dealers who care enough to stock them. If your dealer doesn't, call directory assistance in a few large cities near you until you find Apple's regional sales office. Then ask them for the name of a dealer that stocks the books.

Lam is back

Here's an update to "IIgs missing CALL -144" in your August 1987 issue, page 3.53—CALL -144 is alive and well in the ROM version 01 of the IIgs. In fact, it is officially documented by Apple as "MONZ4" at the top of page 256 of *Apple IIGS Firmware Reference.*

However, the Lam routine still doesn't work if you use $D9C6 as the "return to Applesoft" address. You have recommended this address in the past ("Picking Up Applesoft," February 1985, page 1.12; "A charming difference," March 1985, page 1.23) because on classic Apples it work from within Applesoft subroutines.

On the IIgs, you have to use $D7D2 (or $D823, which jumps to $D7D2) as the return address. This means Lam is back, but not from within subroutines. The Lam routine will also crash if you run the BASIC program after setting a non-zero bank in the monitor. Prefixing the $0300 with a bank byte fixes this problem.

Here, for example, is Lam a'la Kashmarek (March 1986, page 2.16) modified so it will run on the IIgs:.

```
10 : REM Lam a'la Kashmarek/Robbins (IIgs)

100 BELL$ = CHR$ (7)
110 C$ = "00/0300:20 7B DD  20 00 E6  AA A0 00
                B1 5E 09  80 99 00  02 C8 CA
                D0 F5 A2  04 86 48  4C 70 FF
                N 03F6:00 03 N D7D2G"

200  PRINT BELL$;"Start!": REM  Standard Lam
210  FOR I = 1 TO  LEN (C$) :
        POKE 511+I, ASC(MID$(C$,I))+128 :
        NEXT : POKE 72,4 : CALL -144

220  PRINT BELL$ : REM Lam a'la Kashmarek
230  & C$
240  PRINT BELL$
250  END
```

Bill Robbins
Osaka, Japan

A2-Central™

© Copyright 1989 by
A2-Central

Most rights reserved. All *programs* published in ***A2-Central*** are public domain and may be copied and distributed without charge. Apple user groups and significant others may obtain permission to reprint *articles* from time to time by specific written request.

Written, edited, and published by
Tom Weishaar

with help from:
Dennis Doms **Sally Dwyer** **Dianne Plumberg**
Tom Vanderpool **Steve Kelly** **Joyce Hammond**
Dean Esmay

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 three 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.

A2-Central is sold in an unprotected format for your convenience. You are encouraged to make back-up archival copies or easy-to-read enlarged copies for your own use without charge. You may also copy ***A2-Central*** for distribution to others. The distribution fee is 15 cents per page per copy distributed.

WARRANTY AND LIMITATION OF LIABILITY. I warrant that most of the information in ***A2-Central*** is useful and correct, although drivel and mistakes are included from time to time, usually unintentionally. Unsatisfied subscribers may cancel their subscription at any time and receive a full refund of their last subscription payment. The unfilled portion of any paid subscription will be refunded even to satisfied subscribers upon request. MY LIABILITY FOR ERRORS AND OMISSIONS IS LIMITED TO THIS PUBLICATION'S PURCHASE PRICE. In no case shall I or my contributors be liable for any incidental or consequential damages, nor for ANY damages in excess of the fees paid by a subscriber.

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

GEnie mail: A2-CENTRAL
913-469-6502