THE APPLE II GS

The 65C816 processor brings the Apple II into the 16-bit world.

Editor's note: The following is a BYTE *product preview. It is not a review. We provide an advanced look at this new product because we feel it is significant. A complete review will follow in a subsequent issue.*

The Apple II has a curious history. It was originally designed by Steve Wozniak and Alan Baum in 1976 as a homebrew computer that squeezed maximum features out of minimum parts. Over the years, it evolved into the Apple II+, then the IIe, then the IIc. Once, Apple tried to kill it off with the Apple III (which itself died) and, later, with the Macintosh and the IIc. Despite corporate attempts to ignore it and retard its evolution, the Apple II continued to bring in the major part of Apple's income. Finally, in May 1984 Apple acknowledged the reality of the Apple II's success when it titled its day-long introduction of the Apple IIc "Apple II Forever." (Despite Apple's wishes to the contrary, the IIe continued to sell better than the non-expandable IIc—people wanted their expansion slots.) By mid-1985, though, the Apple II began to lose its sales appeal, and Apple engineers were already working on a product called, at various times, Phoenix, Columbia, Cortland, and Granny Smith: the Apple II GS.

The Apple II GS looks back to the past and forward to the future, and the machine might best be summarized by saying that it takes a giant step in both directions. Its new styling and modularity (see photo 1) foreshadows a day when Macintosh and Apple II products will use the same keyboard and 3½-inch disk drives.

SYSTEM DESCRIPTION

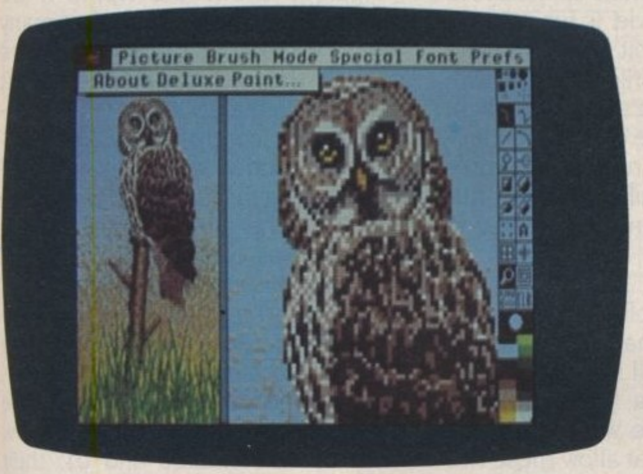
Here are the most important features of the Apple II GS:

- **Apple II compatibility:** The Apple II GS will run most Apple II software and expansion cards. It can run at normal Apple II speed or at a higher rate that makes most software run two to three times faster. [*Editor's note: In this article, "Apple II" refers to the traditional Apple II computer as defined by the Apple II, II+, IIe, and IIc.*] The Apple II GS composite video signal has been corrected so that it will be recorded correctly by a videotape recorder. Apple IIe owners can upgrade to complete II GS compatibility by replacing the motherboard and back/bottom plate with a II GS retrofit kit.
- **A 16-bit, 6502-compatible processor:** With a 16-bit address bus and 8 "bank address" lines, the Western Design Center's W65C816 can address 256 banks of 64K bytes each, for a total of 16 megabytes. It can also go into a 6502 mode, where it emulates the 65C02A used in the Apple IIe and IIc. The processor's accumulator, stack pointer, and all its registers are 16 bits wide, and its instruction set includes 11 new addressing modes.
- **Greatly expanded memory capacity:** The machine's architecture reserves space for 8 megabytes of user RAM and 1 megabyte of system ROM. It comes with 256K bytes of RAM, 128K bytes of system ROM, and 64K bytes of dedicated sound-waveform memory, but you will have to wait for new programs to use most of the memory above the first 128K bytes. Apple currently has plans for 1- and 4-megabyte expansion cards, although an 8-megabyte card is possible.
- **New graphics capabilities:** The Apple II GS adds two "super hi-res" graphics modes: 200 by 320 pixels with a 16-color palette and 200 by 640 pixels with a 4-color palette; the colors come from a color set of 4096. The machine can use up to 16 palettes per screen and change palettes and resolution on a line-by-line basis. Programmers can use two experimental modes: a 640- by 200-pixel, 16-color (with restrictions) palette mode, and a high-speed "fill mode" variation of the 200 by 320, 16-color mode.
- **New sound capabilities:** The 32-voice Ensoniq Digital Oscillator Chip (DOC), used in the Ensoniq Mirage sampled-sound music synthesizer, and system firmware can drive the

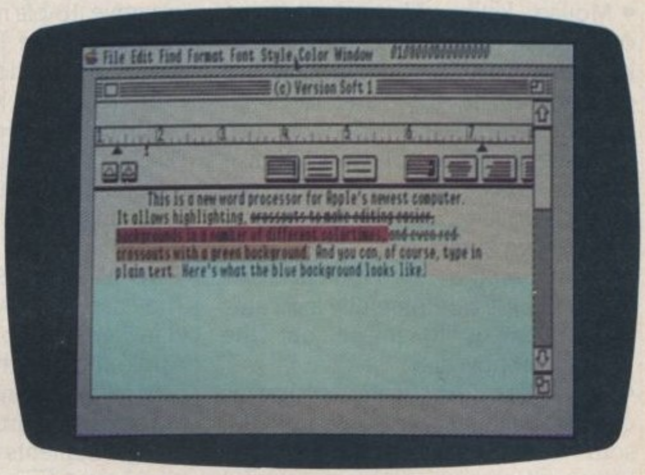
(continued)

Gregg Williams is a senior technical editor at BYTE; *he bought his first Apple II+ in 1980 and now owns an Apple IIe. Richard Grehan, who has owned an Apple II+ since 1985, is a technical editor at* BYTE. *They can be reached at One Phoenix Mill Lane, Peterborough, NH 03458.*

BY GREGG WILLIAMS AND RICHARD GREHAN



Deluxe Paint drawing program. (Photo by Electronic Arts.)



This is a new word processor for Apple's newest computer. It allows highlighting, ~~crossouts to make editing easier,~~ backgrounds in a number of different colors, ~~and even red crossouts with a green background.~~ And you can, of course, type in plain text. Here's what the blue background looks like.

Writer's Choice Elite word processor. (Photo by Activision Inc.)

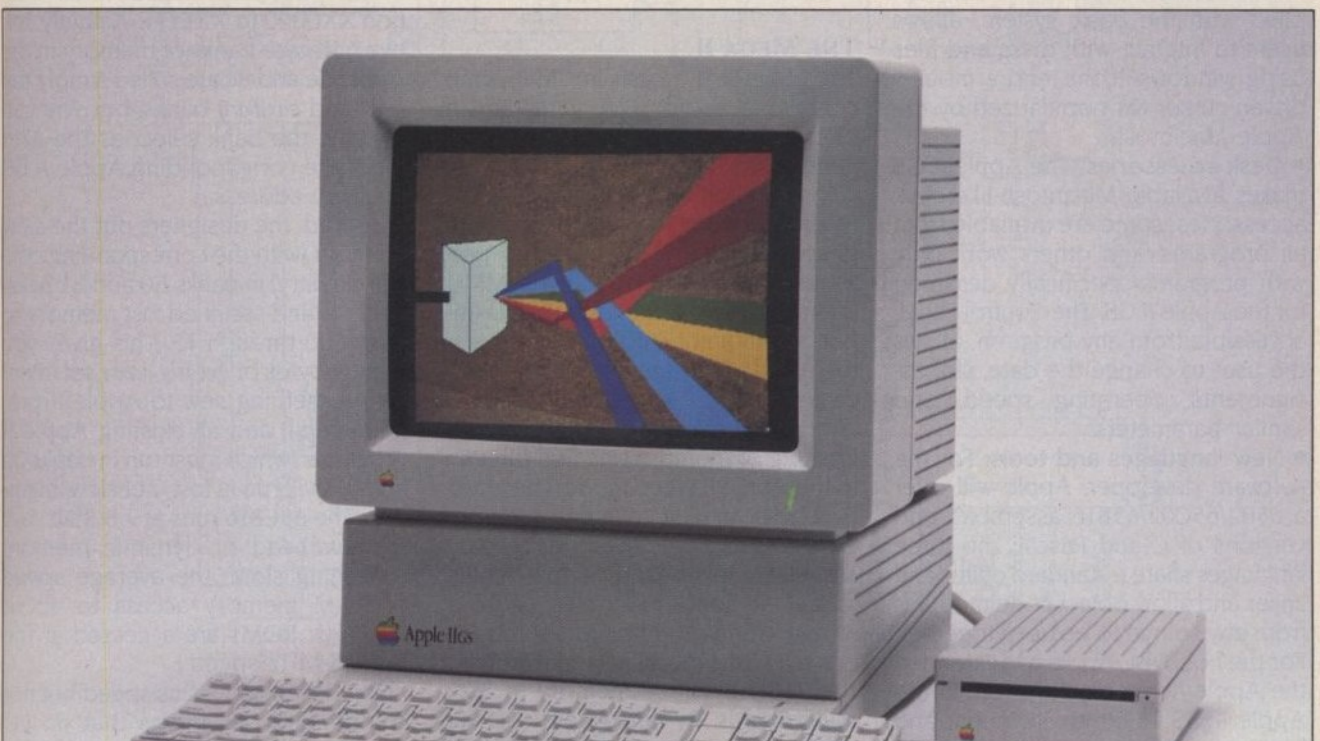


Photo 1: *The Apple* II GS.

SCREEN DRAWING BY MITCHELL RICE

PHOTOGRAPHY BY PAUL AVIS

chip to produce up to 15 musical "instruments."

• **Mouse, keyboard, and disks:** A one-button mouse and a detachable keyboard with keypad are standard equipment. The Apple II GS does not have an internal disk drive, but you can daisy-chain up to two 800K-byte 3½-inch drives and two 140K-byte 5¼-inch drives to the disk drive port on the rear panel. The system software will come on 3½-inch disks, which silently but forcefully indicates Apple's intent to phase out the 5¼-inch floppy disk.

• **The Toolbox:** Application programs can use built-in code (some in ROM, some in RAM) to provide a mouse-driven desktop environment and orderly use of system resources.

• **The Finder:** Finder software, supplied with the basic system, allows users to interact with disks and files using windows, icons, and a mouse-driven cursor (as popularized by the Apple Macintosh).

• **Desk accessories:** The Apple II GS makes available Macintosh-like desk accessories; some are available from all programs, and others work only with programs specifically designed for the Apple II GS. The Control Panel, accessible from any program, allows the user to change the date, slot assignments, operating speed, and similar parameters.

• **New languages and tools:** For the software developer, Apple will offer a 6502/65C02/65816 assembler and versions of C and Pascal; the three languages share a standard editor and linker and allow object code modules from any source to be used together. For the hobbyist, Apple has extended the Apple IIe monitor to work in the Apple II GS 16-bit environment and has added new functions to it.

• **No enhanced, built-in language:** Like the Macintosh (and unlike most other computers), the Apple II GS contains no built-in language (such as Microsoft BASIC) that interacts with the machine's new features. Applesoft BASIC is available in system ROM, but it has no way of directly interacting with the new Apple II GS features.

• **A new 16-bit operating system:** ProDOS 16 extends Apple's ProDOS (which runs on the Apple II+, IIe, and IIc) to be the standard Apple II GS operating system; it runs on the 65816 in native 16-bit mode, is functionally similar to the 8-bit ProDOS, and shares an identical file structure with ProDOS. Apple has also made slight modifications to the 6502-based ProDOS so that it will run on the Apple II GS's Apple II emulation mode; this operating system is named ProDOS 8.

Two Machines

The case of an Apple II GS contains, in a sense, two machines: the full Apple II GS, with all its memory and new features, and a 128K Apple IIe. Much of this article will explain the design elements that allow these two "machines" to exist together. You may want to refer to figure 1, which is a block diagram of the Apple II GS.

The Mega II

The Mega II is a custom CMOS chip containing about 3000 gates and a 2K-byte by 8 ROM (for the character generator). It replaces the following chips from the Apple IIe and IIc: character generator ROMs for eight languages, several TTL chips that perform logic functions, and the MMU (memory management unit), IOU (input/output unit), TMG (timing generator), and GLU (general logic unit) custom chips.

In previous Apple II designs, the refreshing of memory was tied directly to the Apple II video mode. The Mega II includes an 8-bit counter for refreshing the 128K bytes of (slow) memory associated with the Apple IIe/IIc model; it does five cycles of RAM refresh during the horizontal retrace of each video scan line and refreshes the 128K bytes of memory in 3.25 milliseconds. By taking care of RAM refresh, the Mega II chip opens the Apple II design to new video modes that were impossible before.

Speeding Up the II GS

The Apple II GS designers had many conflicting goals. They wanted to make a machine that runs as much existing Apple II software as possible and to make it run software (both old and new) faster than on an Apple II. In order to accomplish this, they changed the memory map and employed a technique called *shadowing*.

Figure 2 shows the Apple II GS memory map. Remember that many memory areas in the Apple II are special; the memory-mapped "soft switches" in the C000-C0FF hexadecimal region control many key functions, interaction with the peripheral cards occurs through locations in the C100-CFFF hexadecimal region, and several areas of memory determine what graphics and/or text are shown on the video display. Many of these areas are limited by the original Apple II design to being accessed at 1 MHz. A straightforward expansion of the Apple II design would put the slow memory in banks 00 and 01, their location in the Apple IIe and IIc. (A bank is defined as the 64K-byte address space from hexadecimal location XX0000 to XXFFFF. Actually, the two 64K-byte banks of memory in the Apple IIe and IIc are called simply the *main* and *auxiliary* banks, but you can imagine the bank select as the 17th bit of the corresponding Apple II GS memory address.)

Instead, the designers put the slow memory (with the corresponding control circuitry) in banks E0 and E1 hexadecimal and assigned fast memory to banks 00 through 7F. This gives you 8 megabytes of *linearly addressed* memory (something new to Apple II programmers), and all existing Apple II programs (which must run in banks 00 and 01) will run in fast, not slow, memory. (The 65C816 runs at 2.8 MHz, but the overhead of dynamic memory refreshing slows the average speed of RAM memory access to about 2.5 MHz; ROMs are accessed at the full 2.8-MHz speed.)

This scheme gives us speed but not compatibility. Programs that do I/O using the peripheral slots and video display write to addresses in banks 00 and 01, but the hardware they need to interact with is tied to the slow memory in banks E0 and E1 hexadecimal. How can we get this scheme to work?

The answer is shadowing. The Apple II GS engineers designed the Fast Processor Interface (FPI) custom chip to monitor any attempt to write to the area to be shadowed (in bank

(continued)

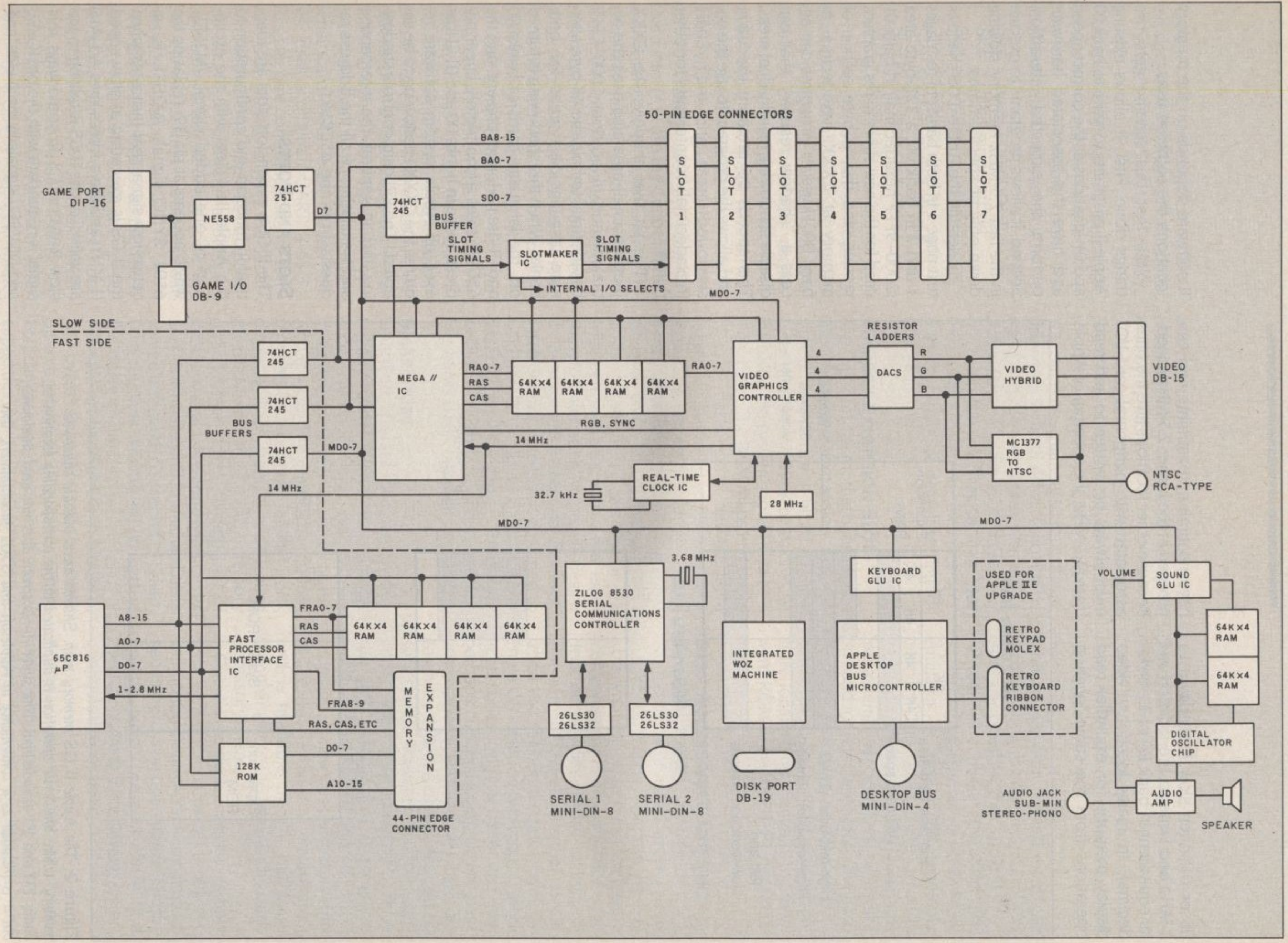


Figure 1: *A block diagram of the Apple II GS.*

00 or 01), then slow itself down to 1 MHz and write to the location *and* its equivalent in bank E0 or E1 hexadecimal. In many locations (video display memory, for example) read operations from the same location have no timing constraints and can proceed at the higher 2.5-MHz speed. Because other locations (like the ones associated with peripheral card I/O) must always be written to and read from at 1 MHz, the speedup of Apple II software depends on the memory locations the software uses.

Note that this scheme gets two things done. First, it allows existing Apple II programs to write to bank 00 and 01 locations at the correct speed and have the associated hardware perform the expected interaction. Second, it allows programs to execute in the fast 2.5-MHz memory, slowing down only at specific times.

By default, the Apple II GS shadows text page 1 in both banks, hi-res pages 1 and 2 in bank 00, a 32K-byte area (2000–9FFF hexadecimal) in bank 01 used for the new super hi-res graphics modes, and the 4K-byte section of memory at CXXX hexadecimal in bank 00. However, programs can access a "shadow register" that can disable shadowing in individual areas. This speeds up program execution and allows the program to use the unshadowed areas and their E0/E1 (hexadecimal) counterparts for other things.

(The bit that shadows the CXXX [hexadecimal] area also uses the actual memory in locations C000–CFFF hexadecimal to hold the 4K-byte alternate language-card areas for both banks. When this shadowing is turned off, the language cards of banks 00 and 01 are no longer present, and the 65C816 sees a completely linear address space in banks 00 and 01. [However, all Apple system software requires the CXXX shadowing to be enabled.] The FPI chip controls shadowing and, in general, the intercepting and translating of all the address requests from the 65C816.)

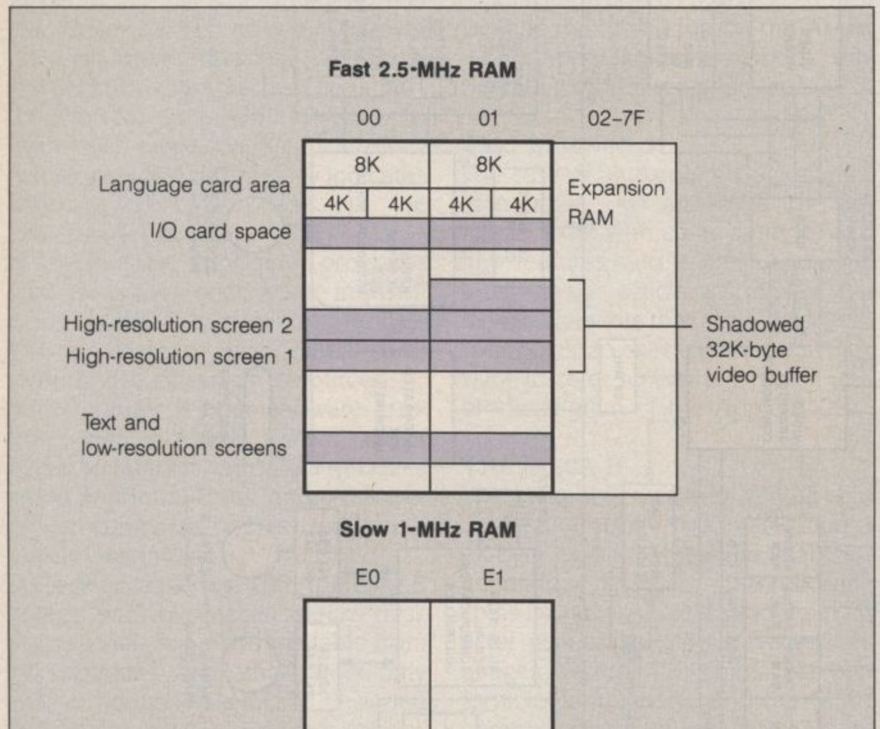


Figure 2: *The Apple II GS memory map. Shaded areas indicate shadowed memory, where any write operation is duplicated to the corresponding location in banks E0 and E1 hexadecimal. The super hi-res screen does not need shadowing, but it is included as a convenience to Applesoft and 6502 programmers, who can only work with banks 00 and 01. Memory areas are not drawn to scale.*

SLOTS AND PORTS

The II GS's expansion slots are identical in function and configuration to the Apple IIe's slots with the exception of one added signal, /M2SEL, which appears at pin 39, replacing the 6502 SYNC signal. /M2SEL is an active-low signal that indicates when the II GS is executing at slow speed (1 MHz) and the address lines A0–A15 are valid (i.e., the II GS is talking to the slow RAM or I/O). In some ways, this signal is redundant with the old IOSEL and DEVSEL signals in the II+/IIe, and boards that use these signals should have no problem operating in an

Apple II GS. Control-signal generation and clock-signal buffering on the ports are handled by the SlotMaker custom IC.

Associated with each I/O expansion slot are built-in circuits and firmware that form an "invisible" port. The default settings for the seven slots are

Slot 1	serial printer port
Slot 2	serial modem port
Slot 3	80-column display
Slot 4	mouse
Slot 5	3½-inch disk drives
Slot 6	5¼-inch disk drives
Slot 7	AppleTalk

It is as though you had an Apple IIe with boards for all the above devices already plugged in. Unfortunately, this wealth of built-in interfaces carries with it some restrictions, the most severe being that if you have an expansion board you want to run in your II GS, you must give up the built-in port of whatever slot you plug the board into. (You use the Control Panel to choose whether a slot is using its associated default port or a plug-in board; this information is retained in battery-backup RAM.) The serial ports of the Apple II GS appear on the rear of the cabinet in the form of a pair of 8-pin mini-DIN connectors. They are pin- and signal-compatible with the serial connectors on the back of the Macintosh Plus—in fact, the USART in the II GS is the same as the Macintosh's: a Zilog 8530 serial communications chip. If you access the serial ports through the firmware, they "appear" identical to the Apple Super Serial Card (SSC), even though the SSC uses a different UART, a 6551. This has dire consequences for software that talks directly to the serial-port hardware on Apple IIs equipped with SSCs. Such software—and this includes practically all of the commercial communication packages—will certainly fail on the II GS.

Programs that bypassed the SSC's firmware did so to break the speed limitations of the firmware's non-interrupt-driven, unbuffered I/O routines, which were virtually useless for dependable communications at 1200 baud. The II GS's serial port firmware solves these troubles: It is interrupt-driven, and each serial port has input and output buffers that default to 2K bytes each but can be set to up to 64K bytes each.

The Apple II GS's built-in disk port is a 19-pin miniature D-type connector in the middle rear of the machine. You can daisy-chain up to four drives, up to two 3½-inch drives followed by up to two 5¼-inch drives. Owners of Apple IIs who might want to use their drives on the II GS can simply plug their Disk II controller into slot 5 or 6 and override the default setting for that slot.

MEMORY EXPANSION SLOT

The Apple II GS motherboard has a special memory-expansion slot designed for a card with up to 8 megabytes of RAM and 896K bytes of ROM (bringing the system's total ROM to 1 megabyte). The RAM maps into banks 02 to 7F hexadecimal, and the ROM maps into banks F0 to FD hexadecimal. It is easiest to design 1- and 4-megabyte RAM cards (using 256K-bit by 1 chips and 1 megabit by 1 chips, respectively), but the II GS engineers said that, with a few extra chips for interfacing, you could design an 8-megabyte RAM card; however, since the machine is not designed to hold user RAM above bank 7F hexadecimal, an 8-megabyte RAM card would be unable to access the top two banks (128K bytes) of its memory.

SMARTPORT

SmartPort is a set of assembly language routines (held in firmware) for accessing block and (as yet undesigned) character I/O devices on the Apple II GS. The SmartPort routines provide support for 3½-inch disk drives, a RAM memory disk (called the /RAM device), or a ROM memory disk (5½-inch drives, though part of the daisy chain, are controlled by the Disk II firmware, and future hard disks can be designed to respond to SmartPort routines without being part of the daisy chain).

SmartPort handles I/O in blocks of 512 bytes; since the routines permit up to a 4-byte block number, SmartPort can manage devices with storage capacities up to 2,199,023,255,552 bytes. SmartPort's basic functions include get device status, reset a device, format a device, read a block from a device, write a block, and send control information.

Any I/O expansion card that adheres to SmartPort conventions will have signature bytes at specific locations in its on-board ROM. The II GS's firmware will hunt for and recognize these at boot-up time, just as ProDOS currently does on the Apple II.

As its name implies, the ROM disk is the equivalent of a RAM disk emulator in nonvolatile read-only memory. This could come in handy for keeping frequently used programs like assemblers, compilers, or the like on hand for rapid execution. The II GS memory space has eight 64K-byte banks set aside for ROM disk expansion, located just beneath the firmware ROM in banks F0–F7 hexadecimal.

DESK ACCESSORIES AND THE CONTROL PANEL

You can think of a desk accessory as a mini-application that can be run from within another program. Macintosh owners are already familiar with desk accessories—those utility programs from the menu bar that appear when you click on the apple symbol. The II GS supports two types of desk accessories (with a tip of the hat, perhaps, to Coca-Cola): classic desk accessories (CDA) and new desk accessories (NDA). A classic desk accessory can be activated only by a keypress. Classic desk accessories can be run with older Apple II programs (such as Appleworks) and new II GS programs. A new desk accessory runs in the II GS's desktop environment and is available from a pull-down menu similar to the Macintosh's desk-accessory menu.

One classic desk accessory is built into the II GS: the Control Panel. You call up the Control Panel by simultaneously pressing open-apple-Control-Escape, which presents you with a menu containing the following system configuration options:

- **Display** selects color or monochrome monitor, display width, and colors for text, background, and border.

(continued)

• **Sound** displays two "slider switches" used to adjust the II GS speaker's pitch and volume.
• **Speed** selects 1.0-MHz or 2.8-MHz ("normal" or "fast") operation of the 65C816 CPU.
• **Clock** sets the II GS system clock time and date.
• **Options** alters parameters of the II GS's keyboard: keyboard layout, keyboard buffering on or off, repeat speed and delay, and others.
• **Slots** lets you indicate for each of the II GS's seven I/O slots whether the slot is running an "invisible" port or a plug-in board.

Other selections from the Control Panel let you set parameters for the serial ports and enable a RAM disk. You can use the Control Panel from within any program; we even used it in the middle of a disk access with no adverse effects.

VIDEO MODES AND THE VGC

Because the Apple II GS emulates the Apple II, it contains all the text and graphics modes of the Apple II: 24 by 40 text, 24 by 80 text, 48 by 40 low-resolution and 48 by 80 medium-resolution graphics with 16 predefined colors, 192 by 140 hi-res graphics with 6 predefined colors, and 192 by 140 double hi-res graphics with 16 predefined colors and 192 by 560 monochrome graphics. (Apple claims double the above numbers for hi-res resolutions in the horizontal directions, but the numbers here more accurately reflect the true nature of hi-res graphics because of the color limitations between adjacent pixels.)

As stated earlier, removing the banks E0 and E1 (hexadecimal) dynamic RAM refreshing from the video display circuitry makes new video modes possible. A new custom chip, the Video Graphics Controller (VGC), implements both old and new video modes as well as unrelated support functions for the built-in clock chip, the disk drives, the interrupt system, and built-in chip and board testing routines. The VGC enhances current text modes by allowing the user to choose from the Control Panel the color (or gray scale value) of the text, its background, and the border outside the active text/graphics area. These modes are available only when using an RGB color or monochrome monitor.

SUPER HI-RES GRAPHICS

The new modes are called "super hi-res." Actually, there are three modes that can be used in four ways; two of them are pretty straightforward and useful, while the other two are more experimental.

Associated with the super hi-res modes is a 32K-byte chunk of memory in bank E1 ranging from addresses 2000 to 9FFF (assume that the addresses in this section are hexadecimal and the quantities are decimal). The pixel map occupies the range from 2000 to 9CFF, an important set of pointers occupies locations 9D00 through 9DFF, and color palette information fills the remainder of the area, from 9E00 to 9FFF. (To get into these modes, write C1 hexadecimal to location C029, and write 41 hexadecimal into it to restore the Apple II modes.)

The pixel map contains exactly 32,000 bytes arranged as 200 rows of 160 bytes each. Apple II programmers, who have always struggled with a convoluted pixel-to-memory mapping scheme, will be surprised by the fact that the super hi-res modes are completely linear, with a row-first stream of pixels corresponding to an unbroken, increasing progression of memory addresses. In other words, the first pixel on the first line uses the high bits of location 2000, while the same pixel in the second line uses location 20A0 (160 bytes later), and so on.

There are two super hi-res modes. Both have 200 lines per screen, but one has 320 pixels per line (see photo 2), while the other has 640 pixels per line. Since each line is represented by 160 bytes, each pixel has 4 bits of memory in the 320 mode and 2 bits in the 640 mode (see figure 3). This scheme gives you 16 colors in the 200 by 320 mode and 4 colors in the 200 by 640 mode, with no restrictions on the color of adjacent pixels (a prob-

(continued)

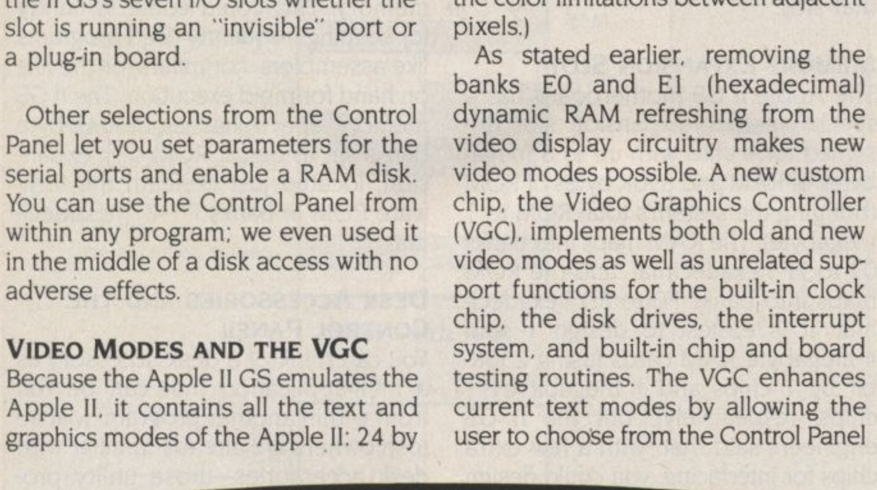


Photo 2: *Apple super hi-res graphics (200 by 320), fill mode, and line-addressable mode switching. The top (green) portion of this photo is in fill mode, while the bottom is not. Note that the colored outline of the cube determines what color a given face will be painted in fill mode. Note also the green line running down the left side the screen; this line makes the background color of the screen green.* [Courtesy of Apple Computer.]

lem that complicates the Apple II hi-res modes).

With the old Apple II hi-res modes, the electronic characteristics of both the Apple II and its video display determined the colors that were available; for example, the hi-res mode gave you the colors violet, blue, green, orange, black, and white. The Apple II GS, through the VGC chip, gives you more control over the colors in your graphic display. The super hi-res mode lets you choose your palette of 16 colors from a color set of 4096.

But which of these 16 colors are used in the 640 by 200, 4-color mode? The answer is all of them, the details of which lead us to one of the experimental super hi-res modes. The two bits of a pixel in this mode can have four values, so they are used in this mode to choose from 4 colors in the 16-color palette. Which 4 colors? Apple II programmers will recognize the answer as yet another convoluted video mode in the Apple II tradition: The 4 colors available for a pixel depend on its position within a byte (see figure 3b).

PALETTES AND POINTERS

Actually, the Apple II GS defines a 512-byte area starting at location 9E00 hexadecimal; this area contains 16 color palettes of 32 bytes each, numbered from 0 to F hexadecimal. Each color in a palette is defined in 2 bytes, using 4 bits each to describe the red, green, and blue components of the color. The first byte contains the values for green (bits 7-4) and blue (bits 3-0); the second byte contains the red value (bits 3-0), with the remaining bits set to zeros.

Why are there 16 palettes? Because each scan line can use any of them in any order. This brings the total number of colors that can appear onscreen to 16 × 16 = 256 colors. Expect to see some uncharacteristic graphics as soon as programmers learn their way around the machine.

The final surprise of the super hi-res graphics modes comes from the pointer area. The pointer byte at location 9D00 hexadecimal corresponds to the top line of the video display, with each successive scan line getting the next byte: 9D01, 9D02, . . . , etc. This byte is read and interpreted during the horizontal retrace of the previous video line.

Within each pointer byte, bits 3 through 0 determine which of the 16 color palettes is to be used. Bit 4 is not used and should be set to 0. Bit 6 does nothing if set to 0. If it and an interrupt register at address C023 hexadecimal are both set to 1, the VGC generates an interrupt at the beginning of the line; this will allow the advanced programmer to wring extra performance out of the super hi-res screen by altering palette values (or making other useful changes) "on the fly"—that is, while the machine is drawing the video display. Bit 7 determines the resolution: 0 for 320 pixels, 1 for 640.

This leaves bit 5, which does nothing if set to 0 but which activates the final, experimental super hi-res mode, called *fill mode* (see photo 2). In fill mode (which works in 200 by 320 resolution only), you have access to 15 colors (numbers 1 through F, hexadecimal). A pixel value of 0 means that its color is the same as the last nonzero pixel to the left. In other words, pixels with the values

3 0 0 0 2 0 0 0 0 0 0 0 9 0 0 0

would appear as colors

3 3 3 3 2 2 2 2 2 2 2 2 9 9 9 9

and you could change the large area painted with color 2 to, say, color 5 by changing one pixel (the fifth one) from a 2 to a 5. (Note that the first pixel in a line must always be nonzero.) This mode will be good for drawing large areas and changing their colors very quickly.

SOUND

The heart of the II GS's sound system is the Digital Oscillator Chip manu-

(continued)

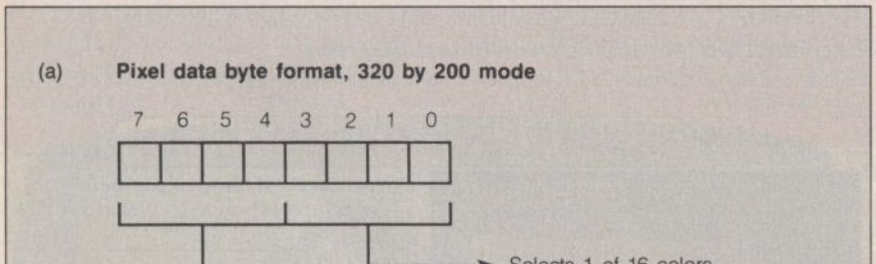

(a)
Pixel data byte format, 320 by 200 mode
7 6 5 4 3 2 1 0
Selects 1 of 16 colors for 2nd pixel
Selects 1 of 16 colors for 1st pixel
(b)
Pixel data byte format, 640 by 200 mode
7 6 5 4 3 2 1 0
Selects from colors 4–7 for 4th pixel
Selects from colors 0–3 for 3rd pixel
Selects from colors C–F for 2nd pixel (hexadecimal)
Selects from colors 8–B for 1st pixel (hexadecimal)

Figure 3: *(a) Pixel decoding in the* 320 *by* 200 *and (b)* 640 *by* 200 *super hi-res modes.*

factured by Ensoniq and used in the Mirage digital synthesizer. The DOC is attached to its own personal bank of 64K memory into which programs store wave tables that the DOC uses to generate sound. This memory is accessible only through special registers in the Sound General Logic Unit, a custom chip that acts as an interface between the DOC and its memory and thus allows sound generation to proceed independent of other processing in the Apple II GS. Additionally, registers within the Sound GLU chip regulate the gain of the II GS's audio amplifier, providing control of the speaker's volume.

The Ensoniq DOC contains 32 oscillators that the II GS firmware operates in pairs to generate a tone. Since one of the oscillators is used by the system to generate a time-slice interrupt for the DOC, the Apple II GS can produce up to 15 independent tones simultaneously.

A wave table is a series of bytes in the DOC's memory such that each byte represents the instantaneous value of the amplitude of the sound's output waveform. An oscillator on the DOC will step through this table fetching bytes and passing them to an on-chip digital-to-analog converter that produces the analog waveform that, after filtering, goes to the speaker. This technique allows generation of a theoretically limitless range of sounds, bounded only by the amount of memory available. The limit of 64K dedicated memory is no impediment, since the II GS allows you to fill one portion of a wave table while the DOC is fetching information out of another.

The II GS passes the unamplified monophonic signal of the sound system's output to a mini stereo phone jack on the back panel of the machine. This output is capable of driving a pair of Walkman-style headphones or, with the proper adapter cable, the input of a stereo amplifier. The designers of the II GS have also provided a connector on the motherboard that gives direct access to several useful DOC signals, including an unfiltered audio output, channel selection logic signals (can be used to implement eight independent audio channels), and an input to the DOC's analog-to-digital converter (for a sound sampler/digitizer).

The Apple II GS Toolbox

In the past, the Apple II could be almost completely described by its hardware features. The Apple II GS ends this tradition with its inclusion of significant amounts of system software in both ROM and RAM meant to be available to all programs. It is not accidental that these routines are similar in name and function to those in the Macintosh computer. The Mac toolbox is an elegant, powerful system proven to work and improved by two years of intensive use.

The Apple II GS toolbox implements the most useful Macintosh toolbox functions, though sometimes it does so in a different way; the Memory Manager, for instance, works quite differently from its Macintosh counterpart because of the way the Apple II GS's memory is divided into 64K-byte banks. However, the Apple II GS doesn't duplicate all of the Macintosh toolbox.

The code in the Apple II GS toolbox is divided into *tool sets*, and the individual routines are called *tool calls*. The tool sets that are in ROM are the Tool Locator, Memory Manager, SANE (Standard Apple Numerics Environment) Numerics, Desk Accessory Manager, Event Manager, Sound Manager, Integer Math Tools, Text Screen Tools, Scheduler, and Miscellaneous Tools. QuickDraw II is divided between ROM and RAM.

The remaining tool sets are stored on disk and loaded into RAM by the application that needs them. Once in memory, they are indistinguishable from tools stored in ROM. They are the Menu Manager, Window Manager, Control Manager, Line Editor, Dialog Manager, Scrap Manager, and Print Manager.

Tool Set Structure

The Apple II GS tool set has no fixed routine entry points and only four fixed addresses associated with its toolkits, yet any program can execute any toolbox routine in RAM or ROM, even if a routine is changed or moved to a different location after the program is written. Both tool sets and tool calls are numbered (starting with 1), and any tool call can be executed by the following assembly language sequence:

```
push  inp1
ldx   #CallID
jsl   Dispatch
```

This pushes any input onto the stack, loads the 16-bit X register with a call ID constant that has the tool call number in the high byte and the tool set number in the low byte, and does a subroutine jump to a fixed entry point. A high-level language would compile a normal procedure call as a series of 0 or more push instructions, followed by a jump to a different location that performs the above function while handling an extra 3-byte return value on the stack. The Apple II GS designers estimate that this type of call has an overhead of about 118 microseconds. Parameters can be passed in several ways, based on the needs of the individual routine: on the stack, in a known block of memory, or in the A, X, and Y registers.

To increase the usability and extensibility of the II GS, its designers provided an identical but parallel structure that allows programmers to build and use their own tool sets without "borrowing" tool set numbers that Apple may later use. The only difference between the two is a different entry point, "UDispatch" instead of "Dispatch."

Toolkit Memory Usage

Many tool calls need their own memory—sometimes page zero locations to speed up their execution, sometimes other memory for passing parameters or sharing or storing data. The Apple II GS designers resolved the conflicting memory needs of many different tool calls by regulating memory usage as follows: The program using the tool sets will itself allocate page zero memory, and tool sets will allocate the other memory they need by asking for it through the Memory Manager. They can then point to it with the WAPT (Work Area Pointer Table) entry reserved for that tool set, and their tool calls will always

(continued)

be able to access that memory in whatever way they wish.

QUICKDRAW II

QuickDraw II deserves mention because of its importance for desktop-based Apple II GS software. It is a tool set, partly in ROM and partly in RAM, that provides a standard set of useful graphics routines for drawing window/menu-oriented screens. Wherever possible and appropriate, it attempts to work equivalently to a subset of Macintosh QuickDraw routines. The pre-release documentation lists 146 QuickDraw II tool calls, of which 114 are listed as being the same as their Macintosh equivalents, 22 are listed as being similar, and 10 are entirely different or absent. The degree of consistency between QuickDraw and QuickDraw II will be very important to Macintosh software developers attempting to convert their software to the Apple II GS.

PRODOS

Apple has crowned ProDOS *the* operating system for the Apple II series of computers, and the company will be guiding ProDOS along a carefully controlled development path that proceeds as follows:

• ProDOS 1.1.1 will continue to be supported for the Apple IIe and IIc computers and many ProDOS 1.1.1 programs will run on the II GS.
• ProDOS 8, an altered version of ProDOS 1.1.1, will become the standard 8-bit operating system for the Apple II. It will work on the IIe, IIc, and II GS.
• ProDOS 16 will be the 16-bit operating system used for Apple II GS software. Version 1.0, supplied with the machine at its introduction, is built on a ProDOS 16 framework but is implemented by a ProDOS 8 core surrounded by a shell handling ProDOS 16–style calls. ProDOS 16 version 2.0 will be released in the first quarter of 1987.

THE FINDER

We did not see the Finder working when we saw the Apple II GS, but its preliminary documentation describes it as "a combination Program Selector/Disk Utility for managing documents and directing traffic between the user and storage devices." It seems to be a pretty faithful imitation of the Macintosh desktop interface, with several exceptions.

First, the "Special" menu has two new items—"Check Drives" and "Format." The first causes the Finder to update its knowledge of what disk is in each drive (remember that, in an Apple II system, you can change the floppy disk in a drive without the computer knowing what you've done). The second will eventually allow you to format a disk in either ProDOS, Apple Pascal, DOS 3.3, or Apple CP/M formats; the initial release, however, will only format disks for ProDOS.

Second, the Finder will interact most fully with ProDOS disks and programs. Since only the ProDOS operating system has subdirectories, only ProDOS disks will have folders in their windows. When you exit a ProDOS program, it will return you to the Finder.

Third, the Finder does not support custom file icons. Each icon will have a shape determined by its file type.

Finally, the Finder will support rudimentary printing of text files.

A future version of the Finder will probably add the Macintosh MFS (old) and HFS (hierarchical) disk formats, Apple Pascal 1.3, and Apple CP/M to the file types supported.

APPLE DESKTOP BUS

The Apple Desktop Bus (ADB) is used for the generalized connection of the computer with up to 16 input devices daisy-chained to a single connector on the back panel; it currently supports multiple keyboards (for educational and other programs) and a mouse (ending the daisy chain), but the design can accommodate other kinds of devices. Devices are connected through a shielded 3-conductor cable using mini-DIN-4 connectors.

The ADB is controlled by a dedicated 8-bit processor called the ADB microcontroller (abbreviated here as ADBM); in addition, the mouse and keyboard are controlled by custom microcontrollers that interact with the ADBM. The ADBM and the intelligent devices "talk" on a bus where only the ADBM can issue commands; the devices reply as appropriate with data or requests for service.

In general, the ADBM handles low-level interaction with the keyboard, mouse, and other input devices, thus freeing the 65C816 processor from having to handle such tasks.

APPLETALK

Unlike any other Apple Computer product, the Apple II GS includes built-in AppleTalk code in RAM and ROM. Through the Control Panel, you can configure slot 7 as AppleTalk; the II GS then uses one of the two serial ports as its AppleTalk port.

The II GS implements the bottom two (of seven) levels of AppleTalk protocol: Link Access Protocol (LAP) and Datagram Delivery Protocol (DDP). It also implements enough of the next

(continued)

EXTENDED APPLE II GS COVERAGE ON BIX

BYTE magazine has a lead time of several months, but the lead time on BIX (BYTE Information Exchange) is measured by how fast we can type. By the time you read this, we will have active a special "Apple II GS event" on BIX that will include further technical details, exact prices, BIX users giving their impressions of the machine as it comes out of the box, and more. Much of this information will be excerpted in the Best of BIX section of BYTE as soon as our lead time allows.

For more details about the special Apple II GS coverage in BIX, log on to BIX, type join apple, and then join the first topic with 'GS' somewhere in its name.

two levels, Name Binding Protocol (NBP) and AppleTalk Transaction Protocol (ATP), to boot the II GS from a remote file server (thus allowing it to be used in a network environment without its own disk drive).

PRICING

The price for the Apple II GS had not been set at the time of this writing, but we expect the price for a starter system with one 3½-inch disk drive and a monochrome monitor to be in the $1400 to $1600 range.

COMPATIBILITY

For many users, especially current owners, software and hardware compatibility will be the make-or-break factor in their decision to buy an Apple II GS. The II GS engineers did an incredible job of designing a new, more powerful computer that is largely compatible with the existing body of Apple II hardware and software. One engineer estimated the II GS's hardware compatibility at "about 80 percent" and its software compatibility at "95 to 99 percent."

Complete software compatibility is impossible, largely because of the completely unregulated way the Apple II has been programmed in the last 10 years. People wrote code that jumped into the middle of ROM routines, used machine language op codes that were unimplemented in the 6502 (but that are in the 65C816), and implemented countless copy-protection schemes, many of which depended on particular hardware details that were replaced in later Apple II designs.

The final verdict must wait until we get to test a production-line machine, but we tested several Apple II game and business programs and found two that fail trying to execute formerly unimplemented op codes (THE Spreadsheet and Serpentine) and one (HomeWord running under ProDOS 1.1.1) that doesn't work because the 65C816 does not completely emulate the way the 6502 wraps an X-register address from FFFF hexadecimal to 0000 (it wraps to 10000 hexadecimal).

Most peripheral cards that do not implement "phantom slots" (where a multifunction card appears to be several cards in different slots) will work, but some cards won't; we were told, for example, that the Mountain Computer Music Card set won't work because of the way it uses interrupts.

As with previous enhancements to the Apple II line, such differences cause problems for the first year or so, then they fade from consciousness as companies revise their products and users find patches or workaround measures for products that don't work. In general, the more recent your Apple II software or hardware, the more likely it is to run properly.

CAVEATS

We wrote this product preview in July 1986, after two days with the Apple II GS engineering staff, much study of seven volumes of developers' technical documentation, and subsequent telephone conversations with the engineers. When we saw the Apple II GS, the firmware was about to be "frozen," and the machine itself was in "final preproduction"; only minor changes are likely to be made at this point. We did not get to see the Finder software, but we had several hours of hands-on experience and ran several impressive sound and graphics demos.

(We wish to thank Rob Moore, Harvey Lehtman, and many other Apple people for their help.)

CONCLUSIONS

What do you say about such innovative energy that has been directed primarily toward preserving a hardware design that is 10 years old? The Apple II GS designers' achievements are remarkable, but the burden of the classic Apple II architecture, now as venerable (and outdated) as COBOL and batch processing, may have weighted them down and denied them any technological leaps beyond an exercise in miniaturization. Also, the 65C816 may prove to be an IC of mixed blessings: While it does provide a means of supporting the 6502 within a processor that also operates in a 16-bit mode, to programmers it represents yet another instruction set that has to be learned and whose oddities will have to be dealt with.

The Apple II GS affirms several trends in microcomputer design that we should not ignore: improved graphics and sound, larger processor and memory capacity, and the use of a mouse and a desktop/icon/windows user interface. The machine also follows a trend but breaks new ground in the Apple II line by including large amounts of system firmware that is as important as the machine's new hardware features.

Because Apple perceives itself as a "premium label," its pricing will not be as aggressive as many users would like it to be. Apple is becoming—dare we say it?—more and more like IBM, selling more on name, reputation, and installed base of software and hardware (not a strong selling point, in the case of Apple II software) than on computing-power-per-dollar value.

The Apple II GS, hog-tied by Apple II compatibility, approaches but does not match or exceed current microcomputer capabilities. The 8086-like segmented memory of the 65C816 is not as elegant as that of the 68000, used in the Apple Macintosh, the Commodore Amiga, and the Atari 520ST. In addition, the 65C816 lacks the hardware multiply and divide instructions available in both the 8086 and the 68000 processors. The Apple II GS's graphics, though now competitive, do not offer any advantages over the Amiga's or the Atari ST's, nor is its price competitive with either. Its only clear superiority is in its sound capabilities, which for many buyers will not outweigh graphics and price.

Ironically, the Apple II GS will suffer from the traditional lack of software and hardware upon its introduction. Vendors will take longer than they expect to come out with new products, and many will enhance existing products for the Apple II instead of writing new software that fully exploits (and is limited to) the Apple II GS. Granted, a tremendous amount of software is already out there, but the Apple IIe and IIc will run it with fewer compatibility problems and at a significantly lower cost. As with new machines before it, people will buy the Apple II GS because they see the unrealized promise of its new features. ■