THE APPLE NOBODY KNOWS

by Alan Anderson

The story of the Apple /// is a fascinating one. Although this computer has only been public knowledge for about a year, its existence has had a profound effect on Apple Computer, Inc. and those of us who use their products. If you own or use an Apple II, you know about the Apple ///. You may have seen it lurking in a corner of your local computer store, with full color horses parading across the screen; you may have read about it in various magazines that have detailed its problems; but you probably have not discovered the real inner workings of this system. If you are a business user, you may be surprised to hear of some of its potential applications; if you a computer hobbyist, you will be interested in the amazing power hidden inside that curved chassis. Now let's explore together the past, present, and future of the still-mysterious Apple ///.

What Happened

The fact that an Apple /// would someday appear was never a secret. For at least a year before the Apple /// was introduced at the National Computer Conference in May 1980, the rumors flew fast and hard. But when Apple introduced the new system, the predictors were caught off guard. **Apple Magazine** proclaimed "A New Star is Born", but there were mutterings of "birth defects"; many were not impressed. Sure, it was pretty, but where was the hard-disk drive? No built-in color monitor? No Pascal in ROM? No 68000 microprocessor?

Of course, the Apple /// was missing these and other things that some folks had decided were essential. If the new computer had had a normal chance to show itself off, these design decisions would have been adequately explained at your local dealer. But a "normal" introduction period for the /// was hardly what occurred.

Of Clocks and Sockets

At NCC, Apple said that the first shipments of Apple ///'s would be dealer demos, one per store, and that they would start shipping in June. That schedule was quite optimistic, and it soon began slipping by several weeks. Then, it got worse: Apple began shipping demos, but they rarely worked. Apple's reputation for quality, won with thousands of trouble-free II's, began to deteriorate as the Apple /// saga unfolded.

Example: Apple discovered that their chip sockets were doing a lousy job of holding the integrated circuits in place, and that the slightest vibration in your friendly freight truck was enough to unseat the chips, placing the /// out of commission. This was a rude surprise for the unsuspecting Apple dealers. Ruder surprises awaited those who bought the first working computers; they frequently went dead after being installed in home or office, adding to the hassle. One solution was to allow the /// to fall vertically for 6-9 inches to a surface capable of providing a sudden stop, jarring the chips into place. The apparent brutality of this celebrated "drop fix" for sophisticated equipment gave rise to the suggestion that the "drop fix" might well be applied to a few engineers and marketeers.

So, new sockets were used, with a tighter grip, but Apple ///'s were still failing. Eventually, Apple discovered that the new sockets were tighter all right, but were jamming the chip pins back around, missing the socket. Again, they fixed the problem. Still, there were other hardware problems. The built-in disk drive sometimes didn't work if a plastic-enclosed monitor was placed on top of the unit. There were rumors of a nasty solder bridge on the motherboard, and of inadequate heat dissipation. Finally, after much work, Apple declared its reliability problems solved.

It should be noted that until June 1981, Apple's repair policy on ///'s was a model of simplicity: you send yours back and they send you a new one, fast. Having been through this procedure, I must say that even the most skeptical user comes out with his feelings soothed. Apple didn't even wait until the sick one got there before shipping the new one. More recently, Apple has started selling Apple /// service kits to its Level I service centers.

Then, there's the clock. When Apple announced the ///, one of its proudest features was a built-in clock/calendar chip that linked it to the operating system and stamped time and date on all your files. Well, bizarre things started occurring with the clock. The month began showing us as "???", and the hour would climb whimsically into the 30's and 40's before realizing that a new day was dawning. Since any boot diskette automatically displayed time and date, this particular black eye got great exposure.

Eventually, Apple announced that it was unable to find a reliable large-volume source for the clock chips, and stopped putting them in. The retail price was lowered $50, and Apple /// owners were offered a $50 rebate. Someday, when good chips can be obtained, the clock will go back. Someday. . .

But What Will it Do?

The Apple ///'s hardships have not been limited to hardware. The Sophisticated Operating System

(SOS) had some problems, quickly resolved, but became known as a memory-eater. Business BASIC still contains bugs, but the /// does a good job of emulating a single-language Apple II. There was virtually no software, except for the 80-column VisiCalc ///; people were hearing things like, "I spent $4000+, and got VisiCalc and a paperweight!"

Schedules and release dates slipped further and further from initial estimates. Pascal was scheduled for August 1981 release. . . see if it's out with this issue of the Apple Orchard. Other languages and software, like COBOL and Fortran, have also been subject to delays. The biggest blot in the software area has been Word Painter, Apple's high-quality word processor. This product is now more than a year behind schedule and is forecast to appear late this year. These delays, it seems, will have been worth it; the products are undergoing extensive testing, and will be of higher quality than if Apple had rushed them to judgment in the rumor-ridden marketplace.

The Past is Behind Us

At last, it look like Apple /// the Product is coming together. Local service is becoming available, as is the Extended Warranty. Reliability is up, says Apple, to a level comparable with the Apple II. And there are fewer gripes, growls, and whines coming from Apple /// owners within my earshot. (Not "none", just "fewer".)

So what's holding it back now? Two things. The first is the acute shortage of software. There are virtually no application programs available, and programming tools are likewise non-existent. There is as yet no assembler which provides the proper interfaces with the Apple /// operating system. The appearance of abundant software would greatly help the Apple ///.

But that's one of the things being retarded by the second problem, which is the image of the Apple /// as a stiff! Until dealers and consumers see the system running reliably, it will not be accepted as the Apple II has been. The tragedy is that public perception lags behind the actual improvements by three to six months; programmers who could solve the software shortage are reluctant to invest time in a machine which they hear has problems. Only time can cure this one, depending on the rate of improvement of the Apple ///'s public image.

The Goodies

Now that I've spent your time telling you about the checkered history of the Apple ///, why should you be interested in hearing more about the thing? Because the Apple /// is a uniquely well-designed personal computer system, remarkably powerful, and it has been plagued by stupid things like bent pins, solder bridges, corporate PR games ("what clock?") and negative attitudes. The Apple /// itself deserves a closer look.

The most obvious factor in the Apple ///'s design is the legacy of the Apple II. The /// reflects many of the things that were done right on the II, such as expansion capabilities; and corrects some of the hassles of the II, such as combining all languages under one operating system. With the Apple II as a sound base, the Apple ///'s design begins to take form. The standard memory configuration is 128K RAM. The microprocessor starts out as a 6502 A, a faster version of the Apple II's brain, and then has its capabilities enlarged by some additional circuitry. The built-in disk drive is basically the same as the ones we get for Apple IIs, with the same 140K bytes of data per diskette. Up to three more drives can be plugged into the back with no additional controller needed.

The Apple ///'s keyboard is a more complete version of the II's. All 128 ASCII characters are typeable, including full upper/lower case and alpha lock key. **All** keys have auto-repeat just by holding them down. A numeric keypad sits adjacent to the main keyboard. There are arrow keys for all four main points to the compass, and each of these keys has auto-repeat with **two** speeds, depending on how hard the key is pressed.

The Apple /// provides three different forms of text screen output, starting with the 40-character wide by 24-line high screen we see on the standard Apple II (40×24). The second mode is 80 characters wide by 24 lines high (80×24). The third mode is 40×24, but with the capability to make each letter, and each letter's background, any of 16 colors! But the real topper is that for all three text modes, the character set is defined in Random Access Memory (RAM), not frozen in Read-only Memory (ROM). This means that you can redefine the way characters look; so you can print different fonts, Japanese characters, even characters that look like horses. (Uh huh. . . that 16-color horse demo you see isn't graphics at all; it's text mode, with the characters redefined.) Those of you who have seen the Hi-Res Character Generator in Apple's DOS Tool Kit are familiar

with the technique of redefining the character set. However, there's a big difference: on the II, this has to be done in graphics mode, and it's slow. On the ///, it's done in text mode, so it's just as fast to print horses, frogs, and Greek as it is to print the English alphabet.

One of the Apple ///'s more interesting concepts is its lack of ROM; the only ROM code in the machine is a 4K byte program which simply runs a quick test on the unit's hardware and then boots the disk. Once that disk is done, this ROM is replaced in memory by RAM—no space wasted.

If you're familiar with the Apple II's insides, you know that the memory from $C000 to $CFFF is used for input/output by built-in and peripheral devices. Well, in the Apple ///, that's how it's used too. . . sometimes. There's another neat little trick in the /// that causes this area to be RAM too. Those of you in the audience who are quick-witted will notice that, with the switches set properly, the ///'s memory looks an awful lot like an Apple II; at other times, the whole memory space becomes RAM. (See Figure 1).

Why have all this RAM? It makes the Apple /// very "open-minded". With no language in ROM, the /// doesn't lock itself to the present selection of languages. If the Apple II had been made with all RAM, we wouldn't need a Language Card to run Pascal. Of course, we would have had to load BASIC by cassette, since the disk drives didn't exist when the II first appeared. But with the ///, everybody has a disk drive, so loading the language—any language—is fast and easy. By the way, not only does the /// load the language from disk, but also all the operating software, the character set, even the keyboard layout which designates how the keys correspond to the character set.

Speak to Me

The Apple /// has interfacing capabilities too. Even the standard I/O is kinda fancy. For example, there are three different video signals available: NTSC (standard) black and white; NTSC color, and RGB (studio quality) color. The black-and-white plug causes the colors to appear as sixteen shades of gray. There are also three different audio generators. One makes a beep, another makes various one-bit sounds (just like the Apple II), and the third is a 6-bit digital-to-analog converter that gives greater resolution to sounds.

The Apple /// has a serial interface built in, suitable for hooking up printers and modems. There is also a built-in interface for Apple's Silentype printer. The Silentype connector and one other port are also joystick hookups, and it's quite simple to modify many of the existing joysticks for use with the ///.

Inside the Apple are four 50-pin slots very much like the ones in the Apple II. In fact, the FCC may not like it, but you can plug in many Apple II peripheral cards and they will work fine.

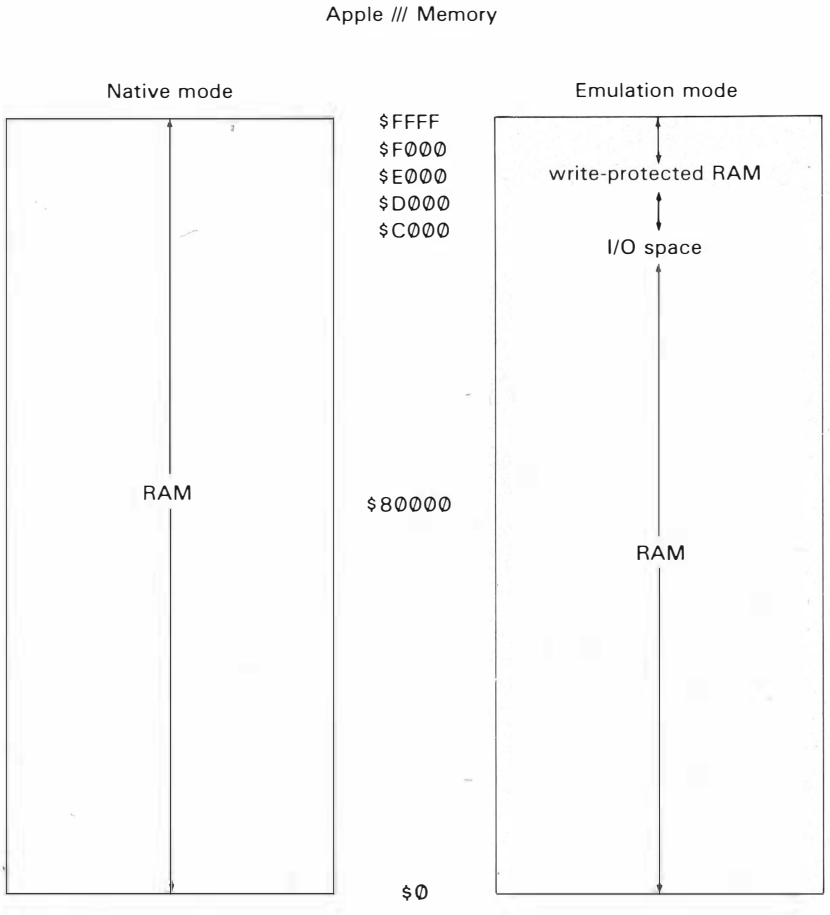

Figure 1

And Now, the Rest of the Story

This part is for the hobbyist, the experimenters, and the curious hackers among you. If you've had your Apple II long enough to remember the discovery of (POKE 33,33) in editing, the advent of the S. H. Lam Monitor routine, or the first CHR$ function for Integer BASIC, then you know what the early days were like. Well friends, come join me as we explore the secrets of the Apple ///. Most of these things are not yet documented, but already they are starting to become known. One day, Apple Computer Inc. will document them, and then we'll all know these things work to a very high degree. Until then, we present for your interest, Untold Stores of the Apple ///!

1. The Monitor Lurks Within

Inside that 4K diagnostic/boot ROM mentioned earlier is the first real development tool available for the ///: the Monitor. The Monitor is based on the Apple II Monitor. Several commands are the same as the II; specifically: the ones for dumping, moving, and verifying memory, and the G command are the same. The existence of this Monitor, though undocumented publicly, is now fairly well known. **(But, people at Apple have said that the Monitor may not be included in Apple ///s after some point.—PCW)**

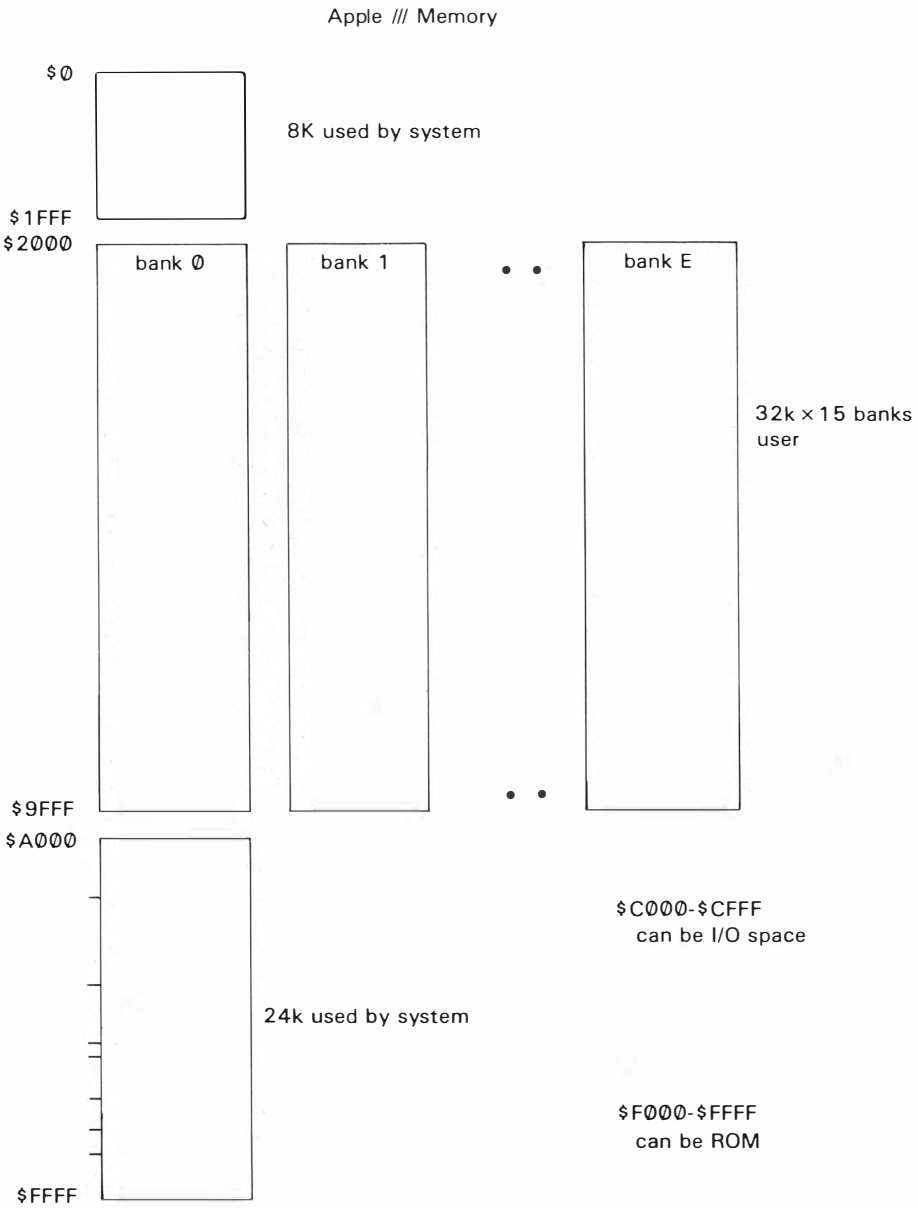

Figure 2

Here are the other commands:

A. (byte)<(addrs) . S
—searches the range of addresses (addrs) for the given byte.
Example: B5<3000.3FFFS

B. (blocknum)<(addrs) R
(blocknum)<(addrs) . (addrs) R
—reads the disk from the given block number into the given addresses. Blocks are 512 ($200) byte chunks, as with Pascal. One diskette contains 280 ($118) blocks. If you use the second form, enough consecutive blocks to fill the address space given will be read.
Examples: 4A<1000R (reads $4A into $1000-$11FF) 107<2500.28FFR (reads blocks $107-108 into $2500-28FF)

C. (blocknum)<(addrs) W
(blocknum)<(addrs) . (addrs) W
—writes to the disk. Analogous to the READ command above.

D. (addrs) J
—performs a jump (JMP) to the address given.

The Monitor comes up in 40-character mode. To switch to 80-column mode, press (ESC)-8. Pressing (ESC)-4 puts you back in 40-mode. To put more than one command on a line, put a slash between each pair. To stop and start video output, press the space bar. Press (TAB) to abort a listing. To enter the Monitor, press and hold (CTRL) and (OPEN APPLE) down; press and release (RESET). Keep holding the others until the Apple beeps. Have fun!

2. Banks for the Memories

One of the most oft-asked questions about the Apple /// is, of course, "How does the 6502 address all that memory?" There are lots of cute answers to that one: "Very carefully...", "With mirrors...", "Slowly...", and others. Actually, the masses of memory in the /// are handled through the time honored tradition of bank switching. Figure 2 gives you an idea of how it works.

First, you'll note that the 32K chunk from $2000 to $9FFF is replicated a number of times. This is called the User Area, and in the 512K Apple /// (theoretical for now) there are 15 of these, numbered from $0 to $E. Each of these 32K areas is called a bank. The one currently being addressed is determined by a memory-mapped register called the bank register (pretty tricky, eh?). This register is located at $FFEF, and you can change it in the Monitor. Normally, SOS handles all the necessary switches between banks from high-level languages. Oh... the present 128K Apple /// contains three User Areas, numbered $0 to $2. Some owners have added a fourth area ($3)... they hope... by upgrading memory to 160K.

In addition to this switching of RAM areas, the /// has two other bank switches. One controls the area from $C000 to $CFFF, selecting whether this space is RAM or I/O ROM. The other controls $F000 through $FFFF, which can be ROM or RAM, as mentioned earlier. These switches are kept in the environment register, described next (read on!).

3. Environmental Impact

Some of the Apple ///'s magic tricks are controlled by another memory-mapped register, the environment register. This register is mapped in at $FFDF and is shown in Figure 3.

Explanation:

Bit 7: When in the Emulator mode, this switch is set for 1 MHz to emulate the Apple II's clock.

Bit 6: This switch chooses what goes in $C000-$CFFF (see bank discussion above.)

Bit 5: This switch allows the video generator to go off, thus speeding up processing.

Bit 4: Turns off (RESET).

Bit 3: Used to write-protect $C000-$CFFF in emulation mode.

Bit 2: Maps the 6502 stack to a different location or normal ($100-$1FF) location.

Bit 1: Selects between two ROMs which may be mapped into the $F000-$FFFF space.

Bit 0: This switch chooses whether RAM or ROM goes into $F000-$FFFF.

Of course, this register may also be manipulated in the Monitor. Warning: it's easy to lose control and be forced to power off-and-on. However, you can't hurt the hardware, so your investment is protected.

In future articles, we'll cover some of these items:

—How to write Invokable Modules for Fun and Profit;

—How to communicate with the omnipresent SOS;

—Low-level secrets of the Mysterious Keyboard;

—and even more!

Stay tuned to the **Apple Orchard** for future developments, and perhaps the Apple /// will no longer be "the Apple Nobody Knows".

BIT	USE	if off	if on
7	Microprocessor speed	2 MHz	1 MHz
6	$C000-$CFFF switch	RAM	I/O ROM
5	Video Output	off	on
4	RESET key	disabled	enabled
3	Write-lock $C000-$CFFF	unprotected	protected
2	Stack	alternate	$100-$1FF
1	$F000-$FFFF	chip A	chip B
0	$F000-$FFFF switch	RAM	ROM

Figure 3

Alan Anderson is a writer and computer programmer who has a number of products and product improvements to his credit. His articles have appeared in virtually every popular journal of the Apple /// world, admittedly a limited circulation so far. He is also an advocate of chemical spraying to prevent software bugs.

Mr. Anderson purchased an Apple /// during the Dark Days, and has stayed with it through thin and thin. We hope to chronicle his and everyone's progress toward the brave new Apple /// world.