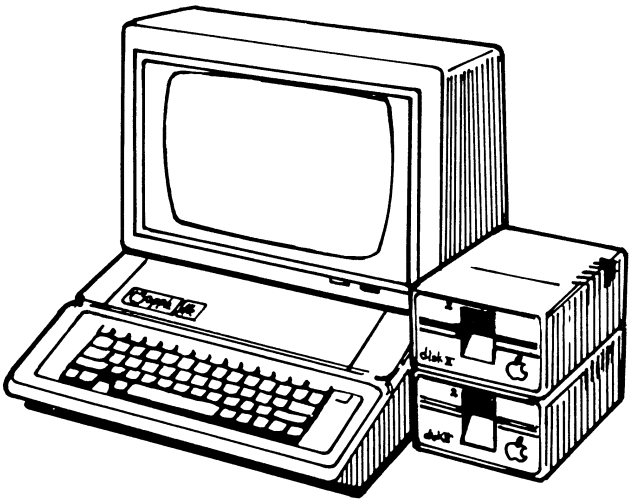
Apple II History

Compiled and written by Steven Weyhrich

Source
http://www.blinkenlights.com/classiccmp/apple2history.html
31 October 2004

TABLE OF CONTENTS

PART 01 -- PRE-APPLE HISTORY
PART 02 -- THE APPLE I
PART 03 -- THE APPLE II
PART 04 -- THE APPLE II, CONT.
PART 05 -- THE DISK II
PART 06 -- THE APPLE II PLUS
PART 07 -- THE APPLE IIE
PART 08 -- THE APPLE IIC
PART 09 -- DISK EVOLUTION / THE APPLE IIC PLUS
PART 10 -- THE APPLE IIGS
PART 11 -- THE APPLE IIGS, CONT.
PART 12 -- PERIPHERALS & THE APPLE II ABROAD
PART 13 -- PERIPHERALS, CONT.
PART 14 -- DOS
PART 15 -- DOS 3.3, PRODOS & BEYOND
PART 16 -- LANGUAGES
PART 17 -- LANGUAGES, CONT.
PART 18 -- SOFTWARE



APPLE II HISTORY

Compiled and written by Steven Weyhrich

(PART 1 -- PRE-APPLE HISTORY)
[v1.1 :: 12 Dec 91]

INTRODUCTION

This project began as a description of how the Apple II evolved into a IIGS, and some of the standards that emerged along the way. It has grown into a history of Apple Computer, with an emphasis on the place of the Apple II in that history. It has been gleaned from a variety of magazine articles and books that I have collected over the years, supplemented by information supplied by individuals who were "there" when it happened. I have tried not to spend much time on information that has been often repeated, but rather on the less known stories that led to the Apple II as we know it (and love it) today. Along the way I hope to present some interesting technical trivia, some thoughts about what the Apple II could have been, and what the Apple II still can be. The Apple II has been described as the computer that refuses to die. This story tells a little bit of why that is true.

If you are a new Apple II owner in 1991 and use any 8-bit Apple II software at all, you may feel bewildered by the seemingly nonsensical way in which certain things are laid out. AppleWorks asks which "slot" your printer is in. If you want to use the 80 column screen in Applesoft BASIC you must type an odd command, "PR#3". If you want to write PROGRAMS for Applesoft, you may have some of those ridiculous PEEKs and POKEs to contend with. The disk layout (which type is supposed to go into which slot) seems to be in some random order! And then there is the alphabet soup of disk systems: DOS 3.3, CP/M, Pascal, ProDOS, and GS/OS (if you have a IIGS). If you use 16-bit software EXCLUSIVELY, you will probably see none of this; however, even the most diehard GS user of the "latest and greatest" 16-bit programs will eventually need to use an 8-bit program. If you can tolerate a history lesson and would like to know "the rest of the story," I will try to make sense of it all.

I think one of the Apple II's greatest strengths is the attention they have paid over the years to be backward compatible. That means that a IIGS "power system" manufactured in 1991, with 8 meg of memory, a hand-held optical scanner, CD-ROM drive, and 150 meg of hard disk storage can still run an Integer BASIC program written in 1977, probably without ANY modification! In the world of microcomputers, where technology continues to advance monthly, and old programs may or may not run on the new models, that consistency is amazing to me. Consider the quantum leap in complexity and function between the original 4K Apple][and the ROM 03 IIGS; the amount of firmware (built-in programs) in the IIGS is larger than the entire RAM SPACE in a fully expanded original Apple][!

This strength of the Apple II could also be considered a weakness, because it presents a major difficulty in making design improvements that keep up with the advances in computer technology between 1976 and the

present, and yet maintain that compatibility with the past. Other early computer makers found it easy to design improvements that created a better machine, but they did so at the expense of their existing user base (Commodore comes to mind, with the PET, Vic 20, Commodore 64, and lastly the Amiga, all completely incompatible). However, this attention to detail is just one of the things that has made the Apple II the long-lived computer that it is.

In examining the development of the Apple II, we will take a look at some pre-Apple microcomputer history, the Apple I, and the formation of Apple Computers, Inc., with some sideroads into ways in which early users overcame the limits of their systems. We will follow through with the development of the Apple IIe, IIc, and IIGS, and lastly make some comments on the current state of affairs at Apple Inc. regarding the Apple II.

PRE-APPLE HISTORY

Let's begin our adventure in history. I've designed a special interface card that plugs into slot 7 on an Apple II. It contains an item its inventor called a "Flux Capacitor" (something about the being able to modify flux and flow of time). The card derives its power from a self-contained generator called "Mr. Fusion" (another item I dug out of the wreckage from a train/auto accident in California a couple of years ago). Connected to the card via a specially shielded line, Mr. Fusion runs on trash (and is, therefore, the ultimate computer peripheral, if you recall the old principal of "garbage in, garbage out"). Let's put a few issues of PC MAGAZINE into Mr. Fusion, and switch on the Flux Capacitor. (Incidentally, for this to work, it needs an Apple II equipped with a specially modified Zip chip running at 88 MHz). Boot the disk and set the time circuits for 1975. Ready? Set? Go! ** CRACKADOOM ** !!

Did you make it all right? (Just don't touch anything -- you don't want to disrupt the space-time continuum, you know!) Now, since the first Apple II wasn't released until 1977, what are we doing back in 1975? Well, to understand how the Apple II came about, it helps to know the environment that produced it. In 1975, the microcomputer industry was still very much in its infancy. There were few "home computers" that you can choose from, and their capabilities were very much limited. The first microprocessor chip, the 4-bit 4004, had been released by Intel back in 1971. The first video game, Pong, was created by Nolan Bushnell of Atari in 1972. Also in 1972, Intel had gone a step further in microprocessor development and released the 8-bit 8008, and then the 8080 in 1973. The year 1974 saw Scelbi Computer Consulting sell what some consider to be the first commercially built microcomputer, the Scelbi 8-H, based on Intel's 8008 chip. However, it had limited distribution and due to the designer's health problems it didn't go very far. The first home-built computer, the Mark 8, was released that same year. The Mark 8 used the Intel 8080 chip, but had no power supply, monitor, keyboard, or case, and only a few hobbyists ever finished their kits. Overall, the microchip had yet to make much of an impact on the general public beyond the introduction of the hand-held calculator.

With the start of 1975 came a significant event in microcomputer history. If you will consider the early microprocessors of the years 1971 through 1974 as a time of germination and "pregnancy" of ideas and various hardware designs, January of 1975 saw the "labor and delivery" of a special

package. The birth announcement was splashed on the front cover of a hacker's magazine, Popular Electronics. The baby's parents, MITS, Inc., named it "Altair 8800"; it measured 18-inches deep by 17 inches wide by 7 inches high, and it weighed in at a massive 256 bytes (that's one fourth of a "K"). Called the "World's First Minicomputer Kit to Rival Commercial Models," the Altair 8800 used the Intel 8080 chip, and sold for $395 (or $498 fully assembled). MITS hoped that they would get about four hundred orders for clones of this baby, trickling in over the months that the two-part article was printed. This would supply the money MITS needed to buy the parts to send to people ordering the kits (one common way those days of "bootstrapping" a small electronics business). This "trickle" of orders would also give MITS time to establish a proper assembly line for packaging the kits. However, they misjudged the burning desire of Popular Electronic's readers to build and operate their own computer. MITS received four hundred orders in ONE AFTERNOON, and in three weeks it had taken in $250,000.<1>

The Popular Electronics article was a bit exuberant in the way the Altair 8800 was described. They called it "a full-blown computer that can hold its own against sophisticated minicomputers now on the market... The Altair 8800 is not a 'demonstrator' or souped-up calculator... [it] is a complete system." The article had an insert that lists some possible applications for the computer, stating that "the Altair 8800 is so powerful, in fact, that many of these applications can be performed simultaneously." Among the possible uses listed are an automated control for a ham station, a digital clock with time zone conversion, an autopilot for planes and boats, navigation computer, a brain for a robot, a pattern-recognition device, and a printed matter-to-Braille converter for the blind.<2> Many of these things will be possible with microcomputers by 1991, but even by then few people will have the hardware add-ons to make some of these applications possible. Also, despite the power that micros will have in that year, the ability to carry out more than one of these applications "simultaneously" will not be not practical or in some cases even possible. The exaggeration by the authors of the Popular Electronics article can perhaps be excused by their excitement in being able to offer a computer that ANYONE can own and use. All this was promised from a computer that came "complete" with only 256 bytes of memory (expandable if you can afford it) and no keyboard, monitor, or storage device.

The IMSAI 8080 (an Altair clone) also came out in 1975 and did fairly well in the hobbyist market. Another popular early computer, the Sol, would not be released until the following year. Other computers released in 1975 that enjoyed limited success were the Altair 680 (also from MITS, Inc., based on the Motorola 6800 processor), the Jupiter II (Wavemate), M6800 (Southwest Technical Products), and the JOLT (Microcomputer Associates), all kits.<3> The entire microcomputer market was still very much a hobbyist market, best suited for those who enjoyed assembling a computer from a kit. After you assembled your computer, you either had to write your own programs (from assembly language) or enter a program someone else wrote. If you could afford the extra memory and the cost of buying a BASIC interpreter, you might have been able to write some small programs that ran in that language instead of having to figure out 8080 assembly language. If you were lucky (or rich) you had 16K of memory, possibly more; if you were REALLY lucky you owned (or could borrow) a surplus paper tape reader to avoid typing in manually your friend's checkbook balancing program. Did I say typing? Many early computer hobbyists didn't even have the interface allowing them to TYPE from a keyboard or teletype. The

"complete" Altair 8800 discussed above could only be programmed by entering data via tiny little switches on its front panel, as either octal (base 8) bytes or hexadecimal (base 16) bytes. With no television monitor available either, the results of the program were read in binary (base 2) from lights on that front panel. This may sound like the old story that begins with the statement, "I had to walk five miles to school through snow three feet deep when I was your age," but it helps to understand how things were at this time to see what a leap forward the Apple II really was (er, will be. Time travel complicates grammar!)

++++++++++++++++++++++++++++

NEXT INSTALLMENT: The Apple I

++++++++++++++++++++++++++++

NOTES

<1> Steven Levy, HACKERS: HEROES OF THE COMPUTER REVOLUTION, pp. 187-192.

<2> H. Edward Roberts and William Yates, "Altair 8800 Minicomputer, Part 1", POPULAR ELECTRONICS, January 1975, pp. 33, 38. The article is interesting also in some of the terminology that is used. The Altair is described as having "256 eight-bit words" of RAM. Apparently, the term "byte" was not in common use yet.

<3> Gene Smarte and Andrew Reinhardt, "15 Years of Bits, Bytes, and Other Great Moments", BYTE, September 1990, pp. 370-371.

This is the ENTIRE series of articles that make up the Apple II History. They are readable in either AppleWorks 2.x or 3.0, but will require an expanded desktop for some segments.

Please feel free to make comments (on GEnie's A2 Roundtable, Category 2, Topic 16) or in E-mail (S.WEYHRICH) about the contents of these files. PLEASE, if you detect any errors or have any corrections, let me know about it. I would like to have as accurate a history as possible.

If you would like to print any of these files in a user group newsletter, I only ask that you print any segment you use in its entirety, and that you give me as the author credit for the work. Also, please send me a copy of any newsletter in which it is printed. My address is:

Steven Weyhrich
Zonker Software
2715 N. 112th St.
Omaha, NE 68164-3666

(402) 498-0246

Enjoy!

APPLE II HISTORY

Compiled and written by Steven Weyhrich

(PART 2 -- THE APPLE I)
[v1.1 :: 12 Dec 91]

THE APPLE I: DEVELOPMENT

At the Homebrew Computer club in Palo Alto, California (in Silicon Valley), Steve Wozniak, a 26 year old employee of Hewlett-Packard and a long-time digital electronics hacker, had been wanting to build a computer of his own for a long time. For years he had designed many on paper, and even written FORTRAN compilers and BASIC interpreters for these theoretical machines, but a lack of money kept him from carrying out his desire. He looked at the Intel 8080 chip (the heart of the Altair), but at $179 decided he couldn't afford it. A decision to NOT use the 8080 was considered foolhardy by other members of the club. Consider this description of the microcomputer "world" as it was in the summer of 1975:

> "That summer at the Homebrew Club the Intel 8080 formed the center of the universe. The Altair was built around the 8080 and its early popularity spawned a cottage industry of small companies that either made machines that would run programs written for the Altair or made attachments that would plug into the computer. The private peculiarities of microprocessors meant that a program or device designed for one would not work on another. The junction of these peripheral devices for the Altair was known as the S-100 bus because it used one hundred signal lines. Disciples of the 8080 formed religious attachments to the 8080 and S-100 even though they readily admitted that the latter was poorly designed. The people who wrote programs or built peripherals for 8080 computers thought that later, competing microprocessors were doomed. The sheer weight of the programs and the choice of peripherals, so the argument went, would make it more useful to more users and more profitable for more companies. The 8080, they liked to say, had critical mass which was sufficient to consign anything else to oblivion."<1>

Another chip, the Motorola 6800, interested Wozniak because it resembled his favorite minicomputers (such as the Data General Nova) more than the 8080. However, cost was still a problem for him until he and his friend Allen Baum discovered a chip that was almost identical to the 6800, while considerably cheaper. MOS Technology sold their 6502 chip for $25, as opposed to the $175 Motorola 6800. Wozniak decided to change his choice of processor to the 6502 and began writing a version of BASIC that would run on it. A friend over at Hewlett-Packard programmed a computer to simulate the function of the 6502, and Wozniak used it to test some of his early routines. When his BASIC interpreter was finished, he turned his

attention to designing the computer he could run it on. Except for some small timing differences, he was able to use the hardware design he had earlier done on paper for the 6800.<2>

To make the computer easier to use, Wozniak favored a keyboard over the front panel switches that came on the Altair. He also made it simple to use a television for a video terminal. (Recall that at this time the most common mechanism used for input/output was a teletype, which consisted of a keyboard, typewriter, and if you were lucky, a paper tape reader/puncher). Functionally, it was a television terminal attached to a computer, all on one printed circuit board (another enhancement over the Altair). Wozniak used two 256 x 4 PROM (programmable read-only memory) chips to create a 256 byte program (called a "monitor") that looked at the keyboard when the computer was turned on. This monitor program could not do much more than allow entry of hex bytes, examine a range of memory, and run a program at a specific address.<3> (The Altair needed these "bootstrapping" instructions to be entered by hand each time the computer was turned on).

Because there were no cheap RAMs available, Woz used shift registers to send text to the TV screen. Consequently, his video terminal was somewhat slow, displaying characters at about 60 characters per second, one character per scan of the TV screen. (This speed would be similar to watching a computer communicate via a modem at 1200 baud). It was slow by 1991 standards, but an advancement over the teletypes that could only type 10 characters per second. The computer had 8K of dynamic RAM. You could load BASIC into 4K of memory and have 4K left over for your own programs. It had a video connector, but you had to connect a monitor on your own. You also had to buy the keyboard separately and wire it into a 16-pin DIP connector. The power supply had to be connected to two transformers to get 5 volts and 12 volts for the motherboard. There was no speaker, no graphics, and no color. There was a single peripheral slot, and when it was first released there was nothing available to plug into this slot. It was entirely contained on a single printed circuit board, about six by eight inches in size (most hobby computers of that time needed at least two boards), used only 30 or 40 chips, and because it could run BASIC programs it got people's attention.<4>

THE APPLE I: MARKETING

Let's adjust our time circuits for 1976, and jump forward in time. By now, Steve Wozniak had completed his 6502-based computer and would display enhancements or modifications at the bi-weekly Homebrew Computer Club meetings. Steve Jobs was a 21 year old friend of Wozniak's and also a visitor at the Homebrew club. He had worked with Wozniak in the past (together they designed the arcade game "Breakout" for Atari) and was very interested in his computer. During the design process Jobs made suggestions that helped shape the final product, such as the use of the newer dynamic RAMs instead of older, more expensive static RAMs. He suggested to Wozniak that they get some printed circuit boards made for the computer and sell it at the club for people to assemble themselves. They pooled their financial resources together to have PC boards made, and on April 1st, 1976 they officially formed the Apple Computer Company. Jobs had recently worked at an organic apple orchard, and liked the name because "he thought of the apple as the perfect fruit--it has a high nutritional content, it comes in a nice package, it doesn't damage easily--and he

wanted Apple to be the perfect company. Besides, they couldn't come up with a better name."<5>

Jobs approached the owner of a new computer store in the bay area called "The Byte Shop." This businessman, Paul Terrell, expressed an interest in the Apple Computer (to be known later as the "Apple I"), but wanted only fully assembled computers to sell. If they could provide this, Terrell told them he would order fifty Apples, and pay cash on delivery. Suddenly, the cost of making (and selling) this computer was considerably more than they expected. Jobs and Wozniak managed to get the parts on "net 30 days" (30 days credit without interest), and set themselves up in Job's garage for assembly and testing of the Apple I. After marathon sessions of stuffing and soldering PC boards, Jobs delivered the computers to the Byte Shop. Although these "fully assembled" computers lacked a power supply, keyboard, or monitor, Terrell bought them as promised. In July of 1976 the Apple I was released and sold for $666.66, which was about twice the cost of the parts plus a 33% dealer markup.<6> Two hundred Apple I computers were manufactured, and all except twenty-five of them sold over a period of ten months.<7>

Although the Apple I was easier to begin using than the Altair (thanks to its built-in ROM code), it was still a time consuming process to set it up to do something useful. Steve Wozniak would have to type in about 3K of hexadecimal bytes before BASIC was ready to use. He could do it in about 20 to 30 minutes, but he almost knew the code by heart. The typical user was more limited in ability to use BASIC on the Apple I. To broaden the appeal of the Apple I (and at the insistence of Paul Terrell), Wozniak designed a cassette interface. It was mounted on a small two-inch-high printed circuit board and plugged into the single slot on the motherboard. The card sold for $75 and a cassette tape of Woz's BASIC was included with it. The advertisement Apple included with the card stated, "Our philosophy is to provide software for our machines free or at minimal cost." The interface worked, but worked well only with cassettes running on expensive tape recorders. To further try to enhance sales, the Byte Shop stores found a local cabinetmaker that made some koa-wood cases for the Apple computer (so it would no longer be just a "naked" circuit board).<8>

Interestingly, although most of the action in the micro world was going on in Silicon Valley, news of the Apple I made its way east. Stan Veit, owner of the east coast's first computer store, bought an Apple I and took it to a meeting of the Association of Computer Machinery. Those attending were quite skeptical that a REAL computer could fit into a small briefcase; they were sure that the machine was just a portable terminal, attached by a hidden phone line to a mainframe somewhere!<9>

++++++++++++++++++++++++++++++

NEXT INSTALLMENT: The Apple II

++++++++++++++++++++++++++++++

NOTES

<1> Michael Moritz, THE LITTLE KINGDOM, p. 123.

<2> Moritz, pp. 124-127.

<3> Williams & Moore, p. A69.

<4> Gregg Williams and Rob Moore, "The Apple Story, Part 1: Early History", BYTE, Dec 1984, pp. A68-A69.

<5> Frank Rose, WEST OF EDEN: THE END OF INNOCENCE AT APPLE COMPUTER, p. 33.

<6> Moritz, pp. 138-144.

<7> Williams & Moore, pp. A69.

<8> Moritz, pp. 147-149.

<9> Chien, Philip, "Apple's First Decade: A Look Back", THE APPLE II REVIEW, Fall/Winter 1986, p. 12.

APPLE II HISTORY

Compiled and written by Steven Weyhrich

(PART 3 -- THE APPLE II)
[v1.1 :: 12 Dec 91]

THE APPLE II: HARDWARE AND FIRMWARE

Moving our time machine on to 1977, we can now look at Steve Wozniak's next generation Apple. Even as the Apple I was completed and was slowly selling, Wozniak was already working on making enhancements that would make his computer faster and more functional. He wanted to make it display in color. He worked to combine the terminal and memory functions of the Apple I by moving the display into main memory, allowing instant screen changes. Many of his changes were not added with the end user specifically in mind. Wozniak stated:

> "A lot of features of the Apple II went in because I had designed Breakout for Atari. I had designed it in hardware. I wanted to write it in software now. So that was the reason that color was added in first--so that games could be programmed. I sat down one night and tried to put it into BASIC. Fortunately I had written the BASIC myself, so I just burned some new ROMs with line drawing commands, color changing commands, and various BASIC commands that would plot in color. I got this ball bouncing around, and I said, 'Well it needs sound,' and I had to add a speaker to the Apple II. It wasn't planned, it was just accidental... Obviously you need paddles, so I had to scratch my head and design a simple minimum-chip paddle circuit, and put on some paddles. So a lot of these features that really made the Apple II stand out in its day came from a game, and the fun features that were built in were only to do one pet project, which was to program a BASIC version of Breakout and show it off at the club."<1>

Wozniak added other features that he felt were important for a computer that was useful, one that he would want to own. Since the 6502 processor could address a total of 64K of memory, he designed the computer with the ability to use either 4K RAM chips, or the newer (and more expensive) 16K RAM chips. The first Apple II's came standard with 4K of memory, and more could be added, to a maximum of 12K (if using the 4K chips) or 48K (if using the 16K chips). Specially wired strapping blocks attached to the motherboard told the Apple II how much memory was present and where it was. According to the 1981 edition of the APPLE II REFERENCE MANUAL, the Apple could have memory in the following sizes: 4K, 8K, 12K, 16K, 20K, 24K, 32K, 36K, or a full 48K. (These sizes were determined by the different ways that three RAM chips, either 4K or 16K, could be installed). The strapping blocks were even designed with the flexibility

of allowing blank spots in memory if there were no RAM chips available to fill those spots.

The first 4K of memory always had to have RAM present, since it was used by the 6502 processor, the ROM routines, and the text screen display. If, for example, you only had two other 4K RAM chips to install and you wanted to display hi-res graphics, you could strap one chip to the lower half of hi-res memory from $2000-$2FFF, and the other to the upper half of hi-res memory from $3000-$3FFF.<2> Since 16K RAM chips cost about $500 when Wozniak designed the Apple II, not many users could afford them. Whereas the Commodore PET and the Radio Shack TRS-80 could not easily be expanded beyond the 4K they came with, the Apple II from the beginning was designed with expansion in mind.<3>

The row of eight expansion slots was another feature about the Apple II that was a strong selling point. Unlike the TRS-80 or PET, you could easily expand the Apple II by simply plugging a card into one of these slots. This degree of expandability made it more expensive to build, however. Steve Jobs didn't believe that anyone would ever need more than two slots, one for a printer and one possibly for a modem. Wozniak knew from his experience with computers at Hewlett-Packard that computer users would always find SOMETHING to fill those extra slots, and insisted that they keep the number at eight.<4>

One problem Apple had to deal with was getting FCC approval for the computer. The RF (radio frequency) modulator that had been designed gave off too much interference, and it was probable that the FCC would not approve it. (The RF modulator allowed a user to attach the Apple to a standard television receiver, instead of requiring the purchase of an expensive computer monitor). Rather than have the release of the Apple II delayed for re-engineering of the RF modulator to get that FCC approval, Apple gave the specifications for the RF modulator to Marty Spergel. He ran a small company (called M&R Electronics) that specialized in obtaining hard-to-get parts that electronics and computer hackers wanted for their projects. Their agreement allowed M&R to make and sell the RF modulators, while Apple could concentrate on making and selling the Apple II. Dealers would sell an Apple II with a "Sup'r Mod" (costing about $30) if the buyer wanted to see the graphics on their color TV. Jobs assured Spergel that the item would sell well, maybe as many as fifty units a month. (Years later Spergel estimated that he had sold about four hundred thousand Sup'r Mods).<5>

Other features that Wozniak (and Allen Baum, who helped him with the project) included in the Apple II ROMs included the terminal software to do screen text display, expanded Monitor functionality, and cassette input/output routines. They added the ability to split the screen into different sized windows. They also wrote a disassembler, which was one of the most important features of the Apple II from the beginning and a significant part of its open design. It allowed ANYONE to view the 6502 code that ANY program used, and matched the philosophy of the Homebrew Club of making all computer knowledge available to everybody. In the Apple I days, when Apple was supplying software "free or at minimal charge", Wozniak and Baum published an early version of their 6502 disassembler in a hacker's magazine. It was designed to be loaded in memory on the Apple I from $800 to $9D8 and the routine could be executed from the monitor. This early code was quit similar to the disassembler that was later included in the Apple II ROM.<6>

Having an expanded Monitor program in ROM and color graphics were not the only features in the Apple II that attracted people to it. Having

Wozniak's BASIC language in ROM, available immediately when the power was turned on, made it possible for non-hackers to write programs that used the Apple II's color graphics.

An interesting bit of trivia about Wozniak's Integer BASIC was that he never had an assembly language source file for it. He wrote it in machine language, assembling it by hand on paper:

> "I wrote this BASIC processor, and I wrote a little ALGOL simulator and got it simulated. It looked like it would work, but I had forgotten to build the machine. I had no assembler, that was another thing. To use an assembler, they figured that somebody was going to buy this processor [the 6502] to use for a company, and their company can pay a few thousand dollars in time-sharing charges to use an assembler that was available in time-share. I didn't have any money like that, so a friend taught me that you just sort of look at each instruction, you write your instructions on the right side of the page, you write the addresses over on the left side, and you then look up the hex data for each instruction--you could assemble it yourself. So I would just sit there and assemble it myself. The [Integer] BASIC, which we shipped with the first Apple II's, was never assembled--ever. There was one handwritten copy, all handwritten, all hand-assembled. So we were in an era that we could not afford tools."<7>

Even to this day there is not an official source code listing of Integer BASIC at Apple. And interestingly, the only error I am aware of in the Integer interpreter is one involving a single byte. If a line is entered that has too many parentheses, the "TOO LONG" error message is displayed instead of the "TOO MANY PARENS" message.<8>

NOW A WORD FROM OUR SPONSOR: BACK TO THE BASICS...

I want to take a short break in this discussion of the Apple II firmware to look at some other items that will make further descriptions easier to understand. If you are a programmer already, you may want to skip this section, since you probably already know this stuff. First we will examine some definitions of terms that are commonly known to programmers, but possibly not to you. Next will be a brief excursion into the realm of hexadecimal, and finally a look at the memory map of the original Apple II.

First, let's look at definitions of some words that I have been loosely throwing around:

BIT	The smallest piece of information that a computer can deal with, it is either a "0" (off, clear) or a "1" (on, set).
BYTE	The most convenient piece of information (for humans) that computers use. One byte consists of eight bits, and ranges from "00000000" (0 decimal) to "11111111" (255 decimal).
NIBBLE	(also spelled "nybble"). One half of a byte, consisting of four bits, ranging from "0000" (0 decimal) to "1111" (15

	decimal).
WORD	Two bytes (or four nibbles, if you prefer), consisting of sixteen bits, and ranging from "00000000 00000000" (0 decimal) to "11111111 11111111" (65535 decimal). Not used much in microcomputers.
BINARY	A system of counting using only two digits, "0" and "1" (base 2). Computers speak in binary at their most basic level; anything else is translated into binary, so the computer can understand it.
DECIMAL	A system of counting using ten digits, "0" through "9" (base 10). Most of the Western world uses this system.
HEXADECIMAL	A system of counting using sixteen digits, "0" through "9" and "A" through "F" (base 16). Programmers use this system as a convenient way of organizing groups of binary numbers.
KILOBYTE	Abbreviated "K", "KB", or "Kbytes", it refers to 1,024 bytes. A 64K computer has 64 x 1024 = 65536 bytes.
MEGABYTE	Abbreviated "M", "MB", or "meg", it refers to 1,024 Kbytes, or 1,024 x 1,024 = 1,048,576 bytes. A 32 MB hard disk, the largest size volume that ProDOS can handle, holds 32 x 1,024 = 32,768 Kbytes, or 32 x 1,024 x 1,024 = 33,554,432 bytes.
GIGABYTE	Abbreviated "G", "GB", or "gig", it refers to 1,024 MB, or 1,048,576 Kbytes, or 10,737,441,824 bytes. The Apple II Smartport (which will be mentioned later in this history) can handle disk devices up to 4 gig in size (although the software to handle that type of size has yet to be written).
RAM	Random Access Memory. Any data stored in this memory disappears when the computer is turned off.
ROM	Read Only Memory. Data cannot be stored in this type of memory, but instead it usually contains programs or other information that does not disappear when the computer is turned off.
HARDWARE	The physical electronic components and mechanical parts that make up a piece of computer equipment. Examples would be the keyboard, disk drive, or television monitor (also called CRT, or Cathode Ray Tube).
SOFTWARE	The digital instructions executed by the computer in RAM. They may act on the hardware that is attached to the computer. Examples would be a BASIC or Pascal program, an assembly language routine to read a clock, or a disk operating system. Since software is executed in RAM, it disappears from memory when the computer is turned off.
FIRMWARE	The same as software, except it is executed from ROM, and does not disappear when the computer is turned off. Almost any software could be in ROM, except programs that modify themselves as they run.

Next, let's look at hexadecimal numbers in more detail. Since computers deal in binary (base 2), the true language of computers is either in terms of "0" (off) or "1" (on). However, it quickly becomes cumbersome to refer to large numbers in binary; the base 10 number "458" is "111001010" in binary. So programmers have decided to group numbers in such a way as to make it easy to convert part or all of that number to binary if necessary, but still have numbers (almost) as easy to deal with as our standard base 10 system.

Now, in the familiar base 10 system there are ten digits, 0 through 9. When counting, after you pass 9, you add one to the digit to the left of the 9, change the 9 to a 0, and continue. So, "09" becomes "10", "19" becomes "20", and so on. However, in the base 16 system there are sixteen digits, 0 through 9, and then A through F (representing decimal 10 through 15). When counting, then, you go 7, 8, 9, then A (not 10), B, C, D, E, F, 10, 11, 12, and so on. In the Apple world we have traditionally used a preceding dollar sign to signify a hexadecimal number, so "25" means twenty-five, but "$25" means thirty-seven (2 x 16, plus 5). To translate a hexadecimal number to decimal, use powers of 16:

```
$B65F = (11 x 16^3)  +  (6 x 16^2)  +  (5 x 16^1)  +  (15 x 16^0)
      = (11 x 4096)  +  (6 x 256)   +  (5 x 16)    +  (15 x 1)
      = 45056        +  1536        +  80          +  15
      = 46687
```

The same thing can be done in reverse to convert base 10 to hexadecimal, starting by dividing the number by 4096, then the remainder by 256, then 16. If the number is greater than 65536, you need a bigger power of 16 (and you are probably not dealing with an 8-bit Apple II!) Or you can just get a programmer's calculator like mine that automatically does the conversion for you...

When dealing with memory addresses on an Apple II, we usually designate them as four digit hex numbers (such as the $B65F example above). Numbers less than $1000 often are printed without the leading blank ($400 instead of $0400), and numbers less than $100 are treated the same way ($32 instead of $0032).

THE APPLE II: MEMORY MAP

To understand the memory layout of the Apple II, consider this analogy: Imagine a cabinet with sixteen shelves, and sixteen separate slots or pigeon holes on each shelf (similar to those found in old roll-top desks). Each slot refers to a specific address in memory on the computer, and each slot can hold a number between 0 and 255. (Since a byte is eight bits wide, the largest number that can be represented by eight binary bits is 255). The bottom shelf is row "0", and the leftmost slot in that row is slot "0". The address of that slot, then, is $00. As we move to the right, the addresses increase, $01, $02, $03, and so on to $0F at the end. We then go up to the next row, (row "1"), and the addresses continue in the same fashion with $10, $11, $12, and so on as before. The sixteenth row is row "F", the rightmost slot in that row is slot "F", and the address of that slot is $FF. This cabinet has, then, 256 slots (16 x 16), and represents what is called a "page" in the Apple memory. The cabinet itself has an address (since computers need addresses for everything), and this one's address is "00". The full address of row "5", slot "A" on cabinet "00" is $005A.

Only the Altair 8800 came with just 256 bytes of memory, so we have to account for the entire 64K memory space that the 6502 chip in the Apple II can handle. There is a cabinet sitting on top of cabinet "00", and it is laid out in the same fashion with its 256 slots in sixteen rows. This is cabinet "01", and on top of that one is cabinet "02"; this continues on up until we reach cabinet "FF" way up at the top. Apple programmers refer to these cabinets as "pages" of memory. There are 256

pages of memory, each with 256 bytes on a page, making a grand total of 256 x 256 = 65536 bytes of memory (or slots that can hold a number, if you prefer the analogy).

In discussing the memory map on the Apple II, we can refer to pages of memory with a hexadecimal two-digit number for shorthand if we wish. The general layout of the Apple II memory is as follows:

```
     Page $00: used by the 6502 processor for storage of information that
               it can access quickly.  This is prime real-estate that is
               seldom available for general use by programmers without
               special care.
     Page $01: used by the 6502 for internal operations as a "stack."
     Page $02: used by the Apple II firmware as an input buffer when using
               the keyboard from BASIC, or when a program uses any of the
               firmware input routines.
     Page $03: general storage area, up to the top three rows (from $3D0
               through $3FF) which are used by the disk operating system
               and the firmware for pointers to internal routines.
Pages $04-$07: used for the 40 column text screen.
Pages $08-$BF: available for use by programs, operating systems, and for
               hi-res graphics.  Within this space, Woz designated pages
               $20-$3F for hi-res "page" one, and pages $40-$5F for hi-res
               "page" two.
     Page $C0: internal I/O and softswitches
Pages $C1-$C7: ROM assigned to each of the seven peripheral cards
Pages $C8-$CF: switchable ROM available for any of the seven cards
Pages $D0-$D7: empty ROM socket #1
Pages $D8-$DF: empty ROM socket #2
Pages $E0-$F7: Integer BASIC ROM
Pages $F8-$FF: Monitor ROM
```

The memory space on the Apple II between $C000 and $CFFF was assigned to handle input and output. From $C000 to $C0FF the space was reserved for various soft-switches used to control the display, and various built-in I/O devices, such as the keyboard, paddles, annunciators, and the cassette port. (A soft-switch is simply a memory location that, when a number is stored there, changes something in the computer--such as switching on graphics mode). From $C100 to $CFFF the space was reserved for ROM on the plug-in peripheral cards for each of the seven slots. Slot 1 was given the space from $C100 to $C1FF, slot 2 from $C200 to $C2FF, and so on. The $C800 to $CFFF space was special slot-selectable ROM that was uniquely available for each of the seven peripheral cards. For example, a program running on the card in slot 6 to control a device could use the $C800-$CFFF space for its own purpose. When control passed to the card in slot 3, that card could use a program of its own that ran in the same $C800-$CFFF space. This was accomplished by allowing each card to have ROM code that covered pages $C8-$CF, and making that space "switchable", depending on which card wanted to use it. Having this space available made writing ROM code simpler, since it would not have to be capable of running at various memory locations (depending on which slot a card was plugged into).

The memory from $D000 to $D7FF and $D800 to $DFFF was empty on all early Apple II computers. On the motherboard were two empty sockets that were available for the user to plug in their own ROM chips. The

$D000-$D7FF space was most often used by a plug-in ROM chip sold by Apple, known as "Programmer's Aid #1." It contained various utilities for Integer BASIC programmers, including machine language routines to do the following:

- Renumber BASIC programs
- Append one BASIC program to the end of another
- Verify a BASIC program that had been saved on tape (to confirm it was an accurate save)
- Verify non-program data that had been saved on tape
- Relocate assembly language routines to a different location in memory (most would only run in one place in memory)
- Test the Apple II RAM
- Generate musical tones through the built-in speaker
- Handle hi-res graphics from BASIC, including code to clear the hi-res screen, set colors, plot points and lines, draw shapes, and load shapes from tape.

All the routines on the Programmer's Aid #1 ROM were written by Wozniak between June 1977 (the RAM test routine) and April 1978 (program renumber and append), except for the music routine, which was written by Gary Shannon.

The other empty ROM socket (covering memory from $D800 to $DFFF) was never filled by Apple. Various third-party vendors sold ROMs for that socket (or for the $D000-$D7FF socket used by the Programmer's Aid #1 ROM), but none made enough of an inroad to be preserved in the INTBASIC file that would later be included on the DOS 3.3 System Master disk. In fact, the $D800-$DFFF space in the INTBASIC file on that disk contains an image of that same space taken directly from the Applesoft ROM! It is completely useless to Integer BASIC, of course, but disk files being what they are, Apple had to fill that space with SOMETHING!

The Integer BASIC interpreter lived in the ROM space between $E000 and $F7FF. However, BASIC only used the space up to $F424. Between $F425-$F4FB and $F63D-$F65D could be found a floating-point math package that was not used by Integer BASIC, but was available for BASIC programmers who were astute enough to figure out how it worked. (An early Apple user group, the Apple Pugetsound Program Library Exchange, or A.P.P.L.E., sold a tape and notes by Steve Wozniak they called "Wozpak", that documented some of the secrets of the Integer BASIC ROM). Between $F500-$F63C there was code that was known as the "miniassembler", which was executed starting at the ominous address $F666. The miniassembler allowed you to enter short machine language programs using the standard 6502 mnemonics (the three letter codes that referred to a specific type of operation; for example, "LDA #" represented the 6502 opcode $A9) instead of entering the program byte by byte in the monitor. The $F689-$F7FC space contained Woz's SWEET 16 interpreter. Wozniak wrote SWEET 16 to simulate a 16-bit processor; it simplified some routines he wrote for the Apple II ROMs, including the Programmer's Aid #1 renumber, append, and relocate routines. Simply put, he took a series of hex bytes, defined them as "opcodes" the way HE wanted them to function, and when executing the code used his SWEET 16 interpreter to translate the code into legal 6502 operations. It ran slower than standard 6502 code, but when memory space was at a premium it was better to have a slow program than to not have enough room for the program at all.

For those who are keeping count, there are a few unreferenced bytes in the latter part of the Integer ROM. Those bytes contained filler bytes that were not used as any program code.<9>,<10>,<11>

The last part of the Apple II memory, from $F800-$FFFF, contained Wozniak's Monitor program which has already been discussed above.

++++++++++++++++++++++++++++++++++++

NEXT INSTALLMENT: The Apple II, cont.

++++++++++++++++++++++++++++++++++++

NOTES

<1> Jack Connick, "...And Then There Was Apple", CALL-A.P.P.L.E., Oct 1986, p. 24.

<2> -----, "Memory Organization", APPLE II REFERENCE MANUAL, 1979, 1981, pp. 70-73.

<3> Val J. Golding, "Applesoft From Bottom To Top", CALL-A.P.P.L.E. IN DEPTH #1, 1981, p. 8.

<4> Michael Moritz, THE LITTLE KINGDOM, p. 157.

<5> Steven Levy, HACKERS: HEROES OF THE COMPUTER REVOLUTION, pp. 260-261.

<6> Steve Wozniak and Allen Baum, "A 6502 Disassembler From Apple", Dr. Dobb's Journal of Computer Calisthenics & Orthodontia, Sep 1976, pp. 22-25.

<7> Jack Connick, p. 23.

<8> Christopher Volpe, "Beep: A Tale of (T)ERROR", CALL-A.P.P.L.E., Mar 1983, p. 114.

<9> Bob Bragner, "Open Discussion", SOFTALK, Nov 1983, pp. 51-52.

<10> -----, PROGRAMMER'S AID #1, 1978.

<11> Dick Sedgewick, "SWEET 16 - Introduction", MERLIN USER'S MANUAL, 1982, pp. 103-109.

APPLE II HISTORY

Compiled and written by Steven Weyhrich

(PART 4 -- THE APPLE II, CONT.)
[v1.1 :: 12 Dec 91]

THE APPLE II: OTHER DESIGN FEATURES

Since Steve Wozniak was the designer of the Apple I and II, exactly what contribution did Steve Jobs make to the effort? Unlike Wozniak, who would not think much of extra wires hanging out of a computer that worked properly, Jobs had an eye for the appearance of the final product. He wanted the Apple II to be a product that people outside the Homebrew Computer Club would want to own:

> "Jobs thought the cigar boxes [housing the home-made computers] that sat on the ... desk tops during Homebrew meetings were as elegant as fly traps. The angular, blue and black sheet-metal case that housed Processor Technology's Sol struck him as clumsy and industrial ... A plastic case was generally considered a needless expense compared to the cheaper and more pliable sheet metal. Hobbyists, so the arguments went, didn't care as much for appearance as they did for substance. Jobs wanted to model the case for the Apple after those Hewlett-Packard used for its calculators. He admired their sleek, fresh lines, their hardy finish, and the way they looked at home on a table or desk."<1>

The final case design made the Apple II look quite different from most of their competition. The other computers looked like they had been assembled at home (and many of them were). The Apple had no visible screws or bolts (the ten screws attached at the bottom). It had the appearance of some variation of a typewriter, but still looked futuristic enough to be a computer. The friendliness of the design even extended to the lid, which popped off easily to allow access to the expansion slots, almost inviting the user to look inside (unlike most electronic devices that held the warning "CAUTION! NO USER SERVICEABLE PARTS INSIDE").<2>

Other aesthetics to which Jobs paid attention were the color of the keyboard, vents for heat dissipation (avoiding the need for a noisy fan), and a shape and color that would blend in with other items in a home or on a desk. He also hired an engineer who was good with analog circuitry (not Wozniak's area of interest) to design a reliable, lightweight power supply that would stay cool. The engineer, Rod Holt, was working at Atari at the time, but was convinced to help Jobs and Wozniak. He developed a new approach (for microcomputers) by taking household current and switching it on and off rapidly, producing a steady current that was safe for the expensive memory chips. The final design of this switching power supply was smaller than a quart carton of milk and was quite reliable. Holt also helped design the television interface for the Apple II.<3>

The new company was racing to have the Apple II ready for the First West Coast Computer Fair in April of 1977. Some last minute bugs had to be eliminated; because of a static electricity problem affecting a sensitive chip, the keyboards went dead every twenty minutes. Chris Espinosa and Randy Wigginton, two high school students who were early employees of Apple, had written programs to demonstrate the computer's color and sound. They were hurriedly working to duplicate these programs on cassette. People at Apple were working to fix blemishes in the computer cases that had returned from the plastics molding company. The name for this new computer was also finalized as "Apple II", following the example of Digital Equipment Company, who had given each newer version of its PDP series a higher number (PDP-1, PDP-6, etc.). They stylized the "II" in the product name by using right and left brackets, and displaying it on the case as "][". The final product bore the mark of each person at Apple:

> "The computer that appeared at the West Coast Computer Faire was not one person's machine. It was the product of collaboration and blended contributions in digital logic design, analog engineering, and aesthetic appeal. The color, the slots, the way in which the memory could be expanded from 4K to 48K bytes, the control of the keyboard and hookup to the cassette recorder, and the BASIC that was stored in the ROM chip--in effect the motherboard--was Wozniak's contribution. Holt had contributed the extremely significant power supply, and Jerry Mannock the case. The engineering advances were officially recognized when, some months later, Wozniak was awarded U.S. Patent #4,136,359 for a microcomputer for use with video display, and Holt was given Patent #4,130,862 for direct current power supply. But behind them all Jobs was poking, prodding, and pushing and it was he, with his seemingly inexhaustible supply of energy, who became the chief arbiter and rejector... [Finally,] the combination of [Mike] Markkula [Apple's first president], Jobs, and the McKenna Agency turned Apple's public bow [at the West Coast Computer Faire] into a coup."<4>

THE APPLE II: PRODUCT INTRODUCTION

As they prepared for the display at the First West Coast Faire, it was decided to create a new corporate logo. The original one, used in sales of the Apple I, was a picture of Isaac Newton sitting under an apple tree, with a phrase from Wordsworth: "Newton...'A Mind Forever Voyaging Through Strange Seas of Thought...Alone.'" Jobs had been concerned that the logo had part of the slow sales of the Apple I, and the Regis McKenna Agency was hired to help in the design of a new one.

> "Rob Janov, a young art director, was assigned to the Apple account and set about designing a corporate logo. Armed with the idea that the computers would be sold to consumers and that their machine was one of the few to offer color, Janov set about drawing still lifes from a bowl of apples ... He gouged a rounded chunk from one side of the Apple, seeing this as a playful comment on the world of bits and bytes but also as a novel

design. To Janov the missing portion 'prevented the apple from looking like a cherry tomato.' He ran six colorful stripes across the Apple, starting with a jaunty sprig of green, and the mixture had a slightly psychedelic tint. The overall result was enticing and warm ..."

"[Steve] Jobs was meticulous about the style and appearance of the logo ... When Janov suggested that the six colors be separated by thin strips to make the reproduction easier, Jobs refused."<5>

For the Faire, Markkula had ordered a smoky, backlit, illuminated plexiglas sign with the new logo. Although Apple had a smaller booth than other companies displaying their products at the Faire, and some of the other microcomputer makers (Processor Technology, IMSAI, and Cromemco) had been in business longer, Apple's booth looked far more professional, thanks to Markkula's sign. Some of the other participants, companies larger than Apple, had done no more than use card tables with signs written in black markers.

Because they had been one of the first to commit themselves to displaying at the Faire, Apple's booth was near the entrance and was visible to everybody entering the convention center. They demonstrated a kaleidoscopic video graphics program (possibly an early version of "BRIAN'S THEME") on a huge Advent display monitor, catching everybody's attention. But, after the Faire its organizer Jim Warren (Homebrew club member and editor of DR. DOBB'S JOURNAL) didn't think that Apple was a strong exhibitor. Byte magazine, in their report of the show, failed to even mention Apple. Despite these early opinions by influential people, over the next few months Apple received about three hundred orders for the Apple II, over a hundred more than the total number of Apple I's sold.<6>

THE APPLE II: COST

Prebuilt systems were also sold by Commodore (the 6502-based PET, for $595), and Radio Shack (the Z80-based TRS-80, for $600). This was quite a bit less than the Apple II's premium price of $1,298 for a 4K computer, a pair of game paddles, and an audio cassette with demo programs. This price did not include a cassette recorder or monitor (which both the PET and TRS-80 did include). The hardware limitations and lack of expandability of those machines, however, offset some of the price difference. Also, one other hardware introduction for the Apple II that happened in mid-1978 set it well ahead of its immediate competitors; we'll get to that shortly.

THE APPLE II: EXPERIENCES OF EARLY USERS

The original manual for the Apple II was sparse. It consisted of thirty photocopied pages, including some handwritten notes from Woz. The cover stated, "simplicity is the ultimate sophistication: introducing Apple][, the personal computer." In early 1978 these original photocopied manuals were replaced with the new "Apple II Technical Reference Manual" (also known as the "Red Book"), and copies were mailed to previous customers. Steve Jobs realized that people often viewed the quality of a product by the quality of its documentation, and so he took pains to get

manuals that were easy to read and had a professional appearance.<7>

Setting up an early Apple II was fairly simple. The lid popped off easily, and one of the first things you would attach was the Sup'r Mod (RF modulator). This was plugged onto two pins sticking up from the back rear of the motherboard, near the video output jack (assuming that you did not also buy a REAL computer monitor). The game paddles were two small black boxes, with a knob on the top attached to a potentiometer (similar to volume controls on a radio) and a tiny black button on the side. These boxes were attached via a narrow cable to a plug that looked (and was) fragile; this plug also went into a small socket in the motherboard. Lastly, you attached your data storage device (the cassette recorder) to the input and output jacks in the back of the computer.

After turning on the Apple II, the first thing to greet you was a screen full of random alphabetic characters and symbols, and possibly some colored blocks (lo-res graphics mode might be turned on). Here you had to press the RESET key in the upper right hand side of the keyboard, which, after releasing the key, would cause a "beep!" and an asterisk to appear in the bottom left-hand corner of the screen. (If the lo-res graphics mode had been on, it would now be off). Next to the asterisk (which was a prompt to show that you were in the Monitor) was a flashing box, the cursor. To get into BASIC, you had to press the "Ctrl" key and the "B" key simultaneously. Now you would see a different prompt, one that looked like a ">".

At this point, you could either begin entering a BASIC program, or try to load one from cassette. To load from cassette was not always easy; it took time to get the right volume and tone settings on the tape player in order to avoid getting the "ERR" or "*** SYNTAX ERR" message. (And if you didn't have much memory, you might get a "*** MEM FULL ERR" message!) When you got it properly loaded, you could type RUN and see what happened. Beyond that, it was more or less up to you to actually find something to DO with your new toy.<8>

THE APPLE II: EARLY HARDWARE ADD-ONS

Aside from the M&R "Sup'r Mod" that allowed early Apple II users to run their computer on their color TV's, some other enterprising hackers designed their own versions of modulators. One used by an early member of an Apple user group in Washington State (Apple Pugetsound Program Library Exchange, or A.P.P.L.E.) was somewhat better shielded than the "Sup'r Mod". It had its own power supply and plugged into the video output jack on the back of the Apple. The "Sup'r Mod" was by far the biggest seller, however.<9>

At first, there were no interface cards for any of Woz's eight slots. With the limited funds that computer purchasers had then (and now) there was not much they could afford after shelling out anywhere from $1200 to $1800 just to get their own Apple II. But they were innovative, and like many other hardware hackers of the day managed to make do with old or surplus parts. Some people, for instance, had gotten their hands on used teletype printers, such as the ASR-33 (called "battleships" because they were so rugged and heavy). Since there weren't any printer interface cards to plug into the slots to allow the computer to communicate with the teletype, they used a trick they learned from Woz himself. The Apple II had four single-bit output pins on the game controller socket that could be used for various purposes. A schematic floated through the various user

groups that showed how to connect the teletype to an annunciator pin; along with it was a machine language program that re-directed output from the screen to that one-bit port, and on to the printer.<10>

++++++++++++++++++++++++++++++

NEXT INSTALLMENT: The Disk II

++++++++++++++++++++++++++++++

NOTES

<1> Michael Moritz, THE LITTLE KINGDOM, p. 186.

<2> Steven Levy, HACKERS: HEROES OF THE COMPUTER REVOLUTION, pp. 263-264.

<3> Moritz, p. 189.

<4> Moritz, pp. 190-191.

<5> Moritz, p. 188.

<6> Moritz, pp. 192-193.

<7> Philip Chien, "The First Ten Years: A Look Back", THE APPLE II REVIEW, Fall/Winter 1986, p. 12.

<8> -----, APPLE II BASIC PROGRAMMING MANUAL, 1978, 1979, 1980, 1981, pp. 1-19.

<9> -----, "A.P.P.L.E. Co-op Celebrates A Decade of Service", CALL-A.P.P.L.E., Feb 1988, pp. 12-27.

<10> Val J. Golding, "Applesoft From Bottom To Top", CALL-A.P.P.L.E. IN DEPTH #1, 1981, p. 8.

APPLE II HISTORY

Compiled and written by Steven Weyhrich

(PART 5 -- THE DISK II)
[v1.1 :: 12 Dec 91]

THE DISK II

Let's put some more trash into Mr. Fusion to fuel the next leg of our trip. How about one of those KIM-1 computers over there in the corner of the Computer Faire auditorium? We might have to break it up a bit to make it fit ... Okay, now we'll just make a small jump, to December of 1977. By this time the Apple II had been generally available for about six months. Most customers used their television as an inexpensive color monitor, and used a cassette recorder to store and retrieve their programs and data. Apple's major competitors were the TRS-80 and the Commodore PET. The products made by these two companies, together with Apple, could be considered as the second generation of microcomputers; they all came fully assembled and ready to use out of the box, with a keyboard and cassette interface. The TRS-80 and the PET even came with a monitors and cassette recorders. The strength of the Apple was expandability and graphics, while the strength of the others was cost (both the TRS-80 and the PET sold for around $600, half the price of the Apple II).

By late 1977, Apple had introduced some enhancements to the II, including their first version of a floating point BASIC (called "Applesoft") on cassette, and a printer interface card to plug into one of the slots on the motherboard. But the Apple II still needed something to make it more attractive to buyers, to stand out above the TRS-80 and the PET. One area that needed improvement was its program and data storage and retrieval system on cassette; it was a continued source of frustration for many users. The cassette system used on the TRS-80 was more sophisticated than that of the Apple II, allowing named files and easier storage of files and data on the same tape. On the Apple II it took VERY careful adjustment of the volume and tone controls on the cassette recorder to get programs or data to successfully load. The Apple cassette system also needed careful attention to the location on the tape where a program was stored, and was no more accurate than the number on the recorder's mechanical tape counter (if it had one).

Apple president Mike Markkula was one Apple II user that was dissatisfied with cassette tape storage. He had a favorite checkbook program, but it took two minutes to read in the program from the tape, and another two minutes to read in the check files.<1> Consequently, at the executive board meeting held in December 1977 he made a list of company goals. At the top of the list was "floppy disk". Although Wozniak didn't know much about how floppy disks worked, he had once looked through a manual from Shugart (a Silicon Valley disk drive manufacturer):

> "As an experiment Woz had [earlier] conceived a circuit that would do much of what the Shugart manual said was needed to

control a disk drive. Woz didn't know how computers actually controlled drives, but his method had seemed to him particularly simple and clever. When Markkula challenged him to put a disk drive on the Apple, he recalled that circuit and began considering its feasibility. He looked at the way other computer companies--including IBM--controlled drives. He also began to examine disk drives--particularly North Star's. After reading the North Star manual, Woz knew that his circuit would do what theirs did and more. He knew he really had a clever design."<2>

Other issues that Wozniak had to deal with involved a way to properly time the reading and writing of information to the disk. IBM used a complex hardware-based circuit to achieve this synchronization. Wozniak, after studying how IBM's drive worked, realized that if the data was written to the disk in a different fashion, all that circuitry was unneeded. Many floppy disks sold at that time were "hard sectored", meaning that they had a hole punched in the disk near the center ring. This hole was used by the disk drive hardware to identify what section of the disk was passing under the read/write head at any particular time. Wozniak's technique would allow the drive to do self-synchronization ("soft sectoring"), not have to deal with that little timing hole, and save on hardware.

Wozniak asked Randy Wigginton for help in writing some software to control the disk drive. During their week of Christmas vacation in 1977 they worked day and night creating a rudimentary disk operating system, working hard to get the drive ready to demonstrate at the Consumer Electronics Show in the first week of 1978. Their system was to allow entry of single letter commands to read files from fixed locations on the disk. However, even this simple system was not working when Wozniak and Wigginton left for the show.

When they got to Las Vegas they helped to set up the booth, and then returned to working on the disk drive. They stayed up all night, and by six in the morning they had a functioning demonstration disk. Randy suggested making a copy of the disk, so they would have a backup if something went wrong. They copied the disk, track by track. When they were done, they found that they had copied the blank disk on top of their working demo! By 7:30 am they had recovered the lost information and went on to display the new disk drive at the show.<3>,<4>

Following the Consumer Electronics Show, Wozniak set out to complete the design of the Disk II. For two weeks, he worked late each night to make a satisfactory design. When he was finished, he found that if he moved a connector he could cut down on feedthroughs, making the board more reliable. To make that move, however, he had to start over in his design. This time it only took twenty hours. He then saw another feedthrough that could be eliminated, and again started over on his design. "The final design was generally recognized by computer engineers as brilliant and was by engineering aesthetics beautiful. Woz later said, 'It's something you can ONLY do if you're the engineer and the PC board layout person yourself. That was an artistic layout. The board has virtually no feedthroughs.'"<5>

THE DISK II: COST

The Disk II was finally available in July 1978 with the first full

version of DOS, 3.1. It had an introductory price of $495 (including the controller card) if you ordered them before Apple had them in stock; otherwise, the price would be $595. Even at that price, however, it was the least expensive floppy disk drive ever sold by a computer company. Early production at Apple was handled by only two people, and they produced about thirty drives a day.<6>,<7>

Apple bought the drives to sell with Woz's disk controller from Shugart, right there in Silicon Valley. To cut costs, however, they decided to go to Alps Electric Company of Japan and ask them to design a less expensive clone. According to Frank Rose, in his book "West Of Eden":

> "The resulting product, the Disk II, was almost obscenely profitable: For about $140 in parts ($80 after the shift to Alps) [not counting labor costs], Apple could package a disk drive and a disk controller in a single box that sold at retail for upwards of $495. Better yet was the impact the Disk II had on computer sales, for it suddenly transformed the Apple II from a gadget only hard-core hobbyists would want to something all sorts of people could use. Few outsiders realized it, but in strategic terms, Woz's invention of the disk controller was as important to the company as his invention of the computer itself."<8>

++++++++++++++++++++++++++++++++++

NEXT INSTALLMENT: The Apple II Plus

++++++++++++++++++++++++++++++++++

NOTES

<1> Gregg Williams and Rob Moore, "The Apple Story, Part 2: More History And The Apple III", BYTE, Jan 1985, pp. 167-168.

<2> Paul Freiberger and Michael Swaine, "Fire In The Valley, Part Two (Book Excerpt)", A+ MAGAZINE, Jan 1985, p. 45.

<3> Williams and Moore, "Part II", p. 168.

<4> Freiberger and Swaine, (Part Two), p. 45.

<5> Freiberger and Swaine, (Part Two), p. 46.

<6> -----, "A.P.P.L.E. Co-op Celebrates A Decade of Service", CALL-A.P.P.L.E., Feb 1988, pp. 12-27.

<7> -----, "Apple and Apple II History", THE APPLE II GUIDE, Fall 1990, pp. 9-16.

<8> Frank Rose, WEST OF EDEN: THE END OF INNOCENCE AT APPLE COMPUTER, 1989, pp. 62.

APPLE II HISTORY

Compiled and written by Steven Weyhrich

(PART 6 -- THE APPLE II PLUS)
[v1.1 :: 12 Dec 91]

THE APPLE II PLUS: HARDWARE

We now go cruising ahead in time about one year, to June of 1979. Applesoft BASIC had been in heavy demand since the introduction in late 1978 of an improved version. It was needed by those wanting to write and use applications that needed the capability of floating-point math. Because of this, Apple engineers had begun working in 1978 on the Apple II Plus, a modest enhancement to the Apple II. The main attraction of this newer Apple would be Applesoft in ROM, available immediately without having to load it from cassette or disk. Also, having it in ROM would move it out of the part of memory where RAM Applesoft conflicted with hi-res graphics (after all, Applesoft had commands specifically written into it for manipulating those graphics, something that Integer BASIC could only do via special CALLs to the routines in the Programmer's Aid #1 chip).

With the decision made to upgrade the Apple II, other changes were made to make it more attractive to new computer buyers. The cost of RAM chips had dropped considerably, so most new II Plus systems came standard with a full 48K of RAM. Since the disk operating system consumed about 10K of memory, having the full complement of available RAM made it easier to use the Disk II with either version of BASIC. Since users would not need to add the smaller 4K memory chips, the strapping blocks that had made it possible to use either 4K or 16K RAM chips on the original Apple II were removed.

Small changes had already been made to the product since it first began distribution. Most of these changes were made primarily to simplify it and decrease costs of manufacturing. First of all, the original Apple II motherboard, designated as "Revision 0", was changed to make it possible to display two more colors in hi-res graphics. The Revision 0 board had only four colors (green, violet, black, white), but Wozniak had learned that by making a simple alteration he could get two more colors (blue and orange) and two more varieties of black and white. The Revision 1 and later boards were capable of displaying all eight colors. The means of making this modification to Revision 0 Apples was described by Wozniak in his reply to an article by Allen Watson III about hi-res graphics (in the June 1979 issue of Byte magazine). With that change, people who were not afraid of doing a little electrical work on their computers had some of the benefits of an updated Apple II.

Hardware bugs that Apple engineers fixed included one that caused text characters to be displayed with green and violet fringing, whether in graphics mode or text mode. The "color killer" circuit they added fixed things so that non-graphics text would display in black and white only. Another problem involved RAM configurations of either 20K or 24K (a 16K RAM chip plus one or two 4K RAM chips). In those systems a hardware bug caused the 8K of memory from $4000 to $5FFF to be duplicated in the next 8K of

memory, from $6000 to $7FFF, whether there was RAM present at those locations or not. This made a 20K Apple appear to have 24K, and a 24K Apple appear to have 36K. The Revision 1 motherboard fixed this problem as well.<1>

Revision 1 boards also modified the cassette input circuit to respond with more accuracy to a weak input signal, making it easier to load data and programs from cassette. Also, one "feature" of the original Apple II was that any sound generated by the internal speaker also appeared as a signal on the cassette output connector; this was fixed in the new motherboards. Lastly, the RESET cycle was made part of the power-up circuitry, eliminating the requirement that the RESET key be pressed after turning on the computer.<2>,<3>

THE APPLE II PLUS: FIRMWARE

More important than the minor hardware changes, however, were the changes in the ROM code. The new ROM replaced the original Monitor with one that, among other things, better supported the new Disk II drive. Since RESET was now automatically activated when the power was turned on, the new ROM code had the computer automatically do a few things. It cleared the screen (displaying "APPLE][" at the top), and began a scan down the slots, starting at slot 7 down to slot 1. It examined the first few bytes of code in each card's ROM for a specific sequence that identified it as a Disk II controller card. If one was found, control was passed to that card, causing the disk drive to startup and begin loading the disk operating system into memory. If no disk controller was found, the ROM code jumped instead to the start of BASIC (instead of leaving the user in the Monitor, as in the old ROM). This "Autostart ROM", as it was called, made it possible to have a system that started up a program on the disk with little action needed by the user.

The RESET code was more intelligent in the Autostart ROM than in the Old Monitor ROM. There was now a "Cold Start" RESET (which functioned as described above), and a "Warm Start" RESET. A Warm Start RESET could occur without re-booting the Disk II (if it was present); in fact, it ensured that the disk operating system remained "connected" after RESET was pressed. This feature was implemented by setting three bytes at the end of page $03 in memory. Two of the bytes were the address of the place in memory to which the Apple should jump if RESET was pressed. The third byte was a specially coded byte created from half of the address byte. When RESET was pressed, this special "power-up" byte was checked with the address byte. If they didn't properly match, the Monitor assumed that the power had just been turned on, and it executed a Cold Start RESET. This feature was extensively used by writers of copy protected software, so users could not modify or copy the code in memory simply by pressing the RESET key.

The other major change, mentioned earlier, was the BASIC that was supplied in ROM. Gone was Steve Wozniak's hand-assembled Integer BASIC, in favor of the newer Applesoft. Since these ROM versions of BASIC used the same memory locations, they could not be used simultaneously. With the introduction of the II Plus, Apple also released the Applesoft Firmware card. This card, which plugged into slot 0, made it possible for previous Apple II owners to have some of the benefits of the II Plus without having to buy an entirely new computer. Even with that card, however, you could not use features of one BASIC while the other was active, and switching

from one BASIC to the other erased any program that was being used at the time. The two BASICs could be told apart by the prompt they used; Integer BASIC used the ">" character, but Applesoft used the "]" character.

Another change made to the Monitor ROM made screen editing easier. The original Apple II's procedure for editing a line typed in BASIC or in the Monitor was tedious at best. To change a line of text in BASIC, you had to list the line, move the cursor up to the start of the line, and then use the right-arrow key to "copy" text from the screen into the input buffer. If you wanted to skip part of the line, you had to move the cursor past the text that you wanted to eliminate WITHOUT using the arrow keys. If you wanted to INSERT something into the line, you had to move the cursor off the line (above it or below it), type the additional text, and then move the cursor back into the line to finish copying the original part of the line.

For example, suppose you had typed this line in Applesoft and displayed it on the 40-column screen:

```
]LIST 100

100  FOR I = 1 TO 100: PRINT "I
    LIKE MY APPLE": NEXT : END
```

To change that line so the PRINT statement read "I REALLY LIKE MY APPLE" meant either retyping the entire line, or using the edit feature. (If the line was particularly long, it was preferable to edit rather than retype the entire line). To edit this line, you would have to move the cursor up to the "1" of "100" and begin pressing the right arrow key. When you got to the "L" of "LIKE" you would have to move the cursor above or below the line, type the word "REALLY" followed by a space, then move the cursor back to the "L" of "LIKE", and continue copying with the right arrow key. After editing a line, the screen might look like this:

```
100  FOR I = 1 TO 100: PRINT "I
    LIKE MY APPLE": NEXT : END
     REALLY
```

(In this example, I moved the cursor down one line, typed "REALLY", and then moved it back to the start of the word "LIKE"). If you didn't make any mistakes it would read like this:

```
]LIST 100

100  FOR I = 1 TO 100: PRINT "I
    REALLY LIKE MY APPLE" : NEXT
    : END
```

However, if you didn't take care to skip over the extra spaces inserted in front of the word "LIKE" by the Applesoft LIST command, it could appear this way:

```
100  FOR I = 1 TO 100: PRINT "I
                REALLY LIKE MY APPLE"
    : NEXT : END
```

The big problem with these cursor moves for editing under the Old Monitor was that each move required two keypresses. To move the cursor up, you had to press "ESC" and then "D" EACH TIME you wanted to move the cursor up. "ESC A" moved right, "ESC B" moved left, and "ESC C" moved the cursor down. With a long line that needed much editing, this would get old real fast. Not only was it cumbersome, but the layout of the keyboard made it difficult to remember the correct letters used for cursor movement; although "D" (up) was above "C" (down), it seemed that "D" should stand for "Down". Also confusing was that "A" was to the left of "B", but their functions were the opposite of their position!

The new Autostart ROM improved this screen editing process just a bit. Now, pressing "ESC" turned on a special editing mode. Repeated presses of "I" (up), "J" (left), "K" (right), and "M" (down) continued to move the cursor until a key other than ESC was pressed. On the keyboard these letters were arranged in a sort of "directional keypad" or diamond, which made remembering the moves a little easier. The previous ESC editing codes were still supported, but still with their previous limitations. Unfortunately, however, you still couldn't tell whether you were in the regular text entry mode or in the ESC editing mode, and often attempts at changing a line took several tries to get it right.<4>,<5>

Other features added in the new Autostart ROM included the ability to pause a listing by pressing Ctrl-S (VERY helpful when trying to scan through a long program!) As mentioned above, pressing RESET would return control through the soft-entry vectors on memory page $03. This would allow a user to exit from a runaway BASIC program by pressing RESET, and still keep program and variables intact in memory (which could not be guaranteed with the old Monitor ROM).<5>

John Arkley at Apple wrote the changes to the original Monitor ROM and created the Autostart ROM in November 1978 (he's the "John A" mentioned in the source code listing found in the 1981 edition of the APPLE II REFERENCE MANUAL). After he had done the work and the ROMs had been created, Apple wanted to publish a new version of the Reference Manual to cover the Apple II Plus. The older Reference Manual (affectionately known as the "Red Book") had included an assembly language source code listing of the Monitor ROM. They wanted to include the source for BOTH versions of the Monitor, but a problem came up. While developing the Monitor, Apple had used a local mainframe computer dial-up service known as "Call Computer." They used a cross-assembler on that computer, assembled the code, and then used the resulting object code to create the ROM. (A cross-assembler is an assembler that creates object code for a processor other than the one the cross-assembler runs on. For example, if you can write 8080 machine code with an assembler running on a 6502-based computer, you are using a cross-assembler). Unfortunately, Call Computer had accidentally done a system backup with the source and destination disks reversed, erasing all the files containing the source code for the Apple II Monitors. There were no disk or cassette copies of the source code for the Autostart ROM back at Apple. Working from the source listing in the Red Book, John recreated the source file for the original Monitor, and then

disassembled his own modifications for the II Plus and re-created his Autostart ROM source file. Those reconstructed listings are what appeared in the 1981 edition of the Apple II Reference Manual.<6>

Not everyone was pleased with the modifications made in the Autostart ROMs, however. Some of the authors of the magazine CALL-A.P.P.L.E. liked to refer to the new computer as the "Apple II Minus", since Arkley had to remove some of their beloved routines from the ROMs to make room for the new features. Missing from the Apple II Plus ROMs were Integer BASIC, the miniassembler, and Woz's SWEET 16 interpreter (that entire space now being used by Applesoft). Missing from the Monitor were the assembly language STEP and TRACE features, and a set of sixteen-bit multiply and divide routines.<5>

THE APPLE II PLUS: COST

The new Apple II Plus, at $1,195, sold for over $100 less than the original Apple II, although it came with more memory and had Applesoft (previously an added expense item) in ROM.

THE APPLE II PLUS: BELL & HOWELL

Apple made a deal early on with Bell & Howell to let them sell the Apple II Plus with a Bell & Howell name plate on it for use in schools. These Apples were black colored (instead of the standard beige), and had screws on the back to keep the lids on (apparently to keep students' hands out). These Apples (sometimes called "Darth Vader" Apples) also had the "shift-key mod" (see below) applied. Since Bell & Howell was a major supplier of school equipment, this was a means for Apple to get a foothold in the school environment.<7>,<8>

Bell & Howell also had electronics correspondence courses, and used the black Apple II Plus for one of their courses. They offered a one year warranty, instead of the ninety-day warranty offered by Apple.<9>,<10>,<11>

THE APPLE II PLUS: EARLY USER EXPERIENCES

An Apple II veteran on GEnie, Dennis Ulm, kindly provided me with the following reproduction of his ORIGINAL Apple II Plus packing list. It gives a little picture of what early non-disk users had to work with:

APPLE II PLUS

PACKING LIST

This package should contain the following items:

item	no.	part number	description
1	1	600-2023	cassette tape: LITTLE BRICKOUT, COLOR DEMOSOFT
2	1	600-2024	cassette tape: RENUMBER/APPEND, ALIGNMENT TEST TONE
3	1	600-2025	cassette tape: FINANCE I, PENNY ARCADE

4	1	600-2026	cassette tape: LEMONADE, HOPALONG CASSIDY
5	1	600-2027	cassette tape: BRIAN'S THEME, PHONE LIST
6	1	030-2057	manual: Introductory Programs for the Apple II Plus
7	1	030-0044	manual: The Applesoft Tutorial
8	1	030-0013	manual: Applesoft II BASIC Programming Reference Manual
9	1	030-0004	manual: Apple II Reference Manual
10	1	030-0035	publication: Apple Magazine
11	1	600-0033	1 pair of game controls
12	1	590-0002	cable: to hook up a cassette recorder
13	1	590-0003	cable: power cord for the Apple II Plus
14	1	030-0001	Apple Warranty Card
15	1	600-0816	Apple II Plus System 16K
			or
		600-0832	Apple II Plus System 32K
			or
		600-0848	Apple II Plus System 48K

(LITTLE BRICKOUT was an abbreviated Applesoft version of Woz's Integer BASIC Breakout game (the reason he designed the Apple II in the first place). BRIAN'S THEME was a hi-res graphics program that drew lines on the screen in various patterns. HOPALONG CASSIDY was a "guess who" program that also used the hi-res screen).<12>,<13>

Also included in Dennis' II Plus box was this photocopied instruction sheet:

TAPE LOADING INSTRUCTIONS

If problems are encountered in LOADing tape programs, it may be necessary to "queue" (sic) the tape before LOADing. To queue a tape, use the following procedure:

1. Rewind the tape.

2. Disconnect the cable from the tape recorder (so you can hear what's on the tape).

3. Start the tape recorder in PLAY mode.

4. When a steady tone is heard, STOP the tape recorder.

5. Connect the cable to the tape recorder and adjust the volume and tone controls on the tape recorder to the recommended levels.

6. Make sure your computer is in BASIC.

7. Type LOAD.

8. START the tape playing.

9. Press RETURN.

The program should LOAD properly. If an error message occurs,

repeat the procedure, but try readjusting the tone and volume controls on the tape recorder.

Dennis says that in his experience it took at least five to ten tries to get anything to load properly from tape!

THE APPLE II PLUS: MORE HARDWARE ADD-ONS

Lower-case was still not supported on the new Apple II Plus, though it was a popular user-modification. The thriving industry for Apple II peripherals made up for this shortcoming, with various vendors supplying small plug-in circuit boards that fit under the keyboard, allowing display of lower-case on the screen (and sometimes direct entry of lower-case from the keyboard). By 1981, when the Revision 7 motherboard was released for the Apple II Plus, a different method of character generation was used, which reduced radio-frequency interference that was generated. For Revision 7 boards, lower-case characters could be displayed with the addition of only a single chip. However, unless a user changed the keyboard encoder with a third-party product, only upper-case characters could be typed.<14>

The keyboard itself underwent some changes, both by users and by Apple. The original RESET key was in the upper right-hand corner of the keyboard. The problem with that key was that it had the same feel as the keys around it, making it possible to accidentally hit RESET and lose the entire program that was being so carefully entered. One user modification was to pop off the RESET keycap and put a rubber washer under it, making it necessary to apply more pressure than usual to do a RESET. Apple fixed this twice, once by replacing the spring under the keycap with a stiffer one, and finally by making it necessary to press the CTRL key and the RESET together to make a RESET cycle happen. The keyboards that had the CTRL-RESET feature made it user selectable via a small slide switch just inside the case (some people didn't want to have to press the CTRL key to do a RESET).

Another keyboard limitation was addressed through a modification that became known as the "shift-key mod". This was such a widely used trick that Apple ended up supporting it in hardware when they designed the Apple IIe. Since the II and II Plus keyboards could not directly generate lower-case characters, early word processing programs had to find some way to make up for that deficiency. Apple's own Apple Writer program used the ESC key as a shift and shift-lock key, displaying upper-case characters in inverse video and lower-case in regular video. Other programs suggested installing the shift-key mod to allow more natural entry of upper-case, using the SHIFT key already present on the keyboard. The user had to attach a wire to the contact under the SHIFT key, and run it to the game port where the input for push-button 2 was found. (This push-button PB2, $C063 in memory, was for one of an optional second pair of game paddles that third-party hardware companies supplied for the Apple II). The program would assume that all letters being typed were in lower-case, unless the SHIFT key (attached now to paddle button PB2) was also being pressed; in that case the letter would be entered as upper-case. Since the PB2 button was not often used for a second pair of game paddles, it was unlikely that this modification would be accidentally triggered by pressing one of the game paddle buttons. This modification did NOT use buttons PB0

or PB1, which were on the first pair of game paddles. (PB0 and PB1 now correspond to the Open-Apple and Solid-Apple/Option keys on modern Apple II computers).

++++++++++++++++++++++++++++++

NEXT INSTALLMENT: The Apple IIe

++++++++++++++++++++++++++++++

NOTES

<1> -----, "Memory Organization", APPLE II REFERENCE MANUAL, 1979, 1981, pp. 70-73.

<2> -----, APPLE II REFERENCE MANUAL, 1979,1981, pp. 25-27, 34-36.

<3> Bruce Field, "A.P.P.L.E. Doctor", CALL-A.P.P.L.E., Jan 1984, pp. 74-75.

<4> -----, "Apple and Apple II History", THE APPLE II GUIDE, Fall 1990, pp. 9-16.

<5> -----, APPLE II REFERENCE MANUAL, 1979,1981, pp. 25-27, 34-36.

<6> John Arkley, (personal telephone call), Sep 9, 1991.

<7> Joe Regan, GEnie A2 ROUNDTABLE, Category 2, Topic 16, Apr 1991.

<8> Dan Paymar, "Curing A Shiftless Apple", CALL-A.P.P.L.E., May 1982, pp. 63-64.

<9> Tom Vanderpool, GEnie A2 ROUNDTABLE, Category 2, Topic 16, Mar & Aug 1991.

<10> Tom Zuchowski, GEnie A2 ROUNDTABLE, Category 2, Topic 16, Mar 1991.

<11> Steve Hirsch, GEnie A2 ROUNDTABLE, Category 2, Topic 16, Mar 1991.

<12> Dennis Ulm, GEnie A2 ROUNDTABLE, Category 2, Topic 16, Apr 1991.

<13> Wes Felty, GEnie. A2 ROUNDTABLE, Category 2, Topic 16, Apr 1991.

<14> Bruce Field, "A.P.P.L.E. Doctor", CALL-A.P.P.L.E., Jan 1984, pp. 74-75.

APPLE II HISTORY

Compiled and written by Steven Weyhrich

(PART 7 -- THE APPLE IIE)
[v1.1 :: 26 Jan 92]

PRELUDE: THE APPLE III PROJECT

As we continue our travels examining the history of the Apple II, let's fine tune the time-machine card on our souped-up Apple II to concentrate specifically on the next version of the II, the IIe. As before, just accelerate the microprocessor speed to 88 MHz, and watch out for the digital fire-trails! Destination: 1982.

Between the years 1979 and 1983, although no new versions of the Apple II were released, it enjoyed a broad popularity and annually increasing sales. The open architecture of the computer, with its fully described hardware and firmware function via the Reference Manual, made it appealing both to hardware and software hackers. Third-party companies designed cards to plug into the internal slots, and their function varied from making it possible to display and use 80-column text, to clocks and cards allowing the Apple II to control a variety of external devices. During this time there was also an explosion of new software written for this easily expandable machine, from the realm of business (VisiCalc and other spreadsheet clones), to utilities, to games of all types. Each month a host of new products would be available for those who wanted to find more things to do with their computer, and the Apple II was finding a place in the home, the classroom, and the office.

At Apple Computer, Inc., however, the Apple II was not viewed with the same degree of loyalty. By September 1979 the Apple II had continued to be a sales leader. However, few at Apple believed that the II could continue to be a best seller for more than another year or two. Since Apple Computer, Inc. was a business, and not just a vehicle for selling the Apple II computer, they began to enlarge the engineering department to begin designing new products.<1> These new design efforts had begun as far back as late 1978. Their first effort was an enhanced Apple II that used some custom chips, but that project was never finished. They also began work on a different, more powerful computer that would use several identical microprocessor chips sharing tasks. The main advantage would be speed, and the ability to do high precision calculations. This computer was code-named Lisa, and because it was such a revolutionary type of design, they knew it would take many years to come to actual production. Because of the power it was to have, Apple executives felt that Lisa was the future of the company.<2>,<14>

Because they knew that the Lisa project would take a long time to complete, and because the Apple II was perceived to have only a short remaining useful life as a product, they began a new computer project called the Apple III. Instead of building upon the Apple II as a basis for this new computer, they decided to start from scratch. Also, although

Wozniak made most of the design decisions for the II, a committee at Apple decided what capabilities the Apple III should have. They decided that the Apple III was to be a business machine, and not have the home or arcade-game reputation that the II had. It was to have a full upper/lowercase keyboard and display, 80-column text, and a more comprehensive operating system. They also decided that since it would be a while before many application programs would be available for this new computer, it should be capable of running existing Apple II software. In some ways this handicapped the project, since it was then necessary to use the same microprocessor and disk drive hardware as was used in the Apple II.<3>

Apple executives also decided that with the introduction of the Apple III they wanted a clear separation between it and the Apple II in regards to marketing. They did not want ANY overlap between the two. The III would be an 80-column business machine and was predicted to have ninety percent of the market, while the Apple II would be a 40-column home and school machine and would have ten percent of the market. Apple's executives were confident that after the release of the Apple III, the Apple II would quickly lose its appeal.<4>

Because of their desire for a strong and distinct product separation, the Apple II emulation mode designed into the Apple III was very limited. The engineers actually ADDED hardware chips that prevented access to the III's more advanced features from Apple II emulation mode. Apple II emulation couldn't use 80 columns, and had access to only 48K memory and none of the better graphics modes. As a result, it wouldn't run some of the better Apple II business software, during a time when there wasn't much NEW business software for the Apple III.

The Apple III engineers were given a one year target date for completion. It was ready for release in the spring of 1980, but there were problems with both design and manufacturing. (It was the first time that Apple as a company tried to come out with a new product; the Apple II had been designed and built by Wozniak when he WAS the engineering department). The first Apple III computers were plagued with nearly 100% defects and had to be recalled for fixes. Although Apple took the unprecedented step of repairing all of the defective computers at no charge, they never recovered the momentum they lost with that first misstep, and the III did not become the success Apple needed it to be.<3>

Although all of the bugs and limitations of the Apple III were eventually overcome, and it became the computer of choice within Apple, it did not capture the market as they had hoped. At that point, they weren't sure exactly what to do with the II. They had purposely ignored and downplayed it for the four years since the II Plus was released, although without its continued strong sales they would not have lasted as a company. In a 1985 interview in Byte magazine, Steve Wozniak stated:

> "When we came out with the Apple III, the engineering staff cancelled every Apple II engineering program that was ongoing, in expectation of the Apple III's success. Every single one was cancelled. We really perceived that the Apple II would not last six months. So the company was almost all Apple III people, and we worked for years after that to try and tell the world how good the Apple III was, because we KNEW [how good it was] ... If you looked at our advertising and R&D dollars, everything we did here was done first on the III, if it was business related. Then

> maybe we'd consider doing a sub-version on the II. To make sure there was a good boundary between the two machines, anything done on the II had to be done at a lower level than on the III. Only now are we discovering that good solutions can be implemented on the II ... We made sure the Apple II was not allowed to have a hard disk or more than 128K of memory. At a time when outside companies had very usable schemes for adding up to a megabyte of memory, we came out with a method of adding 64K to an Apple IIe, which was more difficult to use and somewhat limited. We refused to acknowledge any of the good 80-column cards that were in the outside world--only ours, which had a lot of problems."<4>

Wozniak went on in that interview to say that at one time he had written some fast disk routines for the Pascal system on the Apple II, and was criticized by the Apple III engineers. They didn't think that anything on the II should be allowed to run faster than on a III. That was the mindset of the entire company at the time.

Apple has been much maligned for the attention they gave the Apple III project, while suspending all further development on the Apple II. They pegged their chances for the business market in 1980 on the Apple III. Even Steve Wozniak had stated in another interview, "We'd have sold tons of [computers in the business market] if we'd have let the II evolve ... to become a business machine called the III instead of developing a separate, incompatible computer. We could have added the accessories to make it do the business functions that the outside world is going to IBM for."<3> Part of the problem was the immaturity of the entire microcomputer industry at the time. There had NEVER been a microcomputer that had sold well for more than a couple of years before it was replaced by a more powerful model, usually from another company. The Altair 8800 and IMSAI had fallen to the more popular and easier to use Apple II and TRS-80 and Commodore PET, as well as other new machines based on the Intel 8080 and 8088 processors. It is entirely understandable that Apple's attitude between 1978 and 1980 would be of panic and fear that they wouldn't get a new computer out in time to keep their market share and survive as a company. However, during the entire time when Apple was working on the III as a computer to carry the company through until Lisa would be ready, and during the entire time that the Apple II was ignored by its own company, it continued to quietly climb in sales. It is a credit to both the ingenuity of Wozniak in his original design, and to the users of the Apple II in THEIR ingenuity at finding new uses for the II, that its value increased and stimulated yet more new sales. The Apple II "beat" the odds of survival that historically were against it.

THE APPLE IIE: BEGINNINGS

When Apple saw that the sales on the Apple II were NOT going to dwindle away, they finally decided to take another look at it. The first new look at advancing the design of the II was with a project called "Diana" in 1980. Diana was intended primarily to be an Apple II that had fewer internal components, and would be less expensive to build. The project was later known as "LCA", which stood for "Low Cost Apple". Inside Apple this meant a lower cost of manufacturing, but outsiders who got wind of the project thought it meant a $350 Apple II. Because of that

misconception, the final code name for the updated Apple II was "Super II", and lasted until its release.<5>

THE APPLE IIE: HARDWARE

Part of the IIe project grew out of the earlier work on custom integrated circuits for the Apple II. When they finally decided to go ahead and improve the design by adding new features, one of the original plans was to give the Apple II an 80-column text display and a full upper/lowercase keyboard. Walt Broedner at Apple did much of the original hardware planning, and was one of those at Apple who pushed for the upgrade in the first place. To help maintain compatibility with older 40-column software (which often addressed the screen directly for speed), he decided to make 80-columns work by mirroring the older 40 column text screen onto a 1K memory space parallel to it, with the even columns in main memory and the odd columns in this new "auxiliary" memory. To display 80-column text would require switching between the two memory banks. Broedner realized that with little extra effort he could do the same for the entire 64K memory space and get 128K of bank-switchable memory. They put this extra memory (the 1K "80-column card, or a 64K "extended 80-column card") in a special slot called the "auxiliary" slot that replaced slot 0 (the 16K Language Card was going to be a built-in feature). The 80-column firmware routines were mapped to slot 3, since that was a location commonly used by people who bought 80-column cards for their Apple II's, and was also the place where the Apple Pascal system expected to find an external terminal. The auxiliary slot also supplied some special video signals, and was used during manufacture for testing on the motherboard.

The engineers that worked on the IIe tried hard to make sure that cards designed for the II and II Plus would work properly in the new computer. They even had to "tune" the timing on the IIe to be slightly OFF (to act more like the II Plus) because the Microsoft CP/M Softcard refused to function properly with the new hardware. A socket was included on the motherboard for attaching a numeric keypad, a feature that many business users had been adding (with difficulty) to the II Plus for years. The full keyboard they designed was very similar to the one found on the Apple III, including two unique keys that had first appeared with the III--one with a picture of an hollow apple ("open-apple") and the other with the same apple picture filled in ("solid-apple"). These keys were electrically connected to buttons 0 and 1 on the Apple paddles or joystick. They were available to software designers as modifier keys when pressed with another key; for example, open-apple-H could be programmed to call up a "help" screen. The newer electronics of the keyboard also made it easier to manufacture foreign language versions of the Apple IIe.<6>

Overall, Broedner and Peter Quinn (the design manager for the IIe and later the IIc projects) and their team managed to decrease the number of components on the motherboard from over one hundred to thirty-one, while adding to the capabilities of the computer by the equivalent of another hundred components.

THE APPLE IIE: FIRMWARE

Peter Quinn had to beg for someone to help write the firmware revisions to the Monitor and Applesoft for the IIe. He finally got Rich

Auricchio, who had been a hacker on the Apple II almost from the beginning. Quinn said in a later interview, "You cannot get someone to write firmware for this machine unless he's been around for three or four years. You have to know how to get through the mine field [of unofficial but commonly used entry points]. He [Rick] was extremely good. He added in all the 80-column and Escape-key stuff." Quinn also got Bryan Stearns to work on the new Monitor.<6>,<7>

Changes were made in the ROMs to support the new bank-switching modes made necessary by having two parallel 64K banks of RAM memory. To have enough firmware space for these extra features, the engineers increased the size of the available ROM by making IT bank-switched. This space was taken from a location that had previously not been duplicated before--the memory locations used by cards in the slots on the motherboard. Ordinarily, if you use the Monitor to look at the slot 1 memory locations from $C100 through $C1FF, you get either random numbers (if the slot is empty), or the bytes that made up the controller program on that card. Any card could also have the space from $C800 through $CFFF available for extra ROM code if they needed it. If a card in a slot did a read or write to memory location $CFFF, the $C800-$CFFF ROM that belonged to that card would appear in that space in the Apple II memory. When another card was working, then ITS version of that space would appear. On the IIe, they made a special soft-switch that would switch OUT all the peripheral cards from the memory, and switch IN the new expanded ROM on the motherboard. The firmware in the new bank-switched ROM space was designed to avoid being needed by any card in a slot (to avoid conflicts), and much of it was dedicated to making the 80-column display (mapped to slot 3) work properly.

Also added were enhancements to the ESC routines used to do screen editing. In addition to the original ESC A, B, C, and D, and the ESC I, J, K, and M added with the Apple II Plus, Auricchio added the ability to make the ESC cursor moves work with the left and right arrow keys, and the new up and down arrow keys. The new IIe ROM also included a self-test that was activated by pressing both apple keys, the control key, and RESET simultaneously.<5>

THE APPLE IIE: SUCCESS

The new Apple IIe turned out to be quite profitable for Apple. Not only was it more functional than the II Plus for a similar price, but the cost to the dealers selling it was about three times the cost of manufacture. They had gotten their "Low Cost Apple", and by May of 1983 the Apple IIe was selling sixty to seventy thousand units a month, over twice the average sales of the II Plus. Christmas of 1983 saw the IIe continue to sell extremely well, partly resulting from the delayed availability of the new IBM PCjr. Even after the Apple IIc was released in 1984, IIe sales continued beyond those of the IIc, despite the IIc's built-in features.<8>

THE APPLE IIE: MODIFICATIONS

Early Apple IIe motherboard's were labelled as "Revision A". Engineers determined soon after its introduction that if the same use of parallel memory was applied to the hi-res graphics display as was done with the text display, they could create higher density graphics. These

graphics, which they called "double hi-res", also had the capability of displaying a wider range of colors, similar to those available with the original Apple II lo-res graphics. The IIe motherboards with the necessary modifications to display these double hi-res graphics were labelled "Revision B", and a softswitch was assigned to turning on and off the new graphics mode.

Later versions of the IIe motherboards were again called "Revision A" (for some reason), although they HAD been modified for double hi-res graphics. The difference between the two "Revision A" boards was that the latter had most of the chips soldered to the motherboard. An original "Revision A" board that had been changed to an Enhanced IIe was not necessarily able to handle double hi-res, since the change to the Enhanced version involved only a four-chip change to the motherboard, but not the changes to make double hi-res possible.<9>

THE APPLE IIE: THE ENHANCED IIE

This version of the Apple IIe was introduced in March of 1985. It involved changes to make the IIe more closely compatible with the Apple IIc and II Plus. The upgrade consisted of four chips that were swapped in the motherboard: The 65c02 processor, with more assembly language opcodes, replaced the 6502; two more chips with Applesoft and Monitor ROM changes; and the fourth a character generator ROM that included graphics characters (first introduced on the IIc) called "MouseText". The Enhanced IIe ROM changes fixed most of the known problems with the IIe 80-column firmware, and made it possible to enter Applesoft and Monitor commands in lower-case. The older 80-column routines were slower than most software developers wanted, they disabled interrupts for too long a time, and there were problems in making Applesoft work properly with the 80-column routines. These problems were solved with the newer ROMs.

Monitor changes also included a return of the mini-assembler, absent since the days of Integer BASIC. It was activated by entering a "!" command in the Monitor, instead of a jump to a memory location as in the older Apple][. Also added were an "S" command was added to make it possible to search memory for a byte sequence, and the ability to enter ASCII characters directly into memory. However, the "L" command to disassemble 6502 code still did not handle the new 65c02 opcodes as did the IIc disassembler. Interrupt handling was also improved.

Applesoft was fixed to let commands such as GET, HTAB, TAB, SPC, and comma tabbing work properly in 80-column mode.

The new MouseText characters caused a problem for some older programs at first, until they were upgraded; characters previously displayed as inverse upper-case would sometimes display as MouseText instead.<10>,<11>

THE APPLE IIE: THE PLATINUM IIE

This version of the IIe, introduced in January 1987, had a keyboard that was the same as the IIGS keyboard, but the RESET key was moved above the ESC and "1" keys (as on the IIc), and the power light was above the "/" on the included numeric keypad (the internal numeric keypad connector was left in place). The CLEAR key on the keypad generated the same character as the ESC key, but with a hardware modification it could generate a Ctrl-X as it did on the IIGS. The motherboard had 64K RAM in only two chips

(instead of the previous eight), and one ROM chip instead of two. An "extended 80-column card" with 64K extra memory was included in all units sold, and was smaller than previous versions of that memory card.

No ROM changes were made. The old shift-key modification was installed, making it possible for programs to determine if the shift-key was being pressed. However, if using a game controller that actually used the third push-button (where the shift-key mod was internally connected), pressing shift and the third push-button simultaneously causes a short circuit that shuts down the power supply.<12>

THE APPLE IIE: EMULATION CARD ON MACINTOSH LC

In early 1991, Apple introduced a new version of the Apple IIe. This one was designed to be exactly like the 128K Platinum IIe, with the modification that it had a color Macintosh attached to it. This Apple IIe cost only $199, but the required Macintosh peripheral went for about $2,495, which makes the combination the most expensive Apple II ever made. Apple engineers managed to put the function of an entire IIe onto a card smaller than the old Disk II controller card. With version 2.0 of the Apple II interface software, more of the memory allocated to the Macintosh could be used by the IIe (strange way of designing an Apple II!). However, unlike all previous versions of the IIe, there were no hardware-based slots on the IIe card; instead, it used software-based slots that were allocated by moving icons that represent various peripherals into "slots" on the Mac screen. (Oh, yes; it ran some Mac software, too. This was, of course, the Macintosh LC computer with its optional Apple IIe card).

To use 5.25 disks with this Apple IIe, there was a cable that attached to the card. The cable would split into a game connector (for paddles or joystick operation) and a connector that accepted IIc and IIGS style 5.25 drives. The IIe card ran at a "normal" (1 MHz) speed and a "fast" (2 MHz) speed.<13> It had limitations, however. For a 1991 Apple II, it was limited in being unable to be accelerated beyond 2 MHz (a Zip Chip can run a standard IIe at 8 MHz), and the screen response seemed slow, since it was using a software-based Mac text display instead of the hardware-based Apple II character ROM. As a Macintosh it lacked the power and speed of the newer Macintosh II models (which also ran color displays). But if having a Apple II and a Mac in one machine was important, this was the best way to do it.

++++++++++++++++++++++++++++++++

NEXT INSTALLMENT: The Apple IIc

++++++++++++++++++++++++++++++++

NOTES

<1> Freiberger, Paul, and Swaine, Michael. "Fire In The Valley, Part I (Book Excerpt)", A+ Magazine, Jan 1985, p. 45-48.

<2> Freiberger, Paul, and Swaine, Michael. "Fire In The Valley, Part II (Book Excerpt)", A+ Magazine, Jan 1985, p. 46,51.

<3> Rubin, Charles. "The Life & Death & Life Of The Apple II", Personal Computing, Feb 1985, p. 72.

<4> Williams, Gregg, and Moore, Rob. "The Apple Story, Part 2: More History And The Apple III", Byte, Jan 1985, pp. 177-178.

<5> Tommervik, Al. "Apple IIe: The Difference", Softalk, Feb 1983, pp. 118-127, 142.

<6> Williams, Gregg. "'C' Is For Crunch", Byte, Dec 1984, pp. A75-A78, A121.

<7> Little, Gary. Inside The Apple //c, 1985, pp. 1-7.

<8> Rose, Frank. West Of Eden: The End Of Innocence At Apple Computer, 1989, pp. 98-99.

<9> Weishaar, Tom. "Ask Uncle DOS", Open-Apple, Dec 1986, p. 2.86.

<10> Weishaar, Tom. "A Song Continued", Open-Apple, Mar 1985, pp. 1.20-1.21.

<11> Weishaar, Tom. "Demoralized Apple II Division Announces Enhanced IIe...", Open-Apple, Apr 1985, pp. 1.25-1.27.

<12> Weishaar, Tom. "Apple Introduces An Updated IIe", Open-Apple, Jan 1987, p. 3.1.

<13> Doms, Dennis. "The Apple II as Mac peripheral", Open-Apple, Jul 1991, pp. 7.43-7.44.

<14> This was an early version of the Lisa project. When the 68000 microprocessor became available from Motorola, it was decided to use that as a single processor for the Lisa. Also, after Steve Jobs paid a visit to the Xerox lab and saw the Xerox Star computer with its icon interface and mouse pointing device, he pushed strongly for the Lisa to work in that way.

APPLE II HISTORY

Compiled and written by Steven Weyhrich

(PART 8 -- THE APPLE IIC)
[v1.1 :: 12 May 92]

PRELUDE: STEVE JOBS AND MACINTOSH

Rewind back to 1982, before the Apple IIe was introduced, and adjust the tuning on our Flux Capacitor-enhanced peripheral card. Before dealing specifically with the smallest Apple II, the IIc, it would help to take an aside and look at some other events happening at Apple Computers, Inc. at this time that affected its development.

If you recall, the Lisa project was designated as the computer that was considered to be the future of Apple. From a series of parallel processors and a "bit slicing" architecture, to a focus on the Motorola 68000 microprocessor as the controller of this advanced computer, the project had been progressing very slowly. It was begun back in 1979 with the same focus as any other Apple product: "Both [Apple III and Lisa] had been conceived of as nifty pieces of hardware rather than as products to appeal to a specific market: At Apple you designed a box and people bought it because it was neat, not because any thought had been given to what it would do for them."<1> However, a significant change occurred in 1979 when Xerox bought a large chunk of Apple stock. In return for being allowed this stock purchase, Xerox allowed some of their research ideas to be used in designing an office computer. After Steve Jobs visited the Xerox Palo Alto Research Center in 1979 and saw the user-interface on their Alto computer--icons, graphics-based text characters, overlapping windows, and a pointing device called a "mouse"--the Lisa took on a distinct personality that made it possible to become the ultra-computer Apple needed. This was important, since by 1981 Apple executives were getting sweaty palms worrying about the future. The Apple III was clearly NOT taking the business world by storm.

Unfortunately for Jobs, who was excited about using the Xerox technology in designing a new computer, he was excluded from the Lisa project. After the problems associated with the introduction of the Apple III, a reorganization in 1980 moved the Apple II and III into one division, and the Lisa into another. Lisa was put under the control of John Couch, and Jobs was not allowed to participate. Since Lisa had been taken away, Jobs in 1981 began to assemble a team to "out-Lisa the Lisa" by creating a smaller, less expensive computer that would do the same thing. Jef Raskin, the engineer that helped design it, called it Macintosh.

While the Macintosh developed as a pirate project with a smaller team and less money than Lisa, the concept of an "appliance" computer also emerged. Instead of those messy slots and a lid that popped off (which made the Apple II so popular with the hacker community), Jobs' team was sold on the idea that all necessary features should simply be built-in and the case sealed. It would be something that you just plugged in, turned on, and started using. With the Xerox Alto mouse/icon/window interface it would not only be easy to set up and turn on, but also easy to use.

THE APPLE IIC: BEGINNINGS

What was happening with the Apple II during this time? The efforts to make it less expensive to build were progressing, and the Apple IIe was in the formative stages. In the summer of 1981 someone proposed a portable Apple II, a book-sized computer. It wasn't until Steve Jobs became interested in it as engineering challenge, well after Macintosh was under way, that anything came of the idea:

> "...one day late in '82, Paul Dali showed him [Jobs] a photograph of a Toshiba portable and they started fooling around with the idea of an Apple II that would look like the Toshiba but come with a built-in disk drive. They took out a IIe circuit board and a disk drive and a keyboard and played with them until they arrived at a promising configuration--keyboard in front, disk drive in back, circuit board in between. What got Jobs excited about this idea was the engineering difficulty of squeezing it all into a package not much bigger than a notebook. And a machine so small wouldn't have the expandability that characterized all the other II's. Like Macintosh, it could be taken out of the box, plugged in, and put to work--no extra parts to buy, no cables to figure out. It was the II reinvented as an appliance."<2>

As with all Apple projects, the IIc went by various code names during its development, for the sake of internal communications and to keep outsiders from knowing what was going on. The various names used included VLC (Very Low Cost), Yoda, ET, IIb (for "Book"), and Teddy (which stood for "Testing Every Day"). Also, following a long standing tradition at Apple, some of the code names assigned to the project at various times were names of children of people at Apple: Chels, Jason, Lolly, Sherry, and Zelda. These names persist in the source code for the firmware for the IIc as later printed in the technical reference manual; the serial port driver is called a "Lolly" driver.<3>

During the time the IIc was under development, Apple was working on a change in the look of their products. They planned a more European styling, and a color scheme called "Snow White". The IIc would be the first product with the new appearance and color.

THE APPLE IIC: HARDWARE

As mentioned earlier, the IIc had its origins while the IIe project was going on. When Steve Jobs became involved, he felt they should continue with the open IIe as they had planned, but do this other Apple II as a product "focused" to a specific group of customers, primarily new users. Originally he had planned a closed Apple II that had a built-in mouse port, one serial port, and some other features. What they ended up with at that point was just a computer and a keyboard. Walt Broedner, the engineer who pushed for the Apple IIe to be produced, used some of their previous work with custom IC's for the disk controller and combined both projects together to make the IIc.<4>

Although he was told it was not be possible, Jobs strongly pushed for the mouse in this closed Apple II to be compatible with the Macintosh

mouse--and they managed to make it work.<2> Regarding the plans for a single serial port, however, Apple's marketing people pointed out to Jobs that many people were going to want both a printer AND a modem, so they added a second port to the original design. They decided to use serial ports on the IIc instead of parallel ports for a couple of reasons. First, the socket for a serial port is smaller than a parallel port, and it would fit better onto a small box like the IIc. Also, Apple's general direction at the time was to get consistency in its hardware, and they had decided to make everything they made use a serial interface.<4>

They began work on the Apple IIc in earnest right after the IIe was finished. Because they were trying to squeeze an Apple IIe with 128K, 80 column routines, two serial cards, disk controller, and a mouse card into an 11 by 12-inch case, the design challenges were greater than with the IIe (recall that this was what appealed to Steve Jobs). The size of the case was determined by the decision to make it able to fit into a standard-sized briefcase.<4>

Apple also had the international market in mind when they designed the IIc. A special chip containing the keyboard map could easily be changed depending on the country where the computer would be sold, to make it consistent with regional keyboard differences. The external pushbutton would switch between the two different keyboards, between a UK and German layout, for example. In the U.S. version of the IIc it switched from a standard Sholes keyboard (also known as "QWERTY") to a Dvorak keyboard (which allows faster touch typing). The decision for the foreign keyboards came first; the added bonus for American versions of getting Dvorak came as an extra bonus, to save having two different cases (one for US and one for foreign versions).<4>

One problem in creating such a compact computer was dealing with heat production. Apple engineers wanted it to be able to function in environmental temperatures up to 40 degrees Celsius (about 104 degrees Fahrenheit). One article published at the time of its introduction mentioned jokingly that the designers wanted to make the IIc capable of doing a long disk sort (sorting data in a disk file) while on the beach in Florida in the summer! Their major obstacle was the heat generated by the internal 5.25 disk drive. They tried some special low power drives (which would have been much more expensive), but they didn't overcome the heat problem even with them. Eventually they tried a complicated venting scheme that was designed by drilling holes into a case and putting it into an oven to let them measure internal temperatures. The engineers were surprised when they found that the normal power disk drive worked and generated less overall heat within the case than the special low power drive did. The only explanation they could come up with was that the normal power drive generated enough heat to cause it to rise, which pulled cool air in through the vents by convection.<4>

THE APPLE IIC: FIRMWARE

Since they used the newer 65c02 chip, which ran cooler and had 27 additional commands that could be used by assembly language programs, Apple's programmers had some new power to use in firmware design. Such power was needed to squeeze in all the firmware code for the IIe, plus code for the disk controller, serial cards, mouse card, and 80 column card into 16K of ROM space.

The firmware for the IIc was written by Ernie Beernink, Rich Williams, and James Huston. They designed it to look (to a software application

program) exactly like a IIe with an Apple Super Serial Card in slots 1 and 2, an 80-column card in slot 3, a mouse in slot 4, and a Disk II in slot 6 (though there were NO slots in hardware). Since these first IIc's had nothing emulated in slot 5, the firmware authors immortalized themselves by making a "ghost" peripheral appear to be present in that slot. Entering this Applesoft program:

```
100  IN#5 : INPUT A$ : PRINT A$
```

and running it would print the names of the authors. (They used a decoding scheme to extract the names, character by character, so a simple ASCII scan of the ROM would not show their little trick). This "feature" had to be removed in later revisions of the IIc ROM, because an actual disk device was added then to slot 5.<4>,<5>

What about the unassigned slot 7? Here they put a small piece of code to allow booting from the external 5.25 drive by typing "PR#7" from Applesoft.

The programmers fixed some known bugs in the IIe ROMs, and added 32 graphics characters they called MouseText. To make MouseText fit they removed the ability to use flashing characters (when in 80 column mode) and replaced those characters with MouseText. Apple veteran Bruce Tognazzini designed the MouseText characters, which included a picture of a running man (perhaps to suggest "running" a program). He later sent a letter to Call-A.P.P.L.E. magazine to warn programmers that the Running Man characters (assigned to "F" and "G") had been determined to be unnecessary and would probably be replaced eventually. (This did eventually happen, but not with the IIc).

Beernick, Williams, and Huston also made some minor changes to the Applesoft part of the ROM. They fixed things so Applesoft commands could be entered in lowercase (and translated into uppercase). They removed the Applesoft commands that were specific to the obsolete cassette interface (which was absent in the IIc) and made Applesoft more compatible with 80 columns.<4>,<6> They did NOT go so far as to make any major changes in Applesoft to use the newer 65c02 commands and therefore fix known bugs or add features to this seven year old language. Their reluctance stemmed from the fact that historically many BASIC programs had made use of undocumented assembly language entry points in Applesoft, and any changes they would make here made it more likely that older programs would crash unexpectedly.<4>

THE APPLE IIC: PRODUCT INTRODUCTION

Apple's introduction of the new IIc came at an "event" at the Moscone Center in downtown San Francisco on April 24th, 1984. It was entitled "Apple II Forever", and was described as "part revival meeting, part sermon, part roundtable discussion, part pagan rite, and part county fair". Apple's objectives here were to introduce the Apple IIc, describe how it fit into the company's marketing strategy, show off new software that was made to work with the new computer, and emphasize that Apple was still firmly behind the Apple II line of computers. (Steve Jobs also took some of the time to report on the sales of the Macintosh in its first 100 days).<7>

One of the interesting things they did at the "Apple II Forever" event was the actual introduction of the IIc. Giant video screens were used to show previews of Apple's TV commercials for the IIc, as well as slides and images of the speakers, including Wozniak, Jobs, and Apple's new president,

John Sculley. Sculley spoke of "sharing power", and then demonstrated that in a unique way: "After holding up the tiny IIc for everyone to see and eliciting a response that they'd like to see it better, Sculley ordered the house lights on. As the light burst forth, nearly every fifth person in the audience stood up, waving high a IIc. As startled dealers cheered uproariously, the Apple plants passed the IIcs to them. Within seconds of its introduction, more than a thousand Apple dealers had a production-line IIc in their hands."<7>

When Jobs gave his report on the Mac, it revealed some interesting statistics. He told them that the first industry standard was the Apple II, which sold fifty thousand machines in two and a half years. The second standard was the IBM PC, which sold the same amount in eight months. Macintosh had done sold its fifty thousand machines only 74 days after its introduction. Although sales would not be nearly as good, Apple took orders that day for fifty thousand Apple IIc's in just over seven HOURS.

At the "Apple II Forever" event, they also had a general software exhibition and a setup called the Apple II Museum. This contained Apple memorabilia, and included Woz's original Apple I, and a reproduction of Steve Jobs' garage where it was built. Although not on the schedule, "Apple II Forever" included an early-afternoon earthquake centered south of San Jose that measured 6.2 on the Richter scale.

THE APPLE IIC: SUCCESS?

Their original goal had been to sell the IIc for $995. As productions costs turned out, they found that they couldn't hit that price, so they came up with $1,295, balancing the decision with the number of people who were predicted to buy the optional Monitor IIc or an external Disk IIc drive.

The only problem was that although the IIc was a technological breakthrough in miniaturization, customers at that time didn't value smallness. They viewed something that was too small as also being cheap and lacking power. Although the Apple IIc was equivalent to a IIe loaded with extra memory, a disk drive, two serial cards, and a mouse card, most customers seemed to want the more expandable IIe. Apple marketing went to much effort to make the IIc attractive, but it didn't sell as well as the IIe. Just as IBM overestimated the market when producing its PCjr (which eventually failed and was discontinued), so did Apple when producing the IIc (and the original Macintosh).<7>

THE APPLE IIC: OVERCOMING LIMITATIONS

Although the IIc did not have any slots for plugging in peripheral cards that had traditionally been used in the Apple II, the ports that were built-in had the capability to do much of what the slots had often been used for. The serial ports were compatible with any serial device; this included common ones such as printers and modems, and uncommon ones like security controllers, clocks, and speech synthesizers. Some third party companies also supplied serial-to-parallel converters for IIc owners who wanted to use parallel printers made by Epson, Okidata, and C. Itoh that were popular elsewhere in the computer world.

There was, of course, the AppleMouse IIc sold by Apple. It plugged into the game port on the IIc. Also available were two types of touch tablets: The Power Pad (Chalkboard) and Koala Pad (Koala Technologies),

though the latter sold best. The Koala pad would appear to a program to be the same as a joystick, but could not emulate the mouse.<8>

The disk port on the original IIc was only designed to control an external 5.25 disk drive. Apple sold the Disk IIc for $329, and other companies later sold similar drives for less. Despite this firmware limitation, Quark Engineering released a 10 MB Winchester hard drive called the QC10 that would work with this disk port, and was the first hard disk available for the IIc.<8>

The video port worked with a standard monitor, but had access to all video signals. Included with the original IIc was an RF modulator that allowed it to be connected to a standard television (for color games). An RGB adapter box attached to the video port would allow a true RGB monitor to be attached, giving color and sharp, readable 80 column text on the same monitor. Apple also sold a flat-panel liquid crystal display for the IIc that attached to this video port. It was capable of 80 columns by 24 lines, as well as double hi-res graphics. Apple's price was about $600, but it looked somewhat "squashed" vertically, and did not sell well. Another company marketed a better flat panel liquid crystal display called the C-Vue.

With a battery attached to the 12V input, and a liquid crystal display, the IIc could be made into a truly portable computer.<8>

THE APPLE IIC: ENHANCEMENTS

The earliest change made available for the IIc was a motherboard swap that fixed a hardware bug causing some non-Apple modems to fail if used at 1200 baud. This modification was made only if the owner could show they needed the change (that is, they owned a 1200 baud modem that wouldn't work).

The first significant upgrade available for Apple IIc owners was also available as a free upgrade for previous owners. Changes were made to the disk port firmware to accommodate the new 800K UniDisk 3.5. Using Apple's Protocol Converter scheme (later called "Smartport"), this new IIc could handle four 3.5 disk drives, or three 3.5 disk drives and one 5.25 drive.

With the UniDisk 3.5 upgrade, the internal 16K ROM was increased in size to a 32K ROM that was bank-switched to make space for the extra code necessary to implement the Smartport. Also added were additional serial port commands to improve compatibility with the older Super Serial Card. The Mini-Assembler, absent from the Apple II ROMs since the days of the original Integer BASIC Apple II, was added back in, with support for the extra commands provided by the newer 65c02 processor (the disassembler had always supported those new commands). The STEP and TRACE Monitor commands made a comeback, having also been a casualty of the 1979 Autostart ROM for the Apple II Plus. Rudimentary firmware was also included to allowing the IIc to be attached to an AppleTalk network (a message that said "AppleTalk Offline" would appear if you typed "PR#7" from BASIC), but it was never completed, and did not appear in future revisions of the IIc ROMs. Lastly, the new IIc ROMs included a built-in diagnostic program to do limited testing of the computer for internal failures, and had improved handling of interrupts.<9>

The next Apple IIc upgrade was known as the Memory Expansion Apple IIc. This came as a response to requests for the ability to add extra memory to the IIc. Applied Engineering had already produced a Z-80 coprocessor for the IIc (to allow access to CP/M software), and an expanded memory card, up to 1 MB, which would either act as a RAMdisk for ordinary ProDOS applications,

or as extra memory for the AppleWorks desktop (through a special patching program). Seeing the popularity of this, Apple released this third version of the IIc ROMs and motherboard, this time with a RAM expansion slot included. The Apple IIc Memory Expansion Card could take up to 1 MB of RAM, in 256K increments. The firmware in the new ROMs made it work as a RAMdisk automatically recognized by ProDOS and following the Smartport protocol that had been designed for the UniDisk 3.5. Apple even included code in the new ROM to patch DOS 3.3 so it could be used as a RAMdisk with that system (400K maximum size), and did the same with Pascal v1.3. Also, because this firmware was in the motherboard ROM, ANY company could make memory cards to attach to this version of the IIc.

Other changes made in this version of the IIc ROM included moving the mouse firmware from slot 4 to slot 7, and putting the RAMdisk firmware into slot 4. Also fixed was a bug that caused a write-protected 3.5 disk to be incorrectly identified with early versions of the UniDisk 3.5.<9>,<10>

Since code as complex as ROM firmware rarely makes it out the door without at least one bug, Apple had to make one final improvement to the IIc ROM. The Revised Memory Expansion Apple IIc (ROM version 4) included changes which made it easier to identify if no RAM chips had been installed on the memory card. A problem with keyboard buffering was also fixed. Lastly, this version of the ROM resolved an obscure bug in the slot 2 firmware that was supposed to allow the IIc to function as a simple terminal (with a modem attached to that port). The previous version of the IIc ROM had been assembled with a couple of wrong addresses in the code, and the terminal mode produced garbage. Few people used this feature, so it was not noticeable to most users, and the corrected ROM chip was therefore not as quickly available as the original Memory Expansion upgrade.

++

NEXT INSTALLMENT: Disk Evolution / The Apple IIc Plus

++

NOTES

<1> Rose, Frank. WEST OF EDEN: THE END OF INNOCENCE AT APPLE COMPUTER, 1989, p. 48.

<2> Rose, Frank. ibid, pp. 110-112.

<3> Hogan, Thom. "Apple: The First Ten Years", A+ MAGAZINE, Jan 1987, p. 45.

<4> Williams, Gregg. "'C' Is For Crunch", BYTE, Dec 1984, pp. A75-A78, A121.

<5> Little, Gary. INSIDE THE APPLE //C, 1985, pp. 1-7.

<6> Weishaar, Tom. "Miscellanea", OPEN-APPLE, Aug 1985, pp. 1.61.

<7> Durkee, David. "Marketalk Reviews", SOFTALK, Jun 1984, p. 120.

<8> Baum, Peter. "Expanding The Unexpandable IIc", SOFTALK, Jun 1984, pp. 95-97.

<9> -----. "Preface: The Apple IIc Family", APPLE IIC TECHNICAL REFERENCE MANUAL, 1984, 1986, pp. xxiii-xxv.

<10> -----. APPLE IIC MEMORY EXPANSION CARD OWNER'S GUIDE, 1986, pp. 2-4.

APPLE II HISTORY

Compiled and written by Steven Weyhrich

(PART 9 -- DISK EVOLUTION / THE APPLE IIC PLUS)
[v1.2 :: 21 May 92]

ADVANCES IN APPLE II DISK STORAGE

Since Steve Wozniak's Disk II floppy drive changed the Apple II from a hobbyist toy to a serious home and business computer in the late 1970's, the progress of disk storage has been slow for the Apple II series. In 1978, the year the Disk II was released, Mike Scott (Apple's president) and Randy Wigginton were asked at a user group meeting whether they were going to go to the larger capacity eight-inch floppy drives (which had been around before the 5.25 floppy drives). They answered that no, the Apple II was not going in that direction, but felt it might get a hard disk by 1979 or 1980, and possibly earlier than that a double sided, double density 5.25 disk with 500K per disk.<1> Of course, this never did happen; as we saw in part 7 of this historical overview, the Apple III project began to overtake the hearts and minds of Apple executives by 1979, and anything newer, bigger, or better was reserved for that machine. As a result, DOS 3.2 and 3.3 was hard-coded to work specifically with the Disk II and its 143K of available storage, and never enhanced to easily access larger capacity drives. (Later, when we examine the evolution of Apple II DOS, we will see that it was possible from the beginning for DOS 3.2 and 3.3 to access up to 400K per disk in its catalog structure; however, the low-level disk access routines built-in to DOS were ONLY for the Disk II).

So what changes DID occur in Apple II disk storage? Between 1978, when Apple released their original Shugart 5.25 inch floppy drives, and 1984, nothing much changed. Third party company produced patches that modified DOS 3.2 (and later DOS 3.3) to work with larger drives; from eight-inch floppy drives to hard disks (a whole 10 megabytes for only $5,350 from Corvus!<2>) to other various short-lived innovations, all made to try to end the "floppy shuffle". (One of the more interesting ones put five floppy disks into a cartridge, and through software made them appear to the computer as one large disk drive). Eventually Apple decided that the aging Disk II mechanism needed a face lift, and they introduced in the DuoDisk in May of 1984. This was essentially two Disk II drives in a single cabinet, with a special controller card. The drive mechanism was improved to better read half-tracks on disks (which some copy-protected software used), and at $795 was priced to be less expensive than buying two of the older Disk II drives with a controller card.<3> The most important advantage of this new design was an elimination of the "fried disk drive" problem that happened constantly with the older design. The old Disk II controller had two connectors, one for each Disk II drive that could be connected. The problem was the in the design of the connector; like the game paddle plugs for the original Apple II and II Plus, the plugs for the Disk II drives were simply a series of pins that had to be properly aligned for the drive to function (similar to the delicate pins on a computer chip). If you tried to attach the plug in such a way as to accidentally shift the pins over by one, it would burn out the

motor on the disk drive, requiring a trip for repairs to the local Apple dealer. The new DuoDisk design made connection of the disk mechanism to the controller fool-proof.

With the release of the Apple IIc in April 1984 came an external Disk II drive that was designed to plug into the new disk port in the back of the IIc, and was the same color and design as the IIc case. The Disk IIc was specific to the Apple IIc and could not be used with any older version Apple II, since it used a new, unique connector. However, since it was more expensive than a used Disk II drive, many users found out how to make a conversion cable to connect the older drive to the disk port; some even went the other direction and found ways to connect the new drive to the older Disk II controller cards for the II Plus and IIe.

The next small evolutionary step in disk storage technology for the Apple II was introduced in June 1985, with the release of the UniDisk 5.25. This drive was designed with the same appearance as the DuoDisk, but was a single 5.25 drive. It was also designed to allow one drive to be "daisy-chained" to another (one disk could plug into the back of another, forming a "chain"), instead of the older method of connecting each drive separately to the disk controller card. Its beige color was designed to match the original Apple IIe.<4>,<5>

The last version of the Disk II was called the Apple 5.25 drive. It was identical to the UniDisk 5.25 drive, except for its case, which was designed in the platinum color to match the Apple IIGS and the platinum IIe. The connector it used allowed it to also be connected in a daisy-chain fashion.<5>

NOW A WORD FROM OUR SPONSOR: BASICS OF DISK STORAGE

Let's diverge for a moment from discussing specific Apple disk products and turn to a description on how the data are stored on a disk. There are two important concepts that you need to understand to see why some methods of data storage are "faster" than other methods. The first concept is the physical data layout on the disk, and the second concept is the "logical" data layout.

The physical layout of data on a disk is important to the hardware of the disk drive. If the computer tells the disk drive to retrieve data from the disk, it has to be able to tell the drive exactly WHERE on the disk surface that data are stored. Most disk drives in use today (and when Steve Wozniak designed the original Disk II) store data on disks that are round, magnetically coated pieces of plastic that spin within a protective sleeve. The older 5.25 inch and 8 inch disks were "floppy" disks because they used a flexible protective sleeve (unlike the older yet but larger capacity "hard" or fixed disks, which usually could not be removed). The newer 3.5 inch disks are also made of the same magnetically coated plastic, but their protective sleeve is a hard shell. Within its sleeve the thin plastic disk spins around rapidly while the disk drive motor is on.

When a disk is formatted, certain addresses are written to the disk surface in a pattern that is known to the program (the disk operating system) used by the computer controlling the disk drive. Most computers divide the disk surface up into concentric rings (called "tracks"), and each track is divided up into segments called sectors or blocks. Each segment holds a specific number of bytes of data; for the Apple II, this has been either 256 bytes (sectors on 5.25 disks) or 512 bytes (blocks on newer disk devices). The number of sectors or blocks per track differs, depending on the device in

question; what is important is that the disk operating system knows how to get to the right block when a request is made of it.

The second concept, that of the "logical" layout of the disk, has to do with the way in which the disk operating system organizes the physical blocks on each track. Imagine a phonograph record on a turntable (some of you still own those, don't you?) It physically resembles a floppy disk; it is just larger in size and is not "floppy". Mentally take a white marking pen and draw lines through the center of the record, across the entire surface from side to side, making the record resemble a pizza that has been cut up into wedges.

Now draw a series of concentric circles, from the outside of the record down to the center. Each ring will, of course, be smaller than the previous ring. The rings you have drawn represent "tracks" on our simulated floppy disk, and the lines running through the center of the record represent the division of each track into blocks. Suppose we drew enough lines to divide the record up into twelve "pieces" (of pizza). That means that each "track" has twelve "blocks".

Now that you have your disk divided up (you just "formatted" it!), let's store some data on it. Numbering each "block" from one to twelve (like the numbers on a clock), let's put a checker into each block on the first (outermost) "track" (yes, a checker. You know--from the game?) Repeat the process on the second track, then the third, and so on, as far as you can go. Eventually you won't be able to fit checkers into the blocks, because they will get too small. (This points out one of the limits of floppy disks; at some point the available space on the disk becomes so small it is unusable. A standard 5.25 disk for the Apple II can have anywhere from 35 to 40 tracks (Apple has always supported only 35 tracks), while the 3.5 disk has 80 tracks. The checkers we have put in the "blocks" on this disk have also been labelled, but with the letters "A" through "L" for the first track, and "M" through "X" for the second track, and so on.

Turn on the record player. If you hold your hand over one spot on the first track on the record, you can see the lettered checkers as they move past. As it goes by, grab the "A" checker, then the "B" checker, and so on. Likely, after picking up checker "A" (on block 1), you had to wait for an entire rotation of the record before "B" comes by on block 2. The same goes for "C", "D", and so on. In computer terms, the "interleave" on this disk is 1 to 1 (written as 1:1). If you were EXTREMELY fast, you could pick up "A", "B", "C", etc. as quickly as they went by, without having to wait for the next revolution of the record. While few of us would be that fast, many of us could pick up a checker after about four went by that we didn't need. "Reload" your data on this disk, this time putting checker "A" on block 1, then checker "B" on block 5, checker "C" on block 9, checker "D" on block 2, check "E" on block 6, and so on. Now, as the record spins, you might be able to pick up "A", "B", "C", and so on without having to wait for the next revolution of the record. This would be (approximately) a 4:1 interleave. With this "logical" layout, you can pickup (load) checkers from the disk, and replace (store) checkers on the disk more efficiently. If your hands are still not fast enough, you may need to increase the interleave to 6:1 or even 8:1. If your hands are faster, you could possibly use a 3:1 or 2:1 interleave.

This is roughly what happens with disk access. A disk device and operating system that is extremely quick about processing the data it reads off a disk can have a short interleave (1:1 or 2:1). A slower disk device or operating system may need to use a 4:1 or higher interleave to work most efficiently.

One last note: Because a track on a disk contains a continuous stream of data bits, Apple drives were designed from the beginning to use "self-synchronization" to be able to tell one byte from the next. This continuous series of bits would be similar to having a paragraph of text with no spaces between words. If a sentence read "GODISNOWHERE", does it mean "GOD IS NOWHERE" or "GOD IS NOW HERE"? Some method is needed to let the computer doing the reading know where the "spaces" between bytes exists. I won't go into detail on exactly how this is carried out, but suffice it to say that some bytes on the disk are reserved for this decoding process, and so the true data bytes are specially encoded to not be mistaken for the self-sync bytes. The process of decoding these "raw" data bytes is called de-nibblization, and translates about 700 of the raw bytes read directly from the disk into 512 true data bytes. This is another part of the overhead necessary when reading from or writing to the disk; it would be similar to having to draw something on each checker with a marker as it was removed from the spinning record described above.

THE UNIDISK 3.5 AND APPLE 3.5

The first new disk drive that Apple released after the original Disk II was a 400K, single-sided 3.5 inch drive for the original Macintosh. Then, in September 1985 Apple finally released a similar drive for the Apple II series, one that was not simply a cosmetic improvement of the original Disk II drive. The UniDisk 3.5 drive was a double-sided version of the Mac drive, and could hold 800K of data. The only connection that this new drive had with the original 5.25 drives was a chip used on its controller card; this IWM chip (for "Integrated Woz Machine") put the function of the original Disk II controller onto a single chip, plus the enhancements needed to operate this higher density drive.<4> Apple's design for the UniDisk 3.5 was unique, in that it used a modification to Sony's design that varied the speed of disk rotation, depending on which concentric track was being accessed. This change made it possible for data to be packed compactly enough in the smaller inner tracks to gain an extra 80K beyond the 720K that was originally possible.

The UniDisk was directly supported by the newer Apple IIc motherboards (as mentioned in the previous part of this History), but for the older Apple II's a special controller card was required. The UniDisk 3.5 was designed as an "intelligent" drive, and had a self-contained 65c02 processor and memory to temporarily store ("buffer") data being read from or written to the disk. This was necessary because of the slow 1 MHz speed of the 6502 processors in the Apple II; they could not keep up with the faster data transfer rates possible with the 3.5 disk mechanism, plus the overhead of de-nibblization. This extra processing did cut down the speed in the UniDisk data transfer rate, but compared to the older Disk II drives it seemed MUCH faster.

With the release of the Apple IIGS in September 1986 came a new version of the 800K 3.5 drive called the Apple 3.5. This mechanism could be used on either a Mac or Apple II, fitting into the trend at Apple at making peripherals compatible between the two computers. The major difference between this drive and the original UniDisk 3.5 was that it had been lobotomized to be a "dumb" drive. Gone was the internal 65c02 processor chip used in the UniDisk 3.5 (which made it an "intelligent" drive) and the ability of the drive to buffer its own read and write operations. The newer Apple 3.5 drive did away with the extra circuitry, leaving it to the computer

to handle direct control of the drive. This could be done in the IIGS because of its faster 65816 microprocessor, which could keep up with the higher rate of data transfer. Recall the above discussion of interleave? The original UniDisk 3.5 worked best with an interleave of 4:1, but the Apple 3.5 used 2:1 interleave and could do disk reads and writes faster. Disks formatted with either drive were usable with the other one, but would be slower on the "foreign" drive.<5>

Overall, Apple released four versions of 3.5 drives between 1984 and 1986. First was the 400K drive used on the original Macintosh, then the 800K UniDisk 3.5 (which wouldn't work on the Mac), then an 800K drive for the Mac (which wouldn't work on the Apple II), and finally the Apple 3.5 drive, which worked on the Apple IIGS and the Mac, but not the IIe and original IIc.<5>

THE APPLE IIC PLUS: HARDWARE

Recalibrating our special time-travel card to focus on the final 8-bit version of the Apple II, let's travel to mid-1987. It was at this time that someone at Apple decided that the IIc needed to be upgraded. Shortly before July, three years after its original 1984 introduction, it was felt that the Apple IIc would benefit from the larger capacity Apple 3.5 drive as its internal drive. The primary intent was to make only this change, while leaving the rest of the IIc as it was. As with most other Apple projects, this went by various internal code names during its development, includine Pizza, Raisin, and Adam Ant.<11>

Trying to use the Apple 3.5 drive in the Apple IIc was certainly an engineering problem. As mentioned above, the 1 MHz 65c02 was simply not fast enough to take raw data off the Apple 3.5 drive, de-nibblize it into usable data, and pass it to the operating system. The "intelligent" 3.5 drive was designed in the first place for that very reason. To solve the problem, Apple contracted with an outside firm to design a special digital gate array that made it possible for the 1 MHz 65c02 to just barely keep up with the data transfer rate from the Apple 3.5 drive. In accomplishing this, it needed an extra 2K of static RAM space to de-nibblize the raw data from the 3.5 drive. This extra memory had to be available OUTSIDE the standard Apple IIe/IIc 128K RAM space, since there was simply not enough free memory available to spare even that little bit of space. The code Apple engineers wrote to use the drive was SO tight that there were EXACTLY enough clock cycles to properly time things while controlling the drive. (Each assembly language instruction takes a certain number of clock cycles; these cycles have to be taken into account for timing-sensitive operations such as disk and serial port drivers).

To support older Apple II software that came only on 5.25 disks, the disk port on the back was now changed to handle not only external 3.5 drives (either UniDisk 3.5 or Apple 3.5), but also up to TWO Apple 5.25 drives which could be chained together (the same drives used with the Apple IIGS). These could be chained together as could the 3.5 drives. The IIc Plus, then, could have three additional drives attached, in any mixture of Apple 3.5, UniDisk 3.5, or Apple 5.25 drives.<6>

The IIc Plus design was not thought out completely from start to finish, however. After they did the work with the special gate array to make the original IIc architecture work properly, someone decided that it was not a good idea to release a 1 MHz computer in 1987. People want speed, they reasoned; look at the world of the IBM PC and its clones, where each year faster and faster models are released. They decided then to retrofit the new

IIc with a faster 4 MHz version of the 65c02. That change, had it been done from the start, would have made engineering the internal 3.5 drive simpler; they could have just used the processor at 4 MHz for 3.5 drive access, and then used the true system speed (as selected by the user) for all other functions. The complicated gate array would not have been necessary. But, since the faster speed was added as an afterthought, and the project was under a tight schedule, the gate array design was not changed.

To accomplish the faster processor speed for the IIc Plus, Apple went to another outside firm, Zip Technologies. This company had already marketed an accelerator, the Zip Chip, that was popular as an add-on product for existing Apple II computers. Users could simply remove the 6502 or 65c02 chip in their computer, replace it with the special Zip Chip, and suddenly they had a computer that ran up to four times as fast. Apple licensed this technology from Zip, but engineers balked at actually using the Zip Chip itself for the IIc Plus. Part of this was because of the size of the Zip Chip. The chip was shaped like a standard integrated circuit, but was thicker vertically than a basic 65c02. Inside the extra space was a fast 65c02 processor, plus some caching RAM, all squeezed into a space that would fit even into the original Apple IIc (where space was at a premium). (The Zip Chip "cache" is a piece of RAM memory that is used to hold copies of system memory that the processor is frequently accessing. For instance, if a lot of graphics manipulation is being done by a program, the caching RAM would hold a copy of part of the graphics RAM, and could access it much faster than the standard RAM. This is part of what makes an after-market accelerator work).

Zip had wanted Apple to buy their Zip Chip and simply use that product in the IIc Plus. Obviously, this would have been to Zip's advantage financially. However, the thicker vertical size of the Chip made testing the completed computer more difficult, and it would be a problem to isolate product failures to the Zip Chip, instead of something else on the motherboard. By using a 4 MHz 65c02 and two 8K static RAM chips as separate components in the IIc Plus, Apple engineers could ensure that it would work and be available in a large enough volume for production. When they were designing the IIc Plus, Zip Technologies could not guarantee they could provide reliable products in the volume Apple needed.

The IIc Plus did not have the 12 VDC input on the back panel as did the earlier IIc's; instead, the power supply was built-in. This was not because it was necessarily a better design, as an internal power supply was actually less reliable ultimately than the external power supply. (It exposes the internal components to higher levels of heat over the lifetime of the product). But because many people had criticized Apple about the IIc external power supply (called a "brick on a leash" at Apple), that they had decided to make it internal on the IIc Plus as it was on all their other products. This change apparently did not cause any significant problems, as few people were actually trying to use the IIc as a "portable" computer (with a battery pack).

The memory expansion slot on the IIc Plus was not compatible with the memory cards that Apple had produced for the older IIc. This was primarily a timing problem; it was not because the RAM chips in the memory card were not fast enough to keep up with the 4 MHz speed of the IIc Plus. (Older IIc users can use an Apple Memory Expansion card with an 8 MHz Zip Chip with no problems). The IIc Plus also had an additional connector at the opposite end of a memory card plugged into the expansion slot. Signals from port 2 were made available at that end, so third party companies could make a card that was a combination RAM card and internal modem. However, this never did come

about (see below).

Other changes in the IIc Plus included a slightly redesigned keyboard and mini-DIN-8 connectors on the back panel for its serial ports (to be more compatible with Apple's new Macintosh and IIGS keyboards).

One interesting note: John Arkley, one of the engineers working on the project and a long-time Apple employee, campaigned long and hard to take things a step further. He wanted them to take an Apple IIGS motherboard, remove the slots, change the ROM to support only the internal "slots", and release a IIGS in a IIc case. He felt it would have made a great portable, non-expandable IIGS, but could not get anyone who could approve the plan to get interested in the idea.

THE APPLE IIC PLUS: FIRMWARE

The IIc Plus ROM was called revision 5 (the previous Revised Memory Expansion IIc was labelled as revision 4). The main changes present were the ones that supported the internal Apple 3.5 drive. Firmware on the new IIc was not any larger than the 32K on the previous models, but it did use the entire space (the previous IIc didn't use the last 8K available in the ROM).

One minor bug that slipped by in the IIc Plus firmware was an inability to deal with 400K (single-sided) 3.5 disks. There were few commercial software packages that came on such disks, however.<7>,<8>

THE APPLE IIC PLUS: INTRODUCTION

In September 1988 the Apple IIc Plus was introduced to considerably less fanfare than the original IIc was in April 1984. There were no promises of "Apple II Forever" this time; instead, it warranted little more than a press release in various Apple II magazines of the time. Its selling price was $675 (or $1,099 with a color monitor). This was remarkable, considering that the original Apple IIc WITHOUT a monitor sold for nearly double the price ($1,295) and had far less capacity and power than this new version. Some models of the IIc Plus were even shipped with 256K of extra memory already added. It was faster than any other Apple II ever produced (including the 2.8 MHz IIGS), and was probably the finest 8-bit computer Apple ever produced.

THE APPLE IIC PLUS: LESS THAN A SUCCESS

Early on, the Apple IIc Plus was a big seller, and by January 1989 it was above forecasted sales levels. However, the biggest hurdle that the IIc Plus had to overcome was not the external marketplace, but rather the internal market opinions at Apple Computer, Inc. Since Macintosh-mania was still in full swing at Apple, and that younger brother of the Apple II was getting most of the attention from management, the IIc Plus (as well as the IIGS) suffered. It was not because of a lack of capability, but primarily from failure to thrive due to inadequate home nutrition, so to speak. Also, the IIc Plus had the same problem as the original Apple IIc; customers seemed to want the IIe with its slots, or the greater power of the IIGS.

There were some products that were designed by third-party developers for both the IIc and IIc Plus that never made it to the market for various reasons. Applied Ingenuity (later known as Ingenuity, Inc) had two products

that would have markedly increased the portability of the IIc/IIc Plus. One was an internal hard disk they called "CDrive", which would have replaced the Apple IIc or IIc Plus internal floppy disk drive (converting it into an external floppy drive). Even more unique was "CKeeper", which was a multi-function card with many features. It could hold up to 1.25 MB of extra RAM; it had a clock/calendar chip that was ProDOS compatible; it had firmware routines to support dumping text or graphics screens to the printer; it could function as a built-in assembly language program debugger; and best of all, a feature called RAMSaver, which maintained power to the RAM chips during a power failure or if the power switch was turned off. Both of these products never saw the light of day, primarily because the company went out of business before they could be finished.<9>

Chinook Technologies actually finished design on an internal modem for the IIc Plus, but never released it. This card, 1.5 by 6 inches in size, would have mounted inside the disk drive shield. It connected to a small box attached to the outside of the IIc case, where there were cut-outs provided by Apple for connection of an "anti-theft" cable. This external box had phone jacks for the phone line and a telephone, just like most external modems. Undoubtably it never was released because of Apple's indifference towards the IIc Plus.<10>

With inadequate support by Apple marketing, third-party hardware and software developers had little motivation in designing any new products for the IIc Plus. Therefore, no unique products ever emerged on the market to take advantage of its features. Finally, in September of 1990 the IIc Plus was discontinued by Apple, leaving the platinum Apple IIe and the Apple IIGS as the remaining bearers of Wozniak's legacy.

++++++++++++++++++++++++++++++++

NEXT INSTALLMENT: The Apple IIGS

++++++++++++++++++++++++++++++++

NOTES

<1> Thyng, Mike. "Apple Source", PEEKING AT CALL-A.P.P.L.E., VOL. 1, 1978, pp. 7-8.

<2> -----. -----, APPLE ORCHARD, VOL. 1, NO. 1., Mar-Apr 1980, various.

<3> -----. "Tomorrow's Apples Today", CALL-A.P.P.L.E., May 1984, p. 78.

<4> -----. "The Marketplace", CALL-A.P.P.L.E., Jul 1985, p. 49.

<5> Baum, Peter and Allen. "Speaking Of Hardware", CALL-A.P.P.L.E., Oct 1987, pp. 30-34, 51.

<6> Weishaar, Tom. "Apple rediscovers the Apple II", OPEN-APPLE, Nov 198, p. 4.73.

<7> Weishaar, Tom. "Ask Uncle DOS", OPEN-APPLE, Jan 1989, p. 4.91.

<8> Weishaar, Tom. "Miscellanea", OPEN-APPLE, May 1989, p. 5.27.

<9> -----. "Ingenuity News", II AT WORK, Vol. 2, No. 1, Spring 1990, p. 30.

<10> Hoover, Tom. (personal mail), GEnie, E-MAIL, Nov 1991.

<11> A+ Staff. "NewsPlus", A+ MAGAZINE, Oct 1989, p. 18.

APPLE II HISTORY

Compiled and written by Steven Weyhrich

(PART 10 -- THE APPLE IIGS)
[v1.0 :: 06 Dec 91]

THE APPLE II EVOLVES

While the capabilities of the Apple II slowly advanced as it changed from the II up through the IIc, the one thing that remained essentially unchanged was the 6502 microprocessor that controlled it. Even though the 65c02 had more commands than the 6502, as an 8-bit processor it was inherently limited to directly addressing no more than 64K of memory at one time. (As an 8-bit processor, the 6502 could handle only 8 bits, or one byte at a time. However, its address bus was 16 bits wide, which made for a maximum address of 1111 1111 1111 1111 in binary, $FFFF in hexadecimal, or 65535 in decimal. If you divide 65536 bytes by 1024 bytes per "K", you get 64K as the largest memory size). When Wozniak designed it, 64K was considered to be a massive amount of memory, even for some mainframe computers. (For example, the old mainframe on which I learned programming during college back in 1975 was a ten-year-old IBM 1130 with 8K of memory; this was used for both the operating system AND user programs!) Most hackers of the time would not have known what to DO with four megabytes of memory, even if it had been possible (or affordable) to install that much. Consequently, programs of the day were compact, efficient, and primarily text-based.

The non-Apple II computer world had developed and advanced, and Apple grudgingly allowed the Apple II to make its small, incremental advances. Occasionally, efforts were made within Apple to make a more powerful Apple II, but the lure of "better" computers always turned the attention of management away from allowing such a project to actually make any progress. First the Apple III, then Lisa, and finally Macintosh swallowed the research and development dollars that Apple's cash cow, the Apple II, continued to produce. The latter two computers were based around the 16-bit Motorola 68000 microprocessor, which had the capability to address far more than 64K of memory. The Apple II could make use of more memory only through complicated switching schemes (switching between separate 64K banks). Although "Mac-envy" hit many Apple II enthusiasts both inside and outside of Apple, causing them to move away from the II, there were still many others who continued to press for more power from the II.

Eventually, a company called Western Design Center revealed plans to produce a new microprocessor called the 65816. This chip would have all of the assembly language opcodes (commands) of the 65c02 through an "emulation" mode. However, it would be a true 16-bit processor, with the ability handle 16 bits (two bytes) at a time and to address larger amounts of continuous memory. The address bus was enlarged from 16 to 24 bits, making the 65816 capable of addressing 256 times more memory, or 16 megabytes. The power to make a better Apple II was finally available.

THE RETURN OF WOZNIAK

Back in early 1981, Steve Wozniak was involved with several projects at Apple. He had helped write some fast math routines for a spreadsheet product that Apple had planned to release in competition with Visicalc. Also, Steve Jobs had managed to convince Wozniak to participate with his fledgling Macintosh project. Then, in early February, Wozniak's private plane crashed. He was injured with a concussion that temporarily made it impossible to form new memories. He could not recall that he had an accident; he did not remember playing games with his computer in the hospital; he did not remember who visited him earlier in the day. When he finally did recover from the concussion, he decided it was time to take a leave of absence from Apple. Wozniak married, and returned to college at Berkley under the name "Rocky Clark" (a combination of his dog's name and his wife's maiden name). He decided he wanted to finally graduate, and get his degree in electrical engineering and computer science. When he was done with that, he formed a corporation called "UNUSON" (which stood for "Unite Us In Song") to produce educational computer materials, wanting to make computers easier for students to use. He also decided use UNUSON to sponsor a couple of rock music events, and called them the "US Festival".<1> Held on Labor Day weekend in 1982 and 1983, these music and technology extravaganzas were invigorating for Wozniak, but he lost a bundle of money on both occasions. Though nowhere near drying up the value of his Apple Computer stock, he decided that he was ready to return to work. In June of 1983, Wozniak entered the building on the Apple campus where the Apple II division was housed and asked for something to do.

THE APPLE IIX

When Wozniak returned, he discovered the latest of the Apple II modernization projects, which was code-named "IIx". When he saw what the 65816 could do, he became excited about the potential of the new Apple II and immediately got involved. It was a tremendous boost in morale for the division to have their founder return to work. However, the IIx project was plagued by several problems. Western Design Center was late in delivering samples of the 65816 processor. First promised for November 1983, they finally arrived in February 1984--and didn't work. The second set that came three weeks later also failed.

Other problems came out of the engineering mindset that still existed at Apple at the time. Recall that people there liked designing boxes that would do neat things, but there was not enough of a unified focus from above to pull things together. The marketing department wanted the IIx to have a co-processor slot to allow it to run different microprocessors. The code name of the project by this time was "Brooklyn" and "Golden Gate" (referring to the ability to make it a bridge between the Apple II and Macintosh). The co-processor slot could allow the IIx to easily do what third party companies had done for the original Apple II with their Z-80 boards (which allowed them to run CP/M software). Co-processor boards considered were ones for the Motorola 68000 (the chip used in the Macintosh), and the Intel 8088 (used in the IBM PC). The IIx project got so bogged down in trying to become other computers, they forgot it was supposed to be an advanced Apple II. Politically it also had problems at Apple, because it was being aimed as a high-end business machine, which was where they wanted the Macintosh to go.<2>,<3> Wozniak lost interest as things ran slower and slower, and eventually the project was dropped.

THE 16-BIT APPLE II RETURNS

When the IIx project was cancelled in March 1983, some of the Apple II engineers were assigned the task of reducing the cost of the Apple II. Engineers Dan Hillman and Jay Rickard managed to put almost the entire Apple II circuitry onto a single chip they called the Mega II. Meanwhile, after the "Apple II Forever" event that introduced the IIc, interest in the Apple II revived and sales were quite good. Management saw that sales of the open IIe were better than the sales of the closed IIc, so they were agreeable to the idea of another try at the 16-bit Apple II, possibly utilizing the Mega II chip. By late summer 1984 it was revived with the code name "Phoenix" (rising from the ashes of the IIx project).<3>

THE APPLE IIGS: GOALS OF THE DEVELOPMENT TEAM

The people involved in the Phoenix project were very knowledgeable about the Apple II, from the days of the][through the //c. They knew what THEY wanted in a new computer. It should primarily be an Apple II, not just something NEW that tried to be all things to all people.<4> Dan Hillman, who had also been involved as the engineering manager for the IIx project, stated in an interview, "Our mission was very simple. First we wanted to preserve the Apple II as it exists today. It had to work with Apple IIe software and Apple IIc software. That was goal number 1. But we recognized that the Apple II was an old computer. It had limitations. The new machine needed to address those limitations, break through those barriers--and the barriers were very obvious: We needed to increase the memory size. We had to make it run faster. We needed better graphics. And we had to have better sound. That was our mission." Since advanced graphics and sound were what would make this new Apple really shine, the name eventually assigned to the final product was "Apple IIGS".<3>

Having learned from their experience in building the Apple IIe and IIc, they knew what would make the new 16-bit Apple II more powerful. The Apple IIc was easy to use because the most commonly needed peripherals were already built-in. The Apple IIe, however, excelled in its ability to be easily expanded (via the slots) to do things that were NOT commonly needed or built-in. Harvey Lehtman, system software manager for the project, stated, "We ... wanted the Apple IIGS to be easy to set up, like the IIc, and easy to expand, like the IIe."<3>

THE APPLE IIGS: ARCHITECTURE

Wozniak was quite involved in designing the general layout of the IIGS. Insisting on keeping it simple, he recommended AGAINST a built-in co-processor (as they tried to do with the IIx). He also wanted to keep the 8-bit part of the machine separate from the 16-bit part. To accomplish this, he and the other engineers decided to design it so the memory in the lower 128K of the machine was "slow RAM", which made it possible for it to function just as it did on the older Apple II's. This included the memory allocation for the odd addressing schemes used in the text and graphics modes and (which made sense in 1976, but not in 1986). The rest of the available memory space would be fast, and could be expanded to as much as 16 megabytes. With a

faster microprocessor, it would also be possible to run programs more quickly than on the older Apple II's.<3>

THE APPLE IIGS: GRAPHICS

One area they decided to focus on was bringing the quality of graphics on the new Apple II up to modern standards. Rob Moore, the Phoenix project hardware group manager, helped define the new graphics modes of the IIGS. Because a change that increased the vertical resolution from 200 dots to 400 dots would make the computer too expensive (it would require a special slow-phosphor monitor), they purposely decided not to go in that direction. Instead, they increased the horizontal resolution, and created two new graphics modes (called "super hi-res"); one was 320 x 200 and the other was 640 x 200. This decision also made it easier to keep compatibility with older graphics modes.<3>

As mentioned above, the text and graphics addressing on the old Apple II was odd, from a programming standpoint. When Wozniak originally designed the II, he made the memory allocation for text and graphics to be "non-linear", since this saved several hardware chips and made it less expensive to build. This meant that calculating the memory address of a specific dot on the hi-res graphics screen or a character on the text screen was not as simple as most programmers wanted. The hi-res screen began at $2000 in memory, and the first line on the hi-res screen (line 0) started at that address. Each line on the hi-res screen was made up of 40 bytes of 8 bits each, and seven bits of each byte represented a dot or pixel on the screen, giving a possible 280 dots horizontally. Since 40 bytes is $28 in hex, line 0 then ran from $2000 to $2027 in memory. However, the second line (line 1) of the hi-res screen did NOT start at $2028 as one would expect, but at $2080. The hi-res screen line represented by memory locations $2028 to $204F was line 8, and $2050 to $2077 was line 16. The last eight bytes of this 128 byte section of memory was unused. The next 128 bytes were allocated to screen lines 1, 9, and 17, and so on.

Because this complicated things considerably for programmers, the design team for the IIGS wanted linear addressing, which would allow the memory addresses of line 0 to be followed by the addresses for line 1, and so on. Because the graphics resolution and range of available colors planned was much greater than either of the older graphics modes (hi-res or double hi-res), they needed 32K of continuous memory to use. Because they planned on a minimum memory configuration of 256K for the IIGS as it would be shipped, they could not come up with that much memory in one single block. Engineer Larry Thompson designed a special Video Graphics Controller (VGC) to solve the problem. The chip combined two separate 16K blocks of memory and make it appear as a single continuous 32K block of memory, as far as the graphics programmer was concerned.<3>

The new super hi-res graphics modes also gave far more color choices than either the old hi-res mode (which had six unique colors) or even the double hi-res mode (which had sixteen colors). In the 320 x 200 super hi-res mode, each line could have sixteen colors out of a possible 4,096, and in the 640 x 200 mode, each line could have four colors out of 4,096. This gave graphics power that was not even available on a Macintosh (which was still black and white at the time).

THE APPLE IIGS: SOUND

The second major area of focus for enhancements over the old Apple II was sound reproduction. The original sound chip that had been proposed for the IIGS would have given it the sound quality of a typical arcade game. However, this was no better than what other computers in 1986 could do. Rob Moore suggested using a sound chip made by Ensoniq, one that was used in the Mirage music synthesizer. He had to push hard to get this included in the final design, but was able to convince management of its importance because he told them it would be "enabling technology" (borrowing a phrase from a Macintosh marketing book). He told them "it would enable people to do things they'd never dreamed of doing."<3>

The Ensoniq chip was capable of synthesizing FIFTEEN simultaneous musical voices. To help it in doing such complex sound reproduction, they gave the chip a separate 64K block of RAM memory dedicated specifically for that purpose.

THE APPLE IIGS: MEMORY

The 65816 is designed to address up to 16 MB of memory. The IIGS, however, was designed to support only 8 MB of RAM, and up to 1 MB of ROM (in high memory). With cards specially designed by third-party companies, up to 12 MB of RAM could be added, but the memory manager in ROM was only aware of the first 8 MB. A special patch was needed to allow the system to use memory beyond that point.

Building on the traditional memory organization from 6502 days, memory in the IIGS was usually referred to in banks, from $00 through $FF. Each bank refers to a 64K chunk of memory. The lowest bank, $00, was identical to the 64K memory space in the original Apple II. The next bank, $01, was the same as the auxiliary memory bank used on the Apple IIe and IIc. (Additionally, the super hi-res graphics display was found in 32K of the memory in bank $00, from $2000 to $9FFF). The banks from $02-$7F were also for RAM storage, and covered things up to the 8 MB limit. Banks $80-$DF could be used for another 4.25 MB of RAM, but as mentioned above they were unusable (without a patch) because the memory manager didn't know how to access it.

The memory expansion slot designed for the IIGS only had two lines to decode addresses. This allowed for direct access to each of four 256K RAM chips, or four 1 MB RAM chips. In order to make use of the next 4 MB of RAM some special logic was needed to find and use it. RAM cards with more than 4 MB were never directly supported by Apple.<5>

Banks $E0 and $E1 were a special part of RAM that was used to duplicate ("shadow") banks $00 and $01. This RAM was designed as "slow" RAM, and would better be able to run some of the older 8-bit Apple II software. When shadowing was active, anything a program did to addresses in banks $00 and $01 was duplicated in banks $E0 and $E1. Although it appeared to a program that it was running in the lower two banks, it was really running in the slow RAM in banks $E0 and $E1.<6>

Banks $E2-$EF were undefined. The last one MB from $F0-$FF was allocated to ROM. The lower 512K (banks $F0-$F7) were set aside for a ROMdisk. (A ROMdisk is just like a RAMdisk, except it will not lose its contents when power is turned off). For a ROMdisk to be installed, a device driver for the disk had to be located at the beginning of bank $F0 (at address $F0/0000), and the driver had to start with the phrase "ROMDISK". The most common way this was used by third-party hardware providers was to

take some of the GS memory, protect it with a battery (so its contents didn't disappear when the computer was turned off), and designate it properly to the IIGS as a ROMdisk (even though it was simply protected RAM, and not true ROM).<7>

The rest of the space from $F8-$FF was reserved for system ROM. The original IIGS had ROM code only from $FE-$FF, while later versions expanded this space to include $FC and $FD.

++++++++++++++++++++++++++++++++++++++

NEXT INSTALLMENT: The Apple IIGS, cont.

++++++++++++++++++++++++++++++++++++++

NOTES

<1> Miller, Jonathan. "The Life And Times Of Rocky Clark", SOFTALK, June 1982, pp. 141-144.

<2> Pinella, Paul. "In The Beginning: An Interview With Harvey Lehtman", APPLE IIGS: GRAPHICS AND SOUND, Fall/Winter 1986, pp. 38-44.

<3> Duprau, Jeanne, and Tyson, Molly. "The Making Of The Apple IIGS", A+ MAGAZINE, Nov 1986, pp. 57-74.

<4> Hogan, Thom. "Apple: The First Ten Years", A+ MAGAZINE, Jan 1987, p. 45.

<5> Regan, Joe. A2PRO ROUNDTABLE, Oct 1991, Category 16, Topic 2.

<6> Williams, Gregg. "The Apple IIGS", BYTE, Oct 1986, pp. 84-98.

<7> Nolan, Sean. "GS Memory Cards Compared", CALL-A.P.P.L.E., Aug 1987, pp. 10-17.

This is the ENTIRE series of articles that make up the Apple II History. They are readable in either AppleWorks 2.x or 3.0, but will require an expanded desktop for some segments.

Please feel free to make comments (on GEnie's A2 Roundtable, Category 2, Topic 16) or in E-mail (S.WEYHRICH) about the contents of these files. PLEASE, if you detect any errors or have any corrections, let me know about it. I would like to have as accurate a history as possible.

If you would like to print any of these files in a user group newsletter, I only ask that you print any segment you use in its entirety, and that you give me as the author credit for the work. Also, please send me a copy of any newsletter in which it is printed. My address is:

Steven Weyhrich
Zonker Software
2715 N. 112th St.

Omaha, NE 68164-3666

(402) 498-0246

Enjoy!

APPLE II HISTORY

Compiled and written by Steven Weyhrich

(PART 11 -- THE APPLE IIGS, CONT.)
[v1.0 :: 06 Dec 91]

THE APPLE IIGS: MISCELLANEOUS HARDWARE

Other features Apple engineers added to make the Apple IIGS a next generation computer included a built-in clock, slot space for internal expansion cards, and the electronic equivalents of seven more expansion cards.<1> Taking the cue from their experience with the Apple IIc, they included as built-in features the peripherals that most users would want to use. They allocated serial ports to slots 1 and 2, the classic 80-column firmware to slot 3, the mouse controller to slot 4, a Smartport controller to slot 5, a 5.25 inch disk controller to slot 6, and AppleTalk capability to slot 7. (AppleTalk was Apple's network protocol that had been designed originally for use with the Macintosh).

Because the engineers wanted to make the IIGS capable of connecting to the AppleTalk network, the serial ports they planned were based on a different communications controller chip than was used in the older Super Serial Card and the Apple IIc serial controller. Although the new controller chips were more capable than the older ones used on the 8-bit Apple II's, telecommunications programs written for those older Apple's wouldn't work. This was because most terminal programs, for the sake of speed, were written to directly control the old Super Serial Card (rather than going through the slower, built-in firmware commands). The controlling commands necessary to manage the newer chip were very different, and so caused such software to "break".<2>

The case and motherboard used in the Apple IIGS was made smaller than that found in the IIe, both in order to make a smaller "footprint" on a desktop, and also to make it easier to make an upgrade available for IIe owners. They had wanted to make it possible even for Apple II and II Plus owners to upgrade, but in the end it turned out to be just too expensive and difficult to execute.<2>

The Macintosh engineering group was at this time designing a protocol for interfacing standard input devices, such as keyboards, mice, and graphics tablets. This protocol, called the "Apple Desktop Bus", was first implemented on the Apple IIGS. It made possible the interchangability of hardware devices between the Macintosh and Apple II lines, allowing Apple to sell a common set of peripherals that both computers could use.<2>

THE APPLE IIGS: FIRMWARE

Firmware, you may recall, is that layer of controlling programs in ROM on a computer that sits between an application program and the hardware it is trying to control. On the IIGS, the firmware was designed after the hardware was finalized. Unlike the older ROM that Wozniak included with the original Apple II, the IIGS software engineers tried to make it more

than just a set of addresses to call to carry out a function (such as clearing the screen). Rather, they wanted to make a more comprehensive system (called a "toolbox") which could be more flexible for future enhancements of the hardware and firmware. In particular, they didn't want to have the addresses for carrying out certain functions to be fixed in a single location as on the older Apples. This toolbox would have a single address to call, and a specific command would be passed on through that address. Set up like this, it would allow Apple's firmware programmers to modify the ROM in the future without having to take trouble to make multiple addresses in the ROM "line up" properly. Additionally, they made it easy to "patch" the toolbox code in the ROM using code loaded from disk, allowing programmers to fix errors that were later found without having to replace the physical ROM chips.

At first, they were given 64K of space for the ROM, over four times as much as was available on the original Apple II. Later, they had to go back and ask for 128K of ROM, because of the many things that they needed and wanted to do. Of course, Applesoft had to be present in ROM in order to maintain compatibility with the older Apple II software. Additionally, they also put all of the mouse-handling tools into the ROM (unlike the II, II Plus, and IIe, which had to have the mouse firmware on a card in a peripheral slot).<1>

A boost to the firmware design of the IIGS came, unexpectedly, as a result of the merger between the Apple II and Macintosh divisions. This merger came as part of the reorganization that coincided with the departure of Steve Jobs from Apple. Since the Macintosh team was now working in the same place as the IIGS designers, they were available to offer help and ideas. Bill Atkinson, the programming wizard who wrote MacPaint and many of the mouse tools for the Macintosh, helped in the creation of the mouse tools and QuickDraw II for the IIGS. (This was the name given to the ROM tools used to draw on the super hi-res screen, and was borrowed from the older QuickDraw routines on the original Macintosh).<1>

To allow the user to easily configure certain features of the IIGS to their own tastes, a "control panel" was designed (another idea borrowed from the Macintosh). It was used to set the clock, the system speed (between a "normal" 1 MHz and a "fast" 2.8 MHz), change the standard text display from 40 to 80 columns, set colors for the text screen, set sensitivity of the mouse and keyboard, and make the standard settings for the printer and modem ports. These preferences were saved in a special battery-powered RAM that would survive even when the system power was turned off.<1>

THE APPLE IIGS: SYSTEM SOFTWARE

ProDOS needed to be updated to better take advantage of the additional memory on the IIGS, as well as the larger storage devices that were not very available when ProDOS was originally written. Back then, five megabytes was felt to be quite a large disk size. By the time the IIGS was designed, 40 megabytes was becoming a common standard. The new IIGS-specific version, called "ProDOS 16", would also be able to handle any number of open files at the same time (the older version of ProDOS was limited to eight files open simultaneously).<1>

The first version of ProDOS 16 was more limited than Apple's designers wanted it to be, but they didn't want to hold up the new IIGS until a better version was ready. The version of ProDOS that would run

8-bit Apple II software (on the IIGS or older Apple II's) was renamed "ProDOS 8". That version was modified to handle system interrupts better, which was important on the IIGS because of the control panel feature and the way in which the Apple Desktop Bus worked. (An interrupt refers to a special signal that is sent to the microprocessor by a hardware device. This signal "interrupts" what the processor is doing, redirects it to do something else, and then returns the processor to what it was previously doing. The mouse on the IIc and the mouse card for the other Apple II's use interrupts to handle movements of the mouse).<2>

(Further details about ProDOS 16 and its later replacement system, GS/OS, will be found in an upcoming part of the Apple II History).

IIGS PROJECT CODE NAMES AND TEAM MEMBERS

The earliest name used internally at Apple for the IIGS project was Phoenix (as mentioned earlier). It was also known as "Rambo" (when the design team was fighting for final approval from the executive staff), "Gumby" (from an impersonation done at Apple's Halloween-day parade), and "Cortland".<1>,<3>

Some of the members of the design team not yet mentioned here include Nancy Stark (an early and energetic champion for the IIGS project); Curtis Sasaki (IIGS product manager); Ed Colby (CPU product manager); Jim Jatczynski (Operating System group manager); Fern Bachman (who worked to ensure compatibility with existing Apple II software); Gus Andrate (who developed the sound tools and the unified drive firmware); and Peter Baum, Rich Williams, Eagle I. Berns, John Worthington, and Steven Glass, who each developed part of the IIGS system software and firmware.<4>

THE APPLE IIGS: PRODUCT INTRODUCTION

In September of 1986, Apple introduced the new Apple IIGS, bundled with an Apple 3.5 drive, for $999 (not including a monitor). Apple management, somewhat surprised by the response that occurred in their "Apple II Forever" event two years earlier, made the decision to heavily promote this new Apple II. Why they came to this change of heart was unclear. Although they showed no slowing in their plans for the Macintosh (which was making steady progress in gaining acceptability in the business world), a multi-million dollar marketing and media blitz was arranged to promote the new IIGS as the ultimate home and recreational use computer. Even employees at Apple who had worked on the IIGS project were startled (but pleased) at the marketing intensity that was begun, and the order for this came directly from the top. John Sculley himself had insisted that the Apple IIGS be given highest priority. (Apple's CEO since 1983, he had just a year earlier ousted founder Steve Jobs from day to day responsibilities at Apple). Rumors flew, but were never confirmed, about a shaken Sculley who had come to an executive staff meeting in July of 1986 with stories of strange things he had experienced. He had supposedly received a frightening nighttime visit from a yellow-garbed alien who called himself "Darth Vader" from the planet Vulcan. "He told me that he would meld my brain if I didn't put all I could into marketing the Apple IIGS! I have to do it!!", he was reported to have said, white-fisted and pale, at that meeting. Despite the obvious references to science-fiction movies and television of the 1960's and late 1970's, the

executive staff bowed to his requests (which were no less firm after Sculley had taken a Valium and had a couple of Diet Pepsi's. After all, he WAS the boss).

Of course, the IIGS was received by the Apple II community with enthusiasm. After initial sales broke all previous records, including those for the Macintosh, Apple re-doubled its efforts to promote this as the computer for nearly everyone. After all, it had ties into the past (compatible with Steve Wozniak's 4K Integer BASIC Apple II at its core), and ties into the future (with the 16-bit technology and expanded memory). Within a year it was outselling the Macintosh (which had also received a boost in sales, thought to be benefiting from the wave of IIGS sales).

By 1988, a significantly enhanced Apple IIGS was released, with more advanced system software (which worked more like the easy-to-use Macintosh interface) and higher density graphics (the cost of better color monitors had come down considerably since the initial design of the IIGS back in 1985). Apple even decided to take the unprecedented move of licensing the Apple II technology to a couple of other companies, who worked on producing IIGS emulators for other computers, including IBM and its clones! Software and hardware sales hit a spiraling upward curve, which stimulated more sales of computers from Apple, which increased software and hardware sales further. Apple even produced a IIGS emulator of its own for the Macintosh and Macintosh II series of computers. Eventually...

(Hold it. Something just doesn't seem right. I don't recall things going NEARLY that well for the IIGS. Computer!

APPLE IIC: [Tweedlesquirge] State request, please.

AUTHOR: Compare time events just outlined in previous section with known events in database notes.

APPLE IIC: Working... [Blinkitydinkitydinkityzeerp] Events just described are from a parallel timeline, which diverged from our own timeline in July 1986.

AUTHOR: Hmmm. Any way of moving into that timeline?

APPLE IIC: Negative. Insufficient energy available in my power supply brick to actually make changes necessary to alter the events in our timeline to allow the above scenario to actually occur.

AUTHOR: Then HOW did we come across that information in the first place?

APPLE IIC: Flux capacitor was affected by a momentary surge in power lines due to a nearby thunderstorm.

AUTHOR: Interesting. Well, maybe someday I'll have to beef up this power supply a bit and have a talk with Mr. Sculley if I can find my yellow radiation suit... So how do we get back to the correct information?

APPLE IIC: You could effect a complete shutdown and memory

purge, then reload correct data from protected archives.

AUTHOR: Very well. Make it so.

APPLE IIC: Working... [Blinkitydinkitydinkityzeerpity...]

PROOFREADER: Your Apple TALKS???

AUTHOR: What? Yes, well I had a CPU conversion done in the early 24th century...

APPLE IIC: Data reload completed. You may proceed when ready.

AUTHOR: Now, let's see if we can get it right this time...)

THE APPLE IIGS: PRODUCT INTRODUCTION (Take 2)

In September of 1986, Apple introduced the new Apple IIGS, bundled with an Apple 3.5 drive, for $999 (not including a monitor). The Apple II community was excited about the new computer, and inCider magazine featured a exuberant Steve Wozniak on the cover of its October 1986 issue with the caption, "It's Amazing!"

Apple, for its part, did do some advertising for the new computer in the pages of current Apple II publications of the time. However, there was no major push for the new computer, and again it seemed destined to be dwarfed by Apple's preoccupation with the Macintosh.

Though announced in September, the IIGS was not widely available until November. Early production models of the IIGS had some problems; one of the new chips did not work properly, and necessary changes to fix them caused a delay. The upgrade that would turn an Apple IIe into a IIGS was also delayed until early 1987.<5>

THE APPLE IIGS: ENHANCEMENTS

In September 1987 Apple made an incremental improvement to the IIGS with the release of a new ROM. The ROM 01 revision made a few changes in the original IIGS ROMs and included an improved video controller chip. Bugs in the ROM code were fixed, and a problem with a "pink fringe" effect with certain graphics displays was fixed. The new ROMs were not compatible with any IIGS System Disks earlier than version 2.0. The new ROM was identified by a message at the bottom of the screen when booting the IIGS that said "ROM Version 01". The original IIGS had no message in this location.<6>

The next change came with the release of the ROM 03 version of the IIGS in August of 1989. This new IIGS computer came standard with 1 meg of RAM on the motherboard, and twice as much ROM (256K versus 128K on the older IIGS). This allowed more of the operating system to be in ROM, rather than having to be loaded from disk when booting. Additionally, fixes were made to known bugs in the ROM 01 firmware. (The latest version of the IIGS system software made patches to ROM 01 to fix those bugs, but these patches still had to be loaded from disk, which slowed startup time. Having the latest new tools and fixed new ones already in ROM made booting

the version 03 IIGS a bit quicker). The new Apple IIGS also had the capability of using both the internal slot firmware as well as using a peripheral card plugged into a slot. The ROM 01 IIGS could, of course, use cards plugged into the slots, but only at the expense of being unable to use the internal firmware for that slot. With so much useful system firmware built-in, a ROM 01 user who wanted, for example, to add a controller card for a hard disk would have to give up either AppleTalk in slot 7 or use of 5.25 disks in slot 6. Almost everything else had to be set in the control panel to the internal firmware.

The ROM 03 IIGS also included enhancements for disabled users. A feature called "sticky keys" made it possible to do multiple keypresses. (To execute an "Option-Control-X" sequence, for example, required pressing three keys at once. This was something that a paralyzed user with a mouth-stick to press keys could not previously do). Also, more things that had required a mouse now had keyboard equivalents (using the keypad). The new IIGS also had somewhat "cleaner" sound and graphics. However, because the improvements made were minimal compared to the cost of providing upgrades to previous owners, no upgrade program was announced by Apple. In any case, many of the new features could be obtained on older IIGS's by upgrading the memory to at least one megabyte and using GS/OS System Software 5.0.2 or greater.<7>

A feature that was added to the ROM 03 firmware that was entirely fun, instead of functional, was accessed by a specific key-sequence. If the computer was booted with no disk in the drive, a message that said "Check startup device" appeared, with an apple symbol sliding back and forth. At that point, if the user pressed the keys "Ctrl", "Open Apple", "Option", and "N" simultaneously, the digitized voices of the Apple IIGS design team could be heard shouting "Apple II!" Also, the names of those people would be displayed on the screen. If running GS/OS System 5.0 or greater, the user would have to hold down the "Option" and "Shift" keys, then pull down the "About" menu in the Finder. It would then say "About the System". Using the mouse to click on that title would cause the names to be displayed and the audio message to be heard.

++

NEXT INSTALLMENT: Peripherals & the Apple II Abroad

++

NOTES

<1> Duprau, Jeanne, and Tyson, Molly. "The Making Of The Apple IIGS", A+ MAGAZINE, Nov 1986, pp. 57-74.

<2> Pinella, Paul. "In The Beginning: An Interview With Harvey Lehtman", APPLE IIGS: GRAPHICS AND SOUND, Fall/Winter 1986, pp. 38-44.

<3> Hogan, Thom. "Apple: The First Ten Years", A+ MAGAZINE, Jan 1987, p. 45.

<4> Szetela, David. "The New II", NIBBLE, Oct 1986, pp. 5-6.

<5> Weishaar, Tom. "Miscellanea", OPEN-APPLE, Nov 1986, p. 2.74.

<6> Platt, Robert, and Field, Bruce. "A.P.P.L.E. Doctor", CALL-A.P.P.L.E., Nov 1987, p. 58.

<7> Doms, Dennis. "Apple upgrades IIGS hardware", OPEN-APPLE, Sep 1989, p. 5.57.

APPLE II HISTORY

Compiled and written by Steven Weyhrich

(PART 12 -- PERIPHERALS & THE APPLE II ABROAD)
[v1.1 :: 11 Jul 92]

THE APPLE II ABROAD

Early on, Apple got involved in selling the Apple II in Europe and the Far East. To function in those parts of the world called for a change to handle a different voltage (240V instead of the 120V we use in the U.S.). Also, the language differences had to be overcome. It was easiest in Europe where, for the most part, the standard Roman alphabet was used. The primary differences were in symbols used together with letters for certain specific uses. Apple's Europlus][had a modified ROM, and certain ESC key sequences could generate the German umlaut symbol to go with certain vowels.<1>

When the IIe was released there were some other differences. The German version was built with a an external switch below the keyboard, allowing the user to change between a standard U.S. layout and a German layout. (American versions of the IIe lacked the switch, but had a place on the motherboard that could be modified to allow a Dvorak keyboard layout to be switched in instead of the standard keyboard). The IIe auxiliary slot, which was placed in line with the old slot 0 on American versions (but moved forward on the motherboard) was placed in front of slot 3 on German versions. This was because the European Apple IIe's also had added circuitry to follow the PAL protocol for video output used for televisions and computer monitors in Europe (in the U.S. the NTSC protocol is followed). Because of the extra space needed on the IIe motherboard for the PAL circuits, the auxiliary slot had to be moved to be in line with slot 3. Because the 80-column firmware was mapped to slot 3, if an 80-column card was installed in the auxiliary slot it was not possible to use any other card in slot 3. Versions of the IIe made for other European countries had similar modifications to account for regional differences.<1>,<2>

When the Apple IIc came along, it was designed from the start to take the foreign market into account. If you recall, the U.S. version of the IIc had a standard layout when the keyboard switch was up, and a Dvorak layout when the switch was down. European versions were similar to the American layout with the switch up, and had regional versions that could be swapped in with the switch down. The British version only substituted the British pound sign for the American pound sign on the "3" key, but the French, German, Italian, and Spanish versions had several different symbols available. A Canadian version of the IIc was the same as the American with the switch up, and had some other special symbols with the switch down. This version was unique because each keycap had the symbols for both switched versions. For example, the "3" key had the "3" and "#" symbols, plus the British pound symbol, making it a bit more crowded than a typical keycap.

The Apple IIGS continued the practice of making international

versions available, but improved on the design by making the various keyboard layouts all built-in. On the IIGS it was selectable via the control panel, as was the screen display of the special characters for each type of keyboard.

APPLE II PERIPHERALS

Moving on, we will now take a look at hardware items that extend the capability of the Apple II. The ability to add an external hardware device to a computer has been there from the earliest days of the first Altair to the present. In fact, the success of a computer has inevitably led to hackers designing something to make it do things it couldn't do before. The more popular the computer, the more variety you will find in hardware add-ons. The Apple II, designed by a hacker to be as expandable as possible, was once a leader as a platform for launching new and unique hardware gadgets. Today, in 1991, the Apple II unfortunately no longer holds the front position; it has been supplanted by the Macintosh and IBM crowd. However, the Apple II still benefits from the "trickle-down" of some of the best new devices from other computers (SCSI disk devices and hand scanners, for example). This is due partly to emerging standards that make it easier to design a single hardware device that will work on multiple computers, and in the case of the Macintosh, because of Apple's decision to make peripherals somewhat compatible between the two computer lines.

Trying to sort out all the peripheral devices ever designed for the Apple II series of computers into a sensible order is not easy. In this segment of the Apple II History I'll try to give an overview of hardware devices that were either significant in the advancement of the II, or unique, one-of-a-kind devices. Obviously, this cannot be a comprehensive list; I am limited to those peripherals about which I can find information or have had personal experience.

WHAT IS A PERIPHERAL?

A basic definition of a peripheral would be, "Something attached to a computer that makes it possible to do more than it could previously do." It is called a "peripheral" because it usually is connected to the computer after it leaves the factory. An argument could be made that something built-in is not a peripheral, but as things have changed over time there are some devices still called "peripherals" from force of habit, though they are now built-in (hard disks come to mind). Quite probably, in time manydevices that were once considered optional accessories will become so essential that they will always be built-in.

Recall that the earliest computers came with almost nothing built-in. They had a microprocessor, a little memory, some means of data input and display of results, the ability to access some or all of the signals from the microprocessor, and that was all. For those computers, the first things that users added were keyboards and TV monitors to make it easier to use them. Recognizing that the earliest hardware peripherals were keyboards and monitors highlights one fact: Nearly everything that is sold as a peripheral for a computer is either an input device, and output device, or an interface to make it possible to connect input and output devices. Exceptions are cards to add memory, co-processor cards to

allow it to run software from another computer, and accelerators to make the computer run faster.

EARLY PERIPHERALS

When we come to the release of the first Apple II, two important "peripherals" were built-in: A keyboard, and the circuitry to allow easy connection of a TV monitor. It had, of course, the slots for inserting expansion cards (none were available), a game port (for attaching the game paddles that were included), a pin that could be used to connect an RF modulator (so a standard television could be used instead of a computer monitor), and a cassette interface. Since there were no cards available to plug into the slots, you would imagine that the Apple II couldn't make use of any other hardware. However, those early users who had a need usually found a way around these limits.

To get a printed copy of a program listing, for example, was no trivial matter. First, there were very few printers available. Those who could, obtained old used teletypes salvaged from mainframe computers. These noisy, massive clunkers often had no lowercase letters (not a big problem, since the Apple II didn't have it either), and printed at the blazing speed of 10 cps (characters per second). To use these printers when there were yet no printer interface cards to make it easy to connect, hackers used a teletype driver written by Wozniak and distributed in the original Apple II Reference Manual (the "red book"). This driver sent characters to the printer through a connection to the game paddle port. One part of being a hacker, you can see, is improvising with what you have.<3>

Another of the earliest devices designed for the Apple II came from Apple Pugetsound Program Library Exchange (A.P.P.L.E.). They were involved in distributing Integer BASIC programs on cassette to members of the group. To make it easier to send those programs to the person responsible for duplicating the cassette, Darrell Aldrich designed a means of sending the programs over the telephone lines. There were no modems available at the time, so his "Apple Box" was attached to the phone line with alligator clips and then plugged into the cassette port on the Apple II. To send a program, you first called up the person who was to receive it and got the computers on each end connected to the Apple Box. The sender then used the SAVE command in BASIC to tell the computer to save a program to tape. In actuality, the program was being "saved" through the cassette "out" port to the Apple Box, and onto the phone line connected. At the other end of that phone line, the data went into the other Apple Box, which was connected to the cassette "in" port on the other Apple II. That computer was executing the LOAD command in BASIC to "load" the program from the Apple Box. A.P.P.L.E. sold about twenty of these Apple Boxes at $10 apiece.<3>

INTERFACE CARDS

One of the first interface cards made for the Apple II was released, naturally, by Apple. The Apple II Parallel Interface Card was released in 1977 and sold for $180.<4> Wozniak wrote the firmware ROM, and managed to make it fit entirely in only 256 bytes. As a parallel device, it used eight wires to connect the computer with a printer, one line for each data bit in a byte. Various parallel devices also used one or more extra wires

as control lines, including a "busy" line (so the receiving device could tell the sending device to stop until it was ready for more), and a "ready" line (so the receiving device could tell the sending device to resume transmission). Because each of the eight bits needed a separate wire, the cables for parallel devices looked like ribbons and were not very compact. Most of the early printers available required this type of interface.<5> A problem noticed with Apple's card, however, was an inability to properly handle these "busy" and "ready" signals (a process known as "handshaking"). One solution offered by a reader of Call-A.P.P.L.E. magazine in 1979 was to add a couple of chips to the card. If that was not done, however, the only way to do printouts that were very long was to either buy a 2K print buffer that could be used with some early printers, or use the "SPEED=" statement in Applesoft to slow down the speed at which data was sent to the printer.<6>,<7>

Apple released the Centronics parallel printer card in 1978. Selling for $225, it was specifically designed to work with Centronics brand printers.<4> It was similar to the Parallel Printer Interface, but had fewer control codes. The "Centronics standard" used seven data bits and three handshaking bits.<8> It would automatically send certain control codes to the printer when a program sent the proper command (such as a change in line width). As such, it was limited to properly working only with a Centronics printer, but many companies made printers that used the same control codes and would work with it.<5>

In April 1978 the Apple II Communications Card came out, selling for $225.<4> It was intended for use with a modem, and worked for speeds from 110 to 300 baud. The low speed (by today's standards) was for several reasons. One was that most modems of the time were acoustic. With an acoustic modem you dialed up the number yourself, and when you made a connection you put the handset (that's the part you talk and listen with, for you non-technical folks) into rubber sockets to seal out extraneous sound. A tiny speaker and microphone in the modem were then used to send and receive signals. This leads to a second reason for the low speeds of the time, which was that greater than 300 baud communications was not considered possible. In fact, the Phone Company was quite certain that speeds over 300 baud were not possible with any modem, although they would be glad to lease you a special data-quality phone line so you could get the best possible connection at 300 baud.

The Apple II Serial Interface Card ($195) appeared in August of 1978.<4> Serial devices required fewer data transmission lines, and so could work with more compact cables. Instead of sending each byte as eight simultaneous bits as was done in parallel devices, serial interfaces send each byte as a series of eight bits, which only took two wires; one to send and one to receive data. Like the parallel cards, there were a couple of other wires that went with the data lines to control handshaking. Also, serial cards needed a means of letting the sending and receiving devices identify when a byte began and ended, and the speed at which data was being transmitted. This meant that some additional information, such as "start" bits, "stop" bits, and "parity" bits, was needed.

The original version of the Serial Interface Card had a ROM that was called the P8 ROM. It contained the on-card program that allowed a user to print or otherwise communicate with the card without having to know much on the hardware level. The P8 ROM didn't support handshaking that used two ASCII control characters named ETX (Control-C) and ACK (Control-F), so a later revision called the P8A ROM was released. (ASCII stands for American Standard Code for Information Interchange). This worked better with some

printers, but unfortunately the P8A ROM was not compatible with some serial printers that had worked with the earlier P8 ROM.

The Apple Super Serial Card firmware was finished in January 1981. It was called "super" because it replaced both the older Serial Interface Card and the Communications Card. To change from one type of mode to another, however, called for switching a block on the card from one position to another (from printer position to modem position). The Super Serial Card was also able to emulate both the P8 and P8A Serial Cards, making it compatible with most older software written specifically for those cards.<9>

VIDEO CARDS

After getting a printer interface card (and printer), the next variety of peripheral cards popular for the Apple II and II Plus were ones that allowed display of 80 columns of text (which was rapidly becoming a standard outside the Apple II world). An early entry into this market was the Sup'R'Terminal card made by M&R Enterprises, the same company that made the Sup'R'Mod RF modulator for the Apple II. One of the most popular of the 80-column cards was the Videx Videoterm. Videx even made a display card that would display 132 columns card for the Apple II, but it never made much headway in the computer world (being supplanted by bit-mapped graphics displays, ala Macintosh).<3>

Many other companies made 80-column cards, but for the most part they were not very compatible with each other. One problem was deciding on a method to place the characters on the 80-column screen. With the standard Apple 40-column display, you could use either the standard routines in the Monitor, or directly "poke" characters to the screen. With these 80-column cards, they often used a standard from the non-Apple world, that of using special character sequences to indicate a screen position or other functions. For example, to put a character at row 12, column 2, a program needed to send an ESC, followed by a letter, followed by 12 and 02. Similar ESC sequences were used to clear the screen, scroll it up or down, or do other things that Apple's built-in screen routines could do.

When the Apple IIe was released, with its RAM-based method of displaying 80 columns of text, nearly all the older 80-column cards disappeared from the market. As of 1991, only Applied Engineering still makes one for those remaining II and II Plus users that don't yet have an 80-column display.

One unique video product was made by Synetix, Inc. around 1983. Their SuperSprite board plugged into slot 7 (which had access to some video signals not available on other slots), and was promoted as a graphics enhancement system. It worked by overlaying the hi-res screen with animated "sprite" graphics (programmable characters that moved independently on any screen background). Since each sprite was on its own "plane" on the screen, they didn't interfere with each other. Also, it didn't take extra effort bythe 6502 microprocessor to manipulate the sprites; once the programmer placed the sprite on the screen and started it moving, it would continue until told to change. This was much easier than trying to program a hi-res game using standard Apple graphics. Unfortunately, at the price of $395 it never took off. (It was hard for developers to justify writing programs for only a few users that might have this card). Another company later made a similar card called the StarSprite, but it suffered the same fate. Even Apple's own double hi-res

graphics, introduced on the IIe, had the same problem with a small supply of supporting software until the IIc and IIGS market got large enough to guarantee that enough owners had the capability of displaying double hi-res.<10>

ROM/RAM EXPANSION CARDS

All peripheral cards released for the Apple II up to the time of the Apple II Plus were usable only in slots 1 through 7. Slot 0 was designed differently, and until the release of the Applesoft Firmware Card ($200) in 1979 nothing had been built to make use of it. The Firmware Card contained ROM that paralleled the upper 12K of Apple II memory. If you recall from the discussion in Part 3 of this History, Integer BASIC and the ROM version of Applesoft covered the same space in memory, and so could not co-exist. When it was clear that a floating-point BASIC (Applesoft) was what many people wanted, the II Plus came out with Applesoft in ROM. To make sure that the previous Apple II owners were not left out, Apple released the Applesoft Firmware Card to plug into slot 0. It had a switch that allowed the user to select which BASIC should be active. In one position, the motherboard ROM would be selected, and in the other position the Applesoft and Autostart ROM was selected. Because there were quite a few Integer BASIC programs that Apple II Plus users wanted to run, the Firmware Card also came out in an Integer BASIC version with the old Monitor ROM, that allowed II Plus users to simulate owning a standard II.<4>

One of the benefits of the Integer BASIC ROM was the lack of something known as a "RESET vector" in the Autostart ROM. The Autostart Monitor was called that because it would automatically try to boot the Disk II drive when the power was turned on, and jumped to a known memory location when the RESET key was pressed. This allowed the disk operating system to reconnect itself, but more importantly made it possible to create copy-protected software. Since the Autostart ROM made it possible for a programmer to do something on RESET that prevented a user from examining his program, it was popular with companies producing programs that they didn't want copied and freely given away. Usually, a RESET on a protected program would restart the program, erase the program from memory, or re-boot the disk. The Integer BASIC and Old Monitor ROM lacked this feature; a RESET would just drop the user into the Monitor. This, of course, was just what hackers and those who liked to break copy-protection wanted. The users with non-Plus Apple II's or with the Integer BASIC Firmware Card on a II Plus could prevent a RESET from restarting anything, allowing them to hack a program as much as they wanted.

The next card Apple released for slot 0 was called the Language Card. It was released in 1979 with Pascal, and expanded a 48K Apple II into a full 64K memory computer. It did not remove the upper 16K of ROM, but the card contained 16K of RAM that was electronically parallel to the ROM. Using "soft switches" (recall that these are memory locations that, when read or written to, caused something internally to change) one could switch out the ROM and switch in RAM memory. This extra memory was used to load the Pascal disk system, and under DOS 3.2 and 3.3, to load into RAM the version of BASIC that was not in the ROM. This was a more flexible alternative to the Firmware Card, and opened the way to other languages beyond BASIC for Apple II users.

Since the only way to get Apple's Language Card was to buy the entire Pascal system ($495), it was too expensive for many users. Other companies

eventually came out with similar cards that did not require purchasing Pascal, and some of them designed the cards with more "banks" of memory, making 256K or more of extra memory available. Saturn Systems was one early suppliers of the large RAM cards. Typically, each 16K bank on the card would be switched in to the same memory space occupied by the Language Card RAM through the use of a special softswitch.<11>

CO-PROCESSORS

Although it did not go into slot 0, another significant card for the Apple II was the Microsoft Z-80 Softcard, which sold for around $300. It was a co-processor card, allowing the Apple II to run software written for the Z-80 microprocessor. The most popular operating system for the Z-80/8080 processors was the CP/M (Control Program for Microcomputers) system. Although the Disk II use a different method of recording data than was used by Z-80 computers, Apple II users managed to get programs such as the WordStar word processor transferred to the Apple CP/M system. Microsoft worked to make it compatible with the 80-column cards that were coming out at the time, since most CP/M software expected a screen of that size.<3>,<12>

After the arrival of the IBM Personal Computer and its wide acceptance by the business world, there was interest in a co-processor for the Apple II that would run IBM software. A company called Rana, which had been producing disk drives for the Apple II for several years, came out with the Rana 8086/2 sometime in 1984. This was a system that plugged into slots on a II Plus or IIe, and would allow the user to run programs written for the IBM PC. It would also read disks formatted for that computer (which also used a completely different data recording system than the one used by the Apple II). One Rana owner, John Russ, wrote to A2-Central (then called Open-Apple) to tell of his experience with it: "We also have one of the Rana 8086/2 boxes, with two [Rana] Elite II compatible drives and a more-or-less (mostly less) IBM-PC compatible computer inside it. Nice idea. Terrible execution. The drives are half-high instead of the full height drives used in the normal Elite II, and are very unreliable for reading or writing in either the Apple or IBM format ... And this product again shows that Rana has no knowledgeable technical folks (or they lock them up very well). We have identified several fatal incompatibilities with IBM programs, such as the system crashing totally if any attempt to generate any sound (even a beep) occurs in a program, or if inverse characters are sent to the display ... The response from Rana has been no response at all, except that we can return the system if we want to. Curious attitude for a company, isn't it?"<13> By August 1985 Rana was trying to reorganize under Chapter 11, and the product was never upgraded or fixed.

A co-processor called the ALF 8088 had limited distribution. It worked with the CPM86 operating system (a predecessor to MS-DOS) was used by some newer computers just before the release of the IBM PC.<14>

Even the Motorola 68000 processor used in the Macintosh came as a co-processor for the Apple II. The Gnome Card worked on the II Plus and IIe, but like other 68000 cards for the II, it didn't make a major impact, with the exception of those who wanted to do cross development (create programs for a computer using a microprocessor other than the one you are using).

The most successful device in this category was the PC Transporter,

produced by Applied Engineering. It was originally designed by a company in the San Jose area called The Engineering Department (TED). The founder was Wendell Sanders, a hardware engineer who formerly had worked at Apple and was involved in the design of the Apple III and parts of the SWIM chip (Super Wozniak Integrated Machine) used in the IIc and IIGS. Around 1986 Applied Engineering began discussions with TED about buying the PC Transporter to sell and market it. At that time, the board was about four times the size it eventually became. AE's people were able to shrink a lot of the components down to just a few custom ASIC chips. The software that helped manage the board originally came from TED also.<15> It was finally released in November 1987, and included a card that plugged into any of the motherboard slots (except slot 3) and one or more IBM-style disk drives. The PC Transporter used an 8086 processor and ran about three times as fast as the original IBM PC. It used its own RAM memory, up to a maximum of 768K, which could be used as a RAMdisk by ProDOS (when not in PC-mode). It used some of the main Apple memory for the interface code that lets the PC Transporter communicate with the hardware.

The PC Transporter has undergone some minor hardware changes and several sets of software changes (mostly bug fixes but a few new features). The major reasons for hardware changes came about because of the availability of cheaper RAM (the original RAM was quite expensive and difficult to obtain). Additionally, changes were made to make the onboard "ROM" software-based, which made it easier to distribute system upgrades that enhanced hardware performance.<16>,<17>,<18> The major limitation for this product has been a reluctance by Applied Engineering to match the changes that have happened in the MS-DOS world and come out with a version of the Transporter that used a more advanced microprocessor (80286, 386, or 486). As of 1991 this is slowly beginning to become more of a limitation for those who wish to use both MS-DOS and Apple II software on the same Apple II computer, since advanced software needing those more powerful processors is beginning to be released for MS-DOS.

ACCELERATORS

The two things that all computer users eventually need (or at least want) are more storage and faster speed. The 1 MHz speed of the 6502 and 65c02 chips is somewhat deceiving, when compared with computers that have processors running at a speed of 20 to 40 MHz. To put things into perspective: Since the 6502 does more than one thing with a single cycle of the clock on the microprocessor, a 1 MHz 6502 is equivalent to a 4 MHz 8086 chip. Therefore, an Apple II with an accelerator board or chip running at 8 MHz is equivalent to an MS-DOS computer running at 32 MHz.

One of the first accelerators for the Apple II was the SpeedDemon, made by MCT. This board used a faster 65c02 chip, with some high-speed internal memory that was used to actually execute the programs (since the internal Apple II memory chips were not fast enough). In essence, it put a second Apple II inside the one you could see, using the original one for input and output. Another speedup board was the Accelerator IIe by Titan Technologies (formerly Saturn Systems; they had to change their name because it was already in use by someone else). This board worked in a similar fashion to the SpeedDemon. Some users felt this product ran faster than the SpeedDemon, but it depended on the application being tested. Both boards were attached to the computer by plugging them into a slot other than slot 0 on the motherboard.

In 1986 Applied Engineering introduced the TransWarp accelerator board. This product has lasted in the marketplace longer than any of the other ones, possibly because AE did far more advertising than the companies producing the older boards. The TransWarp did the acceleration using a different method. Instead of trying to duplicate all of the Apple II RAM within the accelerator, they used a cache. (If you recall from the segment on hard disk drives, a cache is a piece of memory holding frequently accessed information). Because they used the cache, the TransWarp did not require any high-speed RAM on the motherboard. Instead, any memory access was also stored in the cache RAM, which was high-speed RAM. The next time a byte was requested from RAM, the accelerator looked first into the cache memory to see if it was there. If so, it took it (far more quickly) from there; if not, it got it from motherboard RAM and put it into the cache. Early TransWarp boards ran at 2.5 MHz; later versions pushed this speed to 7 MHz (this was the top speed used by the TransWarp GS, released in November 1988 for the Apple IIGS).

The next step in accelerator technology was to put all the components of an accelerator board into a single chip. This happened when two rivals, the Zip Chip and the Rocket Chip, were released. The Zip Chip was introduced at AppleFest in May 1988, and the Rocket Chip soon after. Running at 4 MHz, the Zip Chip was a direct replacement for the 6502 or 65c02 on the Apple II motherboard. It contained its caching RAM within the housing for the processor, the difference being mostly in height (or thickness) of the integrated circuit. Installing it was a bit more tricky than simply putting a board into a slot; the 6502 had to be removed from the motherboard with a chip puller, and the Zip Chip installed (in the correct orientation) in its place. Software to control the speed of the chip was included, and allowed about ten different speeds, including the standard 1 MHz speed (some games simply were too fast to play at 4 MHz, and software that depended on timing loops to produce music had to be slowed down to sound right). The controlling software also let the user determine which (if any) of the peripheral cards should be accelerated. Disk controller cards, since they used tight timing loops to read and write data, usually could not be accelerated, where many serial and parallel printer and modem cards would work at the faster speed. The Zip Chip even allowed the user to decide whether to run all sound at standard speed or at the fast speed.

The Rocket Chip, made by Bits And Pieces Technologies, was almost exactly the same as the Zip Chip, with a few minor exceptions. It was sold with the ability to run programs at 5 MHz, and could be slowed down below the 1 MHz speed (down to 0.05 MHz). Later, when Zip came out with an 8 MHz version of their Zip chip, a 10 MHz Rocket Chip was introduced.

The rivalry between Zip Technologies and Bits And Pieces Technologies came from a mutual blaming of theft of technical information. The Bits & Pieces people insisted that they had done the original work on a single chip accelerator with the Zip people, but had all the plans and specifications taken away without their permission. Consequently, they had to form their own company and start from scratch to design their own chip. Zip, on the other hand, insisted that Bits & Pieces had stolen the technology from them. The problem eventually came to court, and it was decided that Zip Technologies was the originator of the technique and the Rocket Chip had to stop production.

++++++++++++++++++++++++++++++++++

NEXT INSTALLMENT: Peripherals, cont.

++++++++++++++++++++++++++++++++++

NOTES

<1> Huth, Udo. (personal mail), GEnie, E-mail, Mar 1991.

<2> Spring, Michael. "Write-A.P.P.L.E.", Call-A.P.P.L.E., Apr 1984, pp. 49-50.

<3> -----. "A.P.P.L.E. Co-op Celebrates A Decade of Service", Call-A.P.P.L.E., Feb 1988, pp. 12-27.

<4> Peterson, Craig. The Computer Store, Santa Monica, CA, Store Information And Prices, Aug 10, 1979, p. 1.

<5> Bernsten, Jeff. GEnie, A2 Roundtable, Apr 1991, Category 2, Topic 16.

<6> Lewellen, Tom. "Integral Data/Parallel Card Fix", PEEKing At Call-A.P.P.L.E., Vol 2, 1979, p. 113.

<7> Golding, Val J. "Integral Data IP 225 Printer - A Review", PEEKing At Call-A.P.P.L.E., Vol 2, 1979, p. 151.

<8> Wright, Loren. "On Buying A Printer", Micro, Aug 1981, pp. 33-35.

<9> Weishaar, Tom. "Control-I(nterface) S(tandards)", Open-Apple, Oct 1987, pp. 3.65.

<10> -----. "Tomorrow's Apples Today", Call-A.P.P.L.E., Oct 1983, p. 71.

<11> Weishaar, Tom. "A Concise Look At Apple II RAM", Open-Apple, Dec 1986, p. 2.81.

<12> -----. (ads), Call-A.P.P.L.E. In Depth #1, 1981, p. 106.

<13> Weishaar, Tom. "Ask Uncle DOS", Open-Apple, Apr 1985, p. 1.32.

<14> Davidson, Keith. "The ALF 8088 Co-Processor", Call-A.P.P.L.E., Feb 1984, p. 54.

<15> Holcomb, Jeff. GEnie, A2 Roundtable, Mar 1992, Category 11, Topic 7.

<16> Utter, Gary. GEnie, A2 Roundtable, Dec 1991, Category 14, Topic 12.

<17> McKay, Hugh. GEnie, A2 Roundtable, Dec 1991, Category 14,

Topic 12.

<18> Jones, Jay. GEnie, A2 Roundtable, Dec 1991, Category 14, Topic 12.

APPLE II HISTORY

Compiled and written by Steven Weyhrich

(PART 13 -- PERIPHERALS, CONT.)
[v1.0 :: 31 Dec 91]

MODEMS

A modem is a unique peripheral device, because it makes use of two-way communication (both sending and receiving data to and from the computer). After the Apple Box sold by A.P.P.L.E., one of the first commercial modems available for the Apple II was the Micromodem II, made by D.C. Hayes in 1979. It sold for $379, and worked at the standard transmission speeds of the day, 110 and 300 baud. The Micromodem was also available for the S-100 (Altair) series of computers. Hayes' product was so popular that their command set has become a standard for modems as they have advanced over the years.

By the mid-1980's Apple released two modems with their own name on them: The Apple Personal Modem 300 and Personal Modem 1200. Both were external modems, using a direct connection to the phone line (instead of the older acoustic coupler), but were more expensive than similar products of the time. By the later 1980's they were no longer in production.

INPUT DEVICES

The number one input device for the Apple II was, of course, the keyboard. There were expanded keyboards available for the II and II Plus, bypassing the uppercase-only limit. There was once even a keyboard that had plug-in modules that would redefine specialized function keys to make them specific for different programs. Another company sold pressure sensitive pads that were attached to the Apple II keyboard above the top row and could be programmed to generate series of keypresses. The original IIe had a socket for the addition of an external numeric keypad, and the IIGS and later versions of the IIe had this keypad built-in. Because of the detached keyboard in the IIGS it was possible to select between a couple of different versions of keyboards offered by Apple |as well as from some third party companies.

The next most commonly used input device after the keyboard was the set of game paddles included with every II and II Plus. But some users needed more specialized ways to input data to the computer. A large number of interesting input devices were made available through the years; here

follows a brief description of some of them.

Creating pictures on the hi-res graphics screen has always been a challenge, from 1977 until today. Using the game paddles or a joystick is one method that could be used, but there is some difficulty in getting accurate lines and curves. Apple addressed this problem when they released the Apple Graphics Tablet in the late 1970's, which sold for about $650. This was a large flat surface, about thirty inches square, with a grid printed on the surface. Using a stylus attached to a wire leading to the tablet, and appropriate software, this could be used to draw pictures on the Apple II hi-res screen. There were two different releases of the Apple Graphics Tablet. The original one, which was released when the II Plus was the latest machine, was discontinued by FCC order because of RFI (radio frequency interference) problems. The second version, to correct that problem, was released after the IIe was in production. It used two DB-9 connectors to install on the backplate of the computer, leading to the peripheral card plugged into a slot inside. (These DB-9 connectors are the same type used on the back of the IIc and IIGS for connection of a joystick). Currently the Apple Graphics Tablet is not in production.<1>

Koala Technologies has made several input devices over the years. Their first product was the Koala Pad. Released in 1983 and selling originally for $125, this was a small graphics pad (about 8x6 inches) that plugged into the game I/O socket. It was compatible with any software that used a joystick. Using a finger or the supplied stylus, a user could draw on the pad and produce pictures on the hi-res screen with the supplied software or with some other software packages.

In November 1984 Koala released Muppet Learning Keys for $79.95. This was a device to aid preschoolers in using a computer. It was intended to help children ages three and over to learn letters, numbers, and colors, using the Muppets from Sesame Street as a learning aid. The unit used various contact surfaces to send user responses to the computer, and it attached to the Apple II via the game I/O port.<2>

The Gipson Light Pen System was also sold by Koala Technologies in 1985 for $350. Using a card in slot 7, this device used a special pen that allowed drawing directly on the computer's monitor screen.

Other devices have been released to aid in graphics manipulation on the Apple II. The Computer Colorworks released the Digital Paintbrush System in 1984 for $299. It worked on either the II Plus or IIe, and used a stylus attached by two thin dacron lines to potentiometers within the tablet, which tracked the position of the stylus. Movements of the stylus (tracing over a picture) were translated into drawings on the hi-res screen. The software included allowed creation of curves and lines, and used Fontrix fonts for lettering. (Fontrix was a program that could produce detailed hi-res graphics pictures, and had

many characters styles, or fonts, available to label those pictures). A unique feature of the Digital Paintbrush was the ability to connect two computers using the system via a modem and phone line and allow both users to draw pictures that would appear on both computers simultaneously.<3>

The input device that made the most inroads in the Apple II world was the one that was so unique to the Macintosh: The AppleMouse II. It was released in May 1984 with a program called MousePaint (similar to the MacPaint program that came with the original Macintosh). The AppleMouse came with a peripheral card to plug into a slot on the IIe or II Plus; on the IIc it just plugged into the joystick port and the built-in hardware and firmware could handle control of the mouse. MousePaint used the standard hi-res graphics screen and worked only under the ProDOS operating system, but generallygave Apple II users the capability of doing graphics in the same way as Macintosh users had been enjoying, as well as making it possible to design programs that used the mouse as a pointing and input control device.

ComputerEyes was a video acquisition system that came out in July 1984. It allowed use of a video camera to capture images and store them on the hi-res graphics page. It was a slow-scan device that attached to the Apple game I/O socket, and produced black-and-white images in about five seconds. It worked on any Apple II with 48K, Applesoft, and DOS 3.3. Made by Digital Vision, Inc., it originally sold for $129.95 ($349.95 including the video camera).<4>

MUSIC AND VOICE SYNTHESIS

Apple II's have been involved in sound from the beginning, with the inclusion by Steve Wozniak of a speaker so he could make sounds for an Apple II version of "Breakout". As simple as it was, some enterprising programmers have even managed to make this single-voice speaker sound like two and even three different voices (tones) simultaneously ("Electronic Duet" comes to mind). But that was not enough for those who wanted to have better quality music production, and so production of synthesizer cards was in full swing by the early 1980's. Some of those cards included the following:

ALF Music Card (ALF Products, Inc.) was strictly a music synthesizer, with some included software to aid in producing the music. The Mountain Music System (Mountain Computer, Inc.) was a more advanced sixteen oscillator (voice) digital synthesizer, also with software to control it. Soundchaser System (Passport Designs, Inc.) was a package that included the Mountain Music System (using slots 4 and 5), plus the Soundchaser, which was a piano-style keyboard for music input, whose card went in slot 7. It allowed four track recording and sound manipulation, using the Apple II primarily as a controller. This was probably

the most advanced music hardware system available in the days before the release of the IIGS.

The Drum-Key (made by PVI) was specifically a percussion synthesizer. It required an external amplifier and used included software to produce a wide variety of drum and other percussion sounds.<5>

Beginning in the late 1970's there were several speech synthesizers available for the Apple and other home computers. One brand was the TextTalker, and another (made by Mountain Hardware for $279) was the Supertalker. In the 1980's two other popular brands were the Echo II (slot-based) and Cricket (for the modem port on the IIc) synthesizers, made by Street Electronics. These latter also included the ability to product other sound effects, and some games released at the time had enhanced sound output when the presence of those two devices was detected. For speech reproduction, these devices usually used a method of accepting ASCII text from the computer in the form of "phonemes" to describe and produce voice through a built-in speaker. The phonemes were needed because English words have a variety of pronunciation depending on the context in which they are used. Properly programmed, the voice synthesizers could pronounce the word "root" to rhyme with either "boot" or "foot". It wasn't until the IIGS came out with the built-in capability of speech reproduction (via the Ensoniq chip) that software making use of that feature became available in any quantity.

ROBOTS AND DEVICE CONTROL

Although used primarily for education purposes, there were at least two robotic devices made to work with the Apple II. TOPO (made by Androbot, Inc.), and the Tasman Turtle ($1000, with a smaller version called the Tot for $300) were in use during the mid-1980's. Both used the Logo language to control movement of the robot on the floor. Logo has a graphics command set called "turtle" graphics to simplify the concept for children. A small triangle on the hi-res screen was called a "turtle", and it could be given software commands to move forward, turn, draw, or move without drawing. When TOPO or the Tasman Turtle were connected to an Apple II, the Logo language could be configured to send the same turtle graphics commands to the physical "turtle" robot on the floor. This gave students a concrete example of what their logo programs would do in "drawing" a graphics picture.

Education is not the only place where robotics has been used in an Apple II. Because of peripheral boards called "A/D Converters" (analog/digital converters), it is possible to take information from (for example) a wind speed sensor and convert it into digital information. A computer program can then take this information and send a command signal back to another device (perhaps to activate a motor that raises and lowers a cloth deck cover, depending on how

windy it is). Although not a "robot" in the sense that people usually view robots, a computer-controlled device of any kind is, strictly speaking, a robot. This is the concept used in the popular X-10 system used in home control. (The Introl/X-10 made by Mountain Hardware for $279 was one of the first available for the Apple II). This protocol for controlling electric devices in a home has been used for years, and programs exist for the Apple II series (including the IIc) that allow easier programming of the X-10 devices, ranging from security systems to light timers to lawn sprinkler systems.

MISCELLANEOUS HARDWARE

Here follows a short list of some other items that could be found for sale in a typical issue of an Apple computer magazine in the early 1980's:

Larger capacity disk drives were made by Lobo Drives, including an 8 inch floppy drive and other various higher density floppy disks.<6>

Hard disks, such as those made by Corvus Systems. You could get a massive 10 MB for only $5,350 (well, it was massive compared to the 143K DOS 3.3 floppy disks).

Clocks, such as the Apple Clock made by Mountain Hardware, for $199. A clock made it possible to time and date stamp files, and identify which version of a file was the most recent.

RESET Key Protector, which prevented accidental RESET on early Apple II's, was available for only $3.25 from Special Systems Design.

DoubleDOS Plus was a Disk II interface card modification that had a switch to allow the user to easily switch between DOS 3.2 and DOS 3.3. It sold for $39, by Tymac.<6>,<7>

PRINTERS

By the late 1970's and early 1980's many printers were available for use with home computers. However, the cost was often over $1,000, which limited the number of people who could afford to buy one. Most printers offered 96 characters in the standard ASCII set, including both upper and lowercase characters. The cheaper printers could only print uppercase characters, while some of the more expensive ones were capable of accepting programmable characters or had built-in graphics characters.

There were two main types of printers available. One type operated like a typewriter by striking a piece of metal type against a ribbon and onto the paper. This type of printer was often called an "impact" or "letter quality" printer. It used either a type ball like IBM's Selectric typewriters, or a wheel with spokes that radiated out from

the center, with the type characters at the end of the spokes. This latter type of letter quality printer was also called a "daisy wheel" printer, because the changeable print wheels looked something like a daisy. These printers were most commonly used by computers in businesses, as they often cost more than $2,000 and were beyond the reach of the average home hobbyist.

The other type of printer in common use was dot matrix. These less expensive printers formed characters with a series of pins in a vertical row that struck the ribbon and produced dots on the paper. As the print head moved across the paper, the dots were printed in patterns that resembled (sometimes vaguely) letters and numbers. The matrix used to form a character was usually referred to as the number of horizontal dots by the number of vertical dots. A 5x7 matrix, for example, used up to five dots across and up to seven dots down. Some printers (like some computers of the time) did not use "descenders" on the lowercase letters that drop below the baseline ("g", "j", "p", "q", and "y"). To print lowercase letters with descenders often required nine or more vertical pins.

The Centronics 730 may well have been the first "standard" printer for the Apple II (as well as for many other microcomputers). It used a parallel cable whose pin layout went on to also become a standard for use with personal computers. That pin layout on parallel cable plugs is still in use today in 1991.<8> Centronics also had several other models, including the 737 and 739. A less expensive printer made by Centronics, the 779, used 5x7 dot matrix characters, and could print in sizes from 10 to 16.5 cpi (characters per inch), ranging from 60 cps (characters per second) at 10 cpi to 100 cps at 16.5 cpi. It also had a one-line buffer (which held up to 132 characters), but printed a limited 64 character ASCII set, all uppercase plus some special characters. As mentioned before, most personal computers of the time didn't have lowercase anyway, so this limitation wasn't necessarily a drawback. The better printers made by Centronics had a larger matrix and could produce true descenders on lowercase characters.<9>,<10>

A company named Trendcom made two printers that were significant in the history of the Apple II. They had two models, the 100 and the 200. Instead of using the mechanics solenoids that drove pins in a print head, these were thermal printers that needed a special heat-sensitive paper. Their operation was very quiet, about as loud as sliding your finger across a piece of paper. They were inexpensive compared to other printers of the day (most of which cost over $1,000), although the printing looked very much like that produced by a dot-matrix printer. The Trendcom Model 100 printed 40 characters per line on paper that was about 4 1/2 inches wide. The Model 200 could print 80 columns per line on paper 8 1/2 inches wide. Compared to the first printer offered by Radio Shack for their TRS-80 computer (which was also a thermal printer but used an ugly silver paper), the Trendcom printers were very nice.

The significance of the Trendcom printer was that Apple chose it as the first printer they released under the Apple name. It could be programmed to control printing of each dot in a column, and so was ideal as an inexpensive means of printing Apple II hi-res graphics. Apple included a special interface card and released the printer as the "Apple Silentype" in June 1979 for $599. It was identical to Trendcom's Model 200 except for the Apple logo in the lower left corner of the front cover.<11> One legend suggests that part of the popularity of this printer at Apple stemmed from the fact that its small size allowed it to fit under the seat of Steve Wozniak's private airplane.<7>,<12>,<13>

Epson was another company that began early in the business of supplying printers for personal computers, and is one of the few that survives to this day. It got its start in the printer business with the Epson MX-80, one of the first dot matrix printers that sold for less than $1,000. Popular with computer hobbyists of the time, it was capable of printing Apple II hi-res graphics with the optional Graphtrax ROMs. A later version of this printer, the Epson MX-100, became available in early 1982. The MX-100 was a wide carriage model, and could print hi-res graphics without the need to add any special hardware. Epson printers were unique because they had a special feature called a "double print" mode where a line was printed normally, then the paper was advanced 1/216 of an inch and the same line printed again. This filled in some gaps between dots on individual letters, and made printouts more pleasing to the eye. Another feature used in these printers was a "print enhancement" mode, in which the pins hit the ribbon harder and made it possible to make multiple copies using carbons.<10>,<14>

Integral Data Systems was also an early manufacturer of printers. Their IDS 125 and IDS 225 printers came out in 1979 (the 225 sold for around $900).<15> These printers used a 7x7 matrix for creating characters. The IDS 125 used a pressure feed method (similar to the method used by typewriters to hold paper in place), while the IDS 225 used a tractor feed mechanism. The IDS printers had the flexibility of being useable with either parallel or serial interfaces (with serial speeds up to 1200 baud). It could do plotting of dot graphics, and also had an optional graphics character set built-in.<16>

By the late 1970's Integral Data Systems upgraded their printers, giving them more capabilities and flashier names. Their Paper Tiger line of printers (models 440 and 460) had an attractive typeface, and used two vertical rows of pins in the print head, slightly offset from each other. This produced overlapping dots to achieve a more solid appearance. Some models could print up to 160 cps, and of course upper and lowercase characters were supported. They were also capable of reproducing Apple II hi-res graphics (with the appropriate software). IDS also sold a printer called the Prism, which could print in color using a special

multicolored ribbon.<17>

Other early printers were made by Anadex, MPI, and Microtek.

APPLE'S PRINTERS

After the Silentype printer was released in 1979, Apple looked for another printer that would produce better, more permanent output than could be achieved with a thermal printer. One of the main problems with thermal paper was that with time the printing could fade, especially if cellophane tape was used on the paper. The Apple Dot Matrix Printer was released in October 1982 for $699. Made from a modified C. Itoh printer, it was one of the first few dot-matrix printers that sold for under $1,000. Apple needed it as a better quality printer than the Silentype to help promote the Apple III as a business computer. More importantly, it was chosen by Apple because it was capable of doing heavy-duty graphics reproduction (such as output from the Apple Lisa computer, still in development at that time). Known also as the Apple DMP, it used a custom ROM programmed by Apple to control the printer's features.<18>

Because Apple was looking for as many business solutions for its customers as it could find, they also announced at the same time as the DMP a daisy wheel printer called the Apple Letter Quality Printer. Costing a hefty $2,195, and made from a modified Qume brand printer, this printer could print at a blazing 40 cps, but did produce very good quality output. It was released with the Lisa and IIe in January 1983.<18>,<19>

The Apple ImageWriter was released in December 1983 as the successor to the Apple DMP. Also made by C. Itoh, it had a faster print speed (120 cps), and could print in eight different pitches (character widths). It was a very reliable, sturdy printer, and sold originally for $675. Later, a wide carriage version whose abilities were otherwise identical was made available. It was replaced by the ImageWriter II in September 1985. The original Apple DMP and the ImageWriter I came in the same beige color as the Apple II, II Plus, and IIe. The ImageWriter II was the same platinum color as the Apple IIGS and the newer Macintosh computers. Styled a little differently, the ImageWriter II could do everything the original ImageWriter could do, plus it was capable of printing MouseText characters and could print in color (using a special multicolored ribbon).<19>,<20>

As part of its promotion of the Apple IIc, a new printer was released. The Apple Scribe came in the same "Snow White" color as the IIc and was low in cost at $299. It was a thermal printer, but was a significant advancement over the old Silentype. It could print on regular paper (instead of special heat sensitive paper), and could print in four colors. It could do this using a unique heat-transfer method and a wax-impregnated ribbon. It could

print in a "near letter quality" mode (with overlapping dots) at 50 cps, and a draft and graphics mode (80 cps). Its major limitation, however, was a print quality that overall was often not as good as some dot-matrix printers, and a ribbon that was expensive and needed to be replaced too often. The Scribe was eventually discontinued due to these problems and low sales.<19>

In 1984 Hewlett-Packard introduced the LaserJet laser printer. This was a significant breakthrough in printer quality, and was capable of producing documents that looked professionally typeset. Apple decided to develop its own laser printer, and in January of 1985 released the LaserWriter. Although not speedy printers (with best output at four pages a minute by 1991), and very expensive (over $2,000), they were popular with those who wanted high quality printing. At Apple, the new LaserWriter was supported only on the Macintosh, but since the printer did its work through a page description language called "PostScript", it was entirely possible for an Apple II to print on a laser printer. It was only necessary to learn the PostScript language, create a file that gave the necessary commands, and send that file to the printer through a serial interface card. Don Lancaster, long-time Apple II supporter and hacker, wrote a series of articles called "Ask The Guru" in the magazine Computer Shopper, and he gave many examples of using a laser printer with an Apple II.

Unfortunately, to this day the perception still exists that a laser printer will not work with an Apple II, even if it is a IIGS. This is partly because there are few software packages for the Apple II that will produce output as PostScript files that can be properly interpreted on a laser printer. However, programs such as "Publish-It!" will print to a PostScript-capable laser printer even on an Apple IIc. All that needs to be done is to have the right cable to connect the two devices.

One of the newest types of print technology to come to personal computers is known as the ink-jet printer. This type of printer works with a dot-matrix, but does not use pins impacting a ribbon. Rather, it uses a print head that sprays ink through as many as 64 holes in patterns to form characters as moves across the paper. The advantage over dot-matrix impact printers is its ability to form more solid characters. In fact, the quality of printout with an ink-jet printer can be almost as good as that obtained with a laser printer. The advantage over laser printers is cost. Where the best price for a laser printer in 1991 is still well over $1,500, the cost of ink-jet printers is getting as low as $500, and for some brands down to $300. The disadvantage for Apple II users? Although it is easy to get the printers to reproduce text, printing graphics to work may be difficult until Apple II software packages directly support those printers. Fortunately, most of these printers will emulate some brands of dot-matrix printers, and if that brand is supported by a software program, then graphics

reproduction may be possible.

Apple entered the ink-jet printer market in May 1991 when it released the Apple StyleWriter. A modification of Canon's BubbleJet printer, this printer does excellent reproduction of text and graphics--on a Macintosh. Unfortunately, Apple didn't see fit to release drivers (programs to control hardware) to make it possible to use this printer on the IIGS or IIe. It does make use of a new font (typeface) technology called TrueType, which makes it possible to have a single font that can be made any size under software control (instead of having a separate font for each size that you might want to print). It was not until early 1992 when a program called Pointless was made available for the IIGS (not from Apple) that TrueType could be used on that computer.

Although not quite a printer, the Apple Color Plotter was released in June of 1984. It had an advantage over printers, in that it could draw smooth lines and curves. Using four colored pens in a rotating pen head, and selecting them at the computer's command, the Color Plotter worked by moving the paper up and down to draw vertical lines, and the pen left and right to draw horizontal lines. Control of the plotter was accomplished by sending text commands through a serial card, and consisted of two letter commands (DA = Draw Absolute, DR = Draw Relative, etc.) followed by parameters. It could move the pen without drawing, plot points, draw lines, arcs, and circles, and print text at any location, tilt, rotation, or scale. Lines could be drawn as solid or as patterns of dots.

Presumably this product did not take off because of the limited need for this type of graphics, and the price. Today, although the quality of screen and printer graphics is greatly improved over what was available in 1984, a plotter can still be useful in some situations. Usually, however, the right software can reproduce drawings with a dot matrix or laser printer in as good or better detail than a plotter can.<21>

++++++++++++++++++++

NEXT INSTALLMENT: DOS

++++++++++++++++++++

NOTES

<1> Weisman, Tyler. (personal mail), GEnie, E-mail, Aug 1991.

<2> -----. "The Marketplace", Call-A.P.P.L.E., Nov 1984, p. 41.

<3> Neibauer, Larry. "Reviews: Digital Paintbrush",

Call-A.P.P.L.E., Nov 1984, p. 36.

<4> -----. "The Marketplace", Call-A.P.P.L.E., Jul 1984, p. 61.

<5> (various). "Reviews: Music Systems For The Apple II". Call-A.P.P.L.E., Jun 1984, pp. 17-31.

<6> -----. -----, Apple Orchard, Vol. 1, No. 1., Mar-Apr 1980, various.

<7> -----. (ads), Call-A.P.P.L.E. In Depth #1, 198, p. 106.

<8> Zuchowski, Tom. GEnie, A2 Roundtable, Mar 1991, Category 2, Topic 16.

<9> Ulm, Dennis. GEnie, A2 Roundtable, Apr 1991, Category 2, Topic 16.

<10> Wright, Loren. "On Buying A Printer", Micro, Aug 1981, pp. 33-35.

<11> Bernsten, Jeff. GEnie, A2 Roundtable, Apr 1991, Category 2, Topic 16

<12> -----. "A.P.P.L.E. Co-op Celebrates A Decade of Service". Call-A.P.P.L.E., Feb 1988, pp. 12-27.

<13> Felty, Wes. GEnie, A2 Roundtable, Apr 1991, Category 2, Topic 16.

<14> Kindall, Jerry. GEnie, A2 Roundtable, Mar 1991, Category 2, Topic 16.

<15> Peterson, Craig. The Computer Store, Santa Monica, CA, Store Information And Prices, Aug 10, 1979, p. 1.

<16> Golding, Val. "Integral Data IP 225 Printer - A Review", PEEKing At Call-A.P.P.L.E., Vol. 2, 1979, p. 151.

<17> Vanderpool, Tom. GEnie, A2 Roundtable, Mar & Aug 1991, Category 2, Topic 16.

<18> Williams, Gregg. "The Lisa Computer System", Byte, Feb 1983, p. 43.

<19> Baum, Peter. "Expanding The Unexpandable IIc", Softalk, Jun 1984, pp. 95-97.

<20> -----. "The Marketplace", Call-A.P.P.L.E., Nov 1985, p. 50.

<21> Durkee, David. "Marketalk Reviews", Softalk,
Jun 1984, p. 120.

APPLE II HISTORY

Compiled and written by Steven Weyhrich

(PART 14 -- DOS)
[v1.0 :: 12 Jan 92]

APPLE DOS

For a computer to be useful, it must have a means of easy storage and retrieval of data. That storage medium must be both convenient and affordable. In the early days of the Apple II computer, the best that they could achieve was "affordable". The built-in cassette port was the state of the art for personal computers back in 1977; the Apple I computer had a cassette interface available only as an add-on item. But, although a cassette storage system may be inexpensive, it is not very convenient. The simplistic cassette operating system on the Apple II (visual examination of the mechanical index counter on the cassette recorder to know the location of the next program) was downright frustrating to use for many early Apple II owners. Something better was desperately needed.

As you may recall from Part 5 of the History, in December of 1977 Steve Wozniak began a crash effort to develop a floppy disk drive for the Apple II computer. To get it ready for the Consumer Electronics Show in January 1978, Wozniak and Randy Wigginton made a very simple disk operating system that would only load files from fixed locations off the disk in response to one-letter commands. But it was not a true disk operating system (DOS); their rudimentary control program would not be flexible enough for efficient and simple use of the disk drive.

DISK SYSTEM BASICS

To create an operating system that would be both simple to use and yet powerful enough for advanced file manipulations, Apple had much work to do, building on the device driver that Wozniak had written. Among other things, it had to interface well with the BASICs in ROM on the Apple II, and be no more complicated to use than the cassette system. Although Woz's driver routines were efficient in writing and reading data to and from the disk, they could only be used from 6502 assembly language.

Designing a disk operating system from scratch is no trivial matter. On one side is the RAM memory in the Apple II, waiting patiently for a useful program to be loaded and executed. On the other side of an electronic bridge (interface card and connecting cable) is the floppy

disk and disk drive hardware itself. The control program the Woz wrote could be compared to a narrow rope bridge crossing a chasm; it works, but you can't carry much with you, and it is easy to slip and fall (lose data). A complete DOS is more like a concrete and steel bridge, capable of carrying autos and trucks in both directions over the chasm. Woz's "rope bridge" was a foundation, but there was much work yet to do.

A disk drive consists of a recording head that is mechanically moved across the surface of the floppy disk, tracing the radius of the disk from the center to the edge. The disk itself is spinning under the head. This is similar to the stylus on a turntable that plays 33 RPM records (remember those?), but the head on a disk drive can be given a command to move to a different "track" on the spinning disk. Also unlike the turntable, which is a "read-only" device, the head on the disk drive can either reads bits off or write bits onto the disk. To be able to find where data has been stored on a disk, it is "formatted" into a known configuration. A blank disk could be compared to empty land that will be filled with new houses, but currently has no streets, street signs, or house numbers. The initial formatting (called "hard" formatting) of a blank disk is, then, like building the streets and assigning lots for future building. The second part of disk formatting (called "soft" formatting), involves naming the streets, designating addresses, and building houses.

In the case of Apple's Disk II, it was designed with 35 concentric circles ("streets") called tracks. Each track is subdivided into 16 segments ("houses") called sectors. Each sector can hold 256 bytes of information. In the hardware system that Wozniak designed, the timing hole near the center of the floppy disk was not used by the hardware to keep track of which sector was passing the head at any particular time. Because of that, it was necessary for the software to identify in a different way where one sector ended and the next sector began. A complicated method was used of specially encoding each of the 256 bytes so they have a standard, recognizable appearance to a program that is controlling the disk drive, plus some other specialized bytes that identify the start and end of a sector. Although it did decrease somewhat the storage capacity of the disk, the cost savings in less complicated hardware compensated for it.

DOS 3.1 - STRUCTURE & FUNCTION WITH BASIC

With this background, let's get back to tracing the gap between Woz's demo DOS and Apple's first official release, DOS 3.1. Worth and Lechner in their book, "Beneath Apple DOS", divided DOS up into four parts according to function and location in memory. When a computer needs an operating system, it's because there is a need to insulate the user from the complexity of trying to control the

hardware. Consider the four parts of DOS as layers; as you get closer to the bottom layer, you are closer to the hardware (the raw data on the disk and direct control of the disk drive), but you also increase greatly the difficulty of managing it. The farther up you go, the easier it is to manage things on the disk, but the less direct is the control of the disk data and hardware.<1>,<2> When Wozniak wrote his disk controller (driver) routines, he worked at the deepest layer, directly manipulating the disk hardware and raw data. This involved some complex timing and error checking for reading and writing data to the disk. This section is also where the program lies that erases the disk and creates the sectors and their addresses. In memory, this layer of DOS started at $B800 on a 48K Apple II.<2>,<3>

Randy Wigginton wrote a "front end" for Wozniak's controller routines. His part could be considered a thin layer that is part of the lowest layer of disk routines. Together, the two layers made up what came to be known as "RWTS", or "Read/Write Track/Sector". It could do four things only: SEEK (to move the disk arm to the desired track), READ (load a sector from disk into memory), WRITE (save a sector to disk from memory), and FORMAT (discussed above). This layer of DOS, the Disk II driver, started at $B600.<2>,<3>

Apple contracted with an outside consultant, Bob Shepardson, to write much of the rest of DOS (though modifications were made Apple programmers Dick Huston and Rick Auricchio).<4>,<5>,<6> Shepardson's group wrote the layers (parts) of DOS that later became known as the "File Manager" and the "Main DOS routines". The File Manager was the next layer in memory above RWTS. It started at $AAC9 in memory, and was responsible for twelve higher level functions that dealt with files and the disk in general. These functions were OPEN, CLOSE, READ, WRITE, DELETE, CATALOG, LOCK, UNLOCK, RENAME, POSITION, INIT (format a disk and create an empty catalog track), and VERIFY. This set of routines, along with RWTS, would be similar to the file PRODOS in the current 8-bit disk operating system. It handled the disk at the file level, but knew nothing about BASIC.<2>,<3>

The next layer of code above the File Manager contained the Main DOS Routines. These routines started at $9D00 in memory, and were responsible for interfacing BASIC with the disk. This layer would be similar to the file called BASIC.SYSTEM used today in the ProDOS system. Since neither Integer BASIC nor Applesoft were specifically modified to handle disk commands, this part of DOS kept a constant look at any output PRINTed by BASIC. When a BASIC program was running, DOS looked to see if the character Ctrl-D (hex $04) was printed immediately after a Ctrl-M (carriage return). If that sequence was detected, DOS assumed that the next text printed was a command for it. If a BASIC program was not running, then DOS examined anything typed directly from the keyboard. If it decided that a DOS command had been entered, it would execute that command. If

the user typed a command that DOS recognized (such as "RUN PROGRAM" or "SAVE PROGRAM") but which resulted in a disk error, DOS 3.1 would generate an error message. On the other hand, if DOS did not recognize the command, it passed it on to the active BASIC for processing.

The final, uppermost layer of DOS was not a program code area but a set of memory areas called "buffers". One buffer was used by DOS for each open file. These buffers ordinarily started at $9600 in memory.

Here is an example of how the layers of DOS interacted: When a user typed the command "LOAD PROGRAM" at the keyboard, DOS intercepted the statement. The Main DOS Routines determined that it was a legal DOS command. The File Manager was called to 1) OPEN a file named "PROGRAM", 2) READ all the bytes associated with that file into memory starting at a specific location, and then 3) CLOSE the file. The File Manager's OPEN command in turn instructed RWTS where to move the disk read/write head, and in what order to read the correct tracks and sectors to find the contents of the entire file, wherever it happened to be on the disk. Complicated, perhaps, but the only thing the user had to know was how to type "LOAD PROGRAM".

Finally, one piece of trivia: Why was the first DOS released for the Apple II called "DOS 3.1" rather than "DOS 1.0"? According to Steve Wozniak, it was Bob Shepardson's group that decided on calling it "DOS 3". It is unclear why Shepardson decided on "3"; possibly it referred to internal revisions done by Shepardson, or perhaps it was a modification of some DOS routines done for another computer that had used earlier version numbers.<2> (Note: DOS 3 was never actually released to the public; that version apparently had a few bugs left to fix, so "DOS 3.1" came with the first Disk II drives shipped by Apple to their dealers).

DOS 3.1 - MANUAL

When originally introduced with the new Disk II drive in 1978, DOS 3.1 had very little documentation. Because the demand for the disk drive was so great, the engineers at Apple had worked feverishly to produce enough working drives to begin shipping. They went out, although there was not time to complete a real manual on how to use the disk operating system. They did include a leaflet about some of the commands, but there were still, obviously, complaints. One letter to Apple president Mike Markkula made these blunt comments: "You [expletive deleted]. I bought an Apple with floppy and nobody, I mean nobody, in L.A. or San Diego knows how to use the [thing] for random access files. I really feel 'ripped off.' Everybody talks about this great manual in the sky that is coming out soon??? ... [more expletives]! I need this computer now in my business not next year. [Expletive]. I hope your dog dies."<7>

It was not until the release of DOS 3.2 in February

1979 that a true reference manual was made available. It was given the unwieldy title, "Disk II Floppy Disk Subsystem Installation and Operating Manual", and subtitled "Apple Intelligent Subsystems (part #030-0011-00)". It was all of 38 pages long, with weak jokes and typos, but not much else of substance. Instruction on how to READ and WRITE text files was given in a mere ten lines, with no programming examples. The EXEC command was given a little more description, but was still unclear to many users. The manual also talked about " *3D0G ". What it didn't say was that this meant that the user was supposed to type "3D0G" from the Monitor prompt (to allow a return to the active BASIC with DOS connected).<8>,<9>

DOS 3.1 - FEATURES

A catalog of the DOS 3.1 System Master disk would produce this output:

```
  I 007 HELLO
 *I 043 APPLESOFT
  I 016 ANIMALS
  I 009 COLOR DEMOS
 *I 004 MASTER.CREATE
 *B 039 RAWDOS
 *I 007 COPY
 *B 007 COPY.OBJ
```

"HELLO" was the startup file executed when the disk was booted. It just displayed the following:

```
  DISK II MASTER DISKETTE  VERSION 3.1

                             20-JUL-78

  COPYRIGHT 1978   APPLE COMPUTER INC.

>_
```

stopping at the Integer BASIC prompt. "ANIMALS" was an Integer program that gave an example of the use of disk files, and "COLOR DEMOS" was a disk version of a program that had earlier come on cassette. "MASTER CREATE" was a program that could be used to initialize a "master" disk. Using the binary file "RAWDOS", it executed the DOS "INIT" command, but put a version of DOS on the newly formatted disk that was relocatable.<10> When DOS from a "master" disk was booted on an Apple II, it first determined what was size of the memory, and then loaded itself into memory as high as possible. The INIT command properly formatted a new disk, but created what Apple called a "slave" disk; that is, the DOS loaded from a slave disk was fixed in memory to the same size as the computer on which DOS had been booted. In

most cases this would not be a problem. However, the problem would surface if someone whose Apple II had only 16K of RAM shared a disk with a friend whose computer had, say, 32K of memory. Booting that borrowed disk would make the 32K computer appear to have only 16K of RAM (since it forced DOS to load at the highest location available to a 16K machine). A "master" disk was more versatile, being "intelligent" enough to adapt itself to differing memory sizes.

The Integer BASIC file "APPLESOFT" was interesting. It was a 43 sector file that appeared in a catalog as an Integer BASIC program (with the "I" filetype code). If you loaded the file and listed lines 10 through 80, there were lines that would produce the following text:

```
****************************************
*                                      *
*  APPLESOFT ][ FLOATING POINT BASIC   *
*              APRIL 1978              *
****************************************


COPYRIGHT 1978 APPLE COMPUTER, INC.

COPYRIGHT 1976 BY MICROSOFT

ALL RIGHTS RESERVED
```

There were also lines that poked some values into memory, and then jumped to a machine language routine that relocated Applesoft into RAM starting at $800 (the same place where Cassette Applesoft loaded). If you tried to LIST the entire program in memory, the lines after line 80 appeared to be a jumble of Integer BASIC commands. This is because a majority of the file was actually a machine language program that had been appended to the end of the short Integer BASIC program that displayed the title above and did the memory pokes. This machine language code was the Applesoft BASIC interpreter. Now, if the file "APPLESOFT" was executed by typing "RUN APPLESOFT", it would display the title and leave the cursor next to the Applesoft bracket prompt. However, DOS was no longer connected; the result was much like using Cassette Applesoft. To properly use this file with DOS, you had to type "FP" from the Integer BASIC prompt. DOS would then load the "APPLESOFT" file and properly initialize the interpreter, leaving DOS connected. Since this version of Applesoft still had a few bugs in it, this method of using Applesoft was obsoleted by the Applesoft Firmware card and the Apple II Plus.<9>

Interestingly, the error messages produced by DOS 3.1 were made to look similar to those displayed by Integer BASIC. For example, this is what happened if an attempt was made to load a type "B" (binary) file with the "LOAD"

command:

```
>LOAD COPY.OBJ
***DISK: NOT BASIC PROGRAM
>_
```

Integer BASIC had error messages that looked like "*** SYNTAX ERR" (with a space following the asterisks). The possible error messages in this version of DOS that were different from later versions were:

```
SYS ERROR
CMD SYNTAX ERROR
NO FILE BUFFS AVAIL ERROR
NOT BASIC PROGRAM ERROR
NOT BINARY FILE ERROR
```

DOS 3.1 - USER EXPERIENCES

One problem encountered by early users of the Disk II was properly connecting the drive to the controller card, as discussed in Part 9 of this History. Some quirks in DOS that plagued users at the time of the first releases of DOS 3.1 included one in which LOCKing a file sometimes mysteriously caused the length of the first file in the catalog to change. Apple told people not to worry about that; in fact, they told people not to pay attention to the sector counts in the catalog at all, as there was a bug in that part of the catalog routine. Another problem in early versions of DOS 3.1 was an inability to execute READ or WRITE statements in an Applesoft program if they occurred in program lines that were numbered higher than 256. It also wouldn't allow more than one DOS command on the same line of a program, so this was not possible:

```
10 ON ERROR GOTO 1000
20 PRINT D$;"VERIFY FILE": PRINT D$;"OPEN FILE": PRINT
D$;"READ FILE"
```

Other bugs in early versions of DOS 3.1 included not being able to initialize disks with MASTER.CREATE unless the disk controller was moved to slot 7. (Originally, slot 7 was going to be the disk slot, but Apple decided to change it to slot 6 and leave slot 7 for video cards. Why the various 80-column cards that were eventually released were made to go into slot 3 instead of slot 7 is anybody's guess). The A.P.P.L.E. user group had patches to MASTER.CREATE and RAWDOS to fix the slot 7 INIT bug, and the >255 line number bug in Applesoft.<11> Apple later released a modified version of DOS 3.1 that fixed these bugs (without changing the version number).

DOS 3.2 - ENHANCEMENTS

As mentioned above, DOS 3 and 3.1 had a few problems. When the Apple II Plus with the Autostart ROM was released, DOS needed to be updated to handle the changes. DOS 3.2, released in February 1979, contained several modifications, but retained 90 percent of the basic structure of DOS 3.1. One interesting change made to plan for the future was a doubling of the number of possible filetypes. The original DOS used "I" for Integer BASIC files, "A" for Applesoft, "B" for binary files, and "T" for text files. DOS 3.2 added types "S", "R", another "A", and another "B". Of those four types, only "R" was ever officially designated by Apple, and that for relocatable assembler object files.

DOS 3.2 included a program called "UPDATE 3.2", which worked much like the earlier program "MASTER.CREATE" in changing a "slave" DOS disk into a "master" disk. As time went by, and more users had their Apple II's fully populated with 48K RAM, the need for such a utility became less and less important.<12>

DOS 3.2 - FEATURES

A catalog of the DOS 3.2 System Master disk would produce this output:

```
 *I 002 HELLO
 *I 043 APPLESOFT
 *I 018 ANIMALS
 *B 009 UPDATE 3.2
 *I 014 COPY
 *I 009 COLOR DEMO
 *B 003 CHAIN
 *A 009 COLOR DEMOSOFT
 *A 028 LITTLE BRICK OUT
 *A 003 MAKE TEXT
 *A 003 RETRIEVE TEXT
 *A 010 EXEC DEMO
 *A 010 RANDOM
 *T 003 APPLE PROMS
 *A 039 RENUMBER INSTRUCTIONS
 *A 014 RENUMBER
```

The file "RAWDOS" that was on the DOS 3.1 disk was no longer needed, as its function was included in the "UPDATE 3.2" program.<10> As you can see, some of the files from the DOS 3.1 master disk were retained, but some others were added. There were now several Applesoft files, including a version of the color demonstration ("COLOR DEMOSOFT"), a smaller version of the older Integer BASIC game "BRICK OUT" ("LITTLE BRICK OUT"), a couple of files to show simple disk access ("MAKE TEXT" and "RETRIEVE TEXT"), and a program to exhibit the use of random-access disk files ("RANDOM", with the file "APPLE PROMS"). There was finally a program ("EXEC DEMO") that showed how to use the EXEC command in DOS. Also

found on this disk were two utilities for Applesoft. One made it possible to renumber Applesoft programs, and the other ("CHAIN") allowed linking between multiple Applesoft programs, retaining the value of any variables created by the first program. There was a CHAIN command built into DOS, but it worked properly only with Integer BASIC programs.

DOS 3.2.1

In July 1979, DOS 3.2.1 was released. This was merely a minor upgrade to make some patches to RWTS and correct a timing problem that caused the utility "COPY" to fail when copying disks with two disk drives. It also began a system disk version numbering system that persists to this day, that of adding a third digit to indicate a minor upgrade. (For example, GS/OS 5.0 changed to 5.0.1 with some bug fixes, rather than 5.1).<12>

This disk contained the new COPY program, and a program called "UPDATE 3.2.1", which worked just as "UPDATE 3.2" and "MASTER.CREATE" had previously. The update program was used to modify existing DOS 3.2 disks to the 3.2.1 version. As an bonus, Apple added some programs to this Master disk that were just for fun. All written in Integer BASIC, the games and graphics demonstrations included "APPLE-TREK", "THE INFINITE NUMBER OF MONKEYS", "BRIAN'S THEME", and "BRICK OUT" (which was an Apple II version of the arcade game, "Breakout"). The "HELLO" program displayed this when the disk was booted:

```
 MASTER DISKETTE VERSION 3.2.1 STANDARD

                              31-JULY-79

 COPYRIGHT 1979      APPLE COMPUTER INC.
```

++

NEXT INSTALLMENT: DOS 3.3, ProDOS, & Beyond

++

NOTES

<1> Deatherage, Matt. "The Operating System", The Apple II Guide, Fall 1990, pp. 117-125.

<2> Wozniak, Stephen. (personal telephone call), Sep 5, 1991.

<3> Worth, Don, and Lechner, Pieter. Quality

Software, Beneath Apple DOS, Reseda, CA, 1981, pp. 5.1-5.3, 6.4-6.8, 8.1-8.42.

<4> Little, Gary. Addison-Wesley Publishing Company, Inc, Exploring Apple GS/OS And ProDOS 8, Reading, MA, 1988, pp. 2-4.

<5> Little, Gary. Brady Communications Co, Inside The Apple //c, Bowie, MD, 1985, pp. 1-7.

<6> Auricchio, Rick. (personal telephone call), Sep 4, 1991.

<7> Moritz, Michael. William Morrow and Company, Inc, The Little Kingdom, New York, 1984, p. 211.

<8> Worth, Don, and Lechner, Pieter. p. 1.2.

<9> Bragner, Bob. "Open Discussion", Softalk, Nov 1983, pp. 51-52.

<10> Vanderpool, Tom. GEnie, A2 Roundtable, Mar & Aug 1991, Category 2, Topic 16.

<11> Thyng, Mike. "Apple Source", PEEKing At Call-A.P.P.L.E., Vol. 1, 1978, pp. 7-8.

<12> Worth, Don, and Lechner, Pieter. pp. 2.1-2.3.

APPLE II HISTORY

Compiled and written by Steven Weyhrich

(PART 15 -- DOS 3.3, PRODOS & BEYOND)
[v1.2 :: 30 Sep 92]

DOS 3.3

In August of 1980, Apple released an upgrade for DOS, to version 3.3. This upgrade was an important one. It consisted of not only a new System Master disk, but a hardware upgrade chip as well. The original disk drive had been designed with the ability to read and write 35 tracks of 13 sectors each on a 5.25 inch disk. At 256 bytes possible per sector, this made the disk capable of holding 113.75K of information. Since it was designed to have DOS present on each disk in the first three tracks, and the catalog took up another entire track, there was actually only 100.75K available for data storage. Steve Wozniak, the author of the original DOS disk driver (RWTS), had found a way to increase the storage capacity of Apple floppy disks. Changing slightly the method used for encoding data on the disk made it possible to have 16 sectors per track, instead of the original 13 sectors per track in DOS 3.1 and 3.2. This resulted in a disk that could now hold a maximum of 140K of data (124K excluding DOS and the catalog track), a 23 percent increase over the 13 sector disks. The remarkable thing about this upgrade was that the disk drives themselves did not need to be changed to make this possible. Only the ROM program on the Disk II controller card needed to be changed to make the move to DOS 3.3. Those users who bought this upgrade to DOS 3.3 had to change the ROM chip on the disk controller (or have their dealer do it for them). An updated and greatly expanded version of the DOS manual was also included in the DOS 3.3 upgrade.<1>

DOS 3.3 - FEATURES

The DOS 3.3 System Master disk included many programs that had previously been present on the DOS 3.2 Master, plus a few others. The "COPY" program (used to copy entire disks) was translated to Applesoft as "COPYA" for those II Plus users who didn't have access to Integer BASIC. The newer COPY programs also worked properly on single drive systems (previously, you had to have two disk drives in order to use this program to copy a disk). To allow users to startup their older 13 sector DOS 3.2 disks, a binary program called "BOOT13" was included. (Also, a separate disk called "BASICS" was included that could be used in the

same way as a pre-boot for 13 sector disks).<1>

Because of the changes in the ROM controller, it was not easy to read disks formatted under DOS 3.2 directly from DOS 3.3. It could have been incorporated into DOS 3.3, but would have called for a major effort in rewriting the track and sector access routines, as well as making DOS larger than the earlier versions. Instead, Apple supplied on the System Master disk a conversion program called "MUFFIN" to allow files to be moved from 13 sector to 16 sector disks. Enterprising hackers in the Apple II world made modifications to MUFFIN and created DE-MUFFIN, a DOS 3.2 utility to convert the files back to the 13 sector format.<1> Rich Williams at Apple wrote the MUFFIN program (which was supposed to stand for Move Utility For Files In NewDOS).

The System Master disk also contained a new utility called "FID" (which started at version "M"; just like DOS "3", nobody knows why the first public release didn't start with "A"). FID, written entirely in assembly language, allowed easier copying of files, particularly Text and Binary files that couldn't simply be LOADed and SAVEd from one disk to another, as could Applesoft and Integer programs. The name "FID" was odd, however. The Apple manuals said it stood for FIle Developer, but Rich Williams (who also wrote this utility) said that the original name of the program was FISHEAD (which had some sort of mnemonic meaning that he could no longer recall). Apple Marketing said he couldn't name a program FISHEAD, so he changed it to FID, which they said was okay. It really stood for Fishead In Disguise (or Fishead In Drag by some within Apple).<2>,<3>,<4>

Some Apple II users didn't like to have to use utility programs to manage their collections of disks in both the 13 and 16 sector formats. One method that was used to overcome this inconvenience was to piggyback the old and the new disk controller ROMs and use a switch to toggle between systems. The most elegant solution I've found was a ROM chip that plugged into a special card (the ROMPlus made by Mountain Hardware, or the ROMBoard made by Andromeda). A call to a memory location would switch between DOS 3.2 and 3.3, making file conversions quite easy. Soft Ctrl Systems, the company that sold this Dual DOS ROM also sold ROMs that gave instant access to an Applesoft renumber and merge program, an Applesoft editor, and a specialized disk command menu and disk map.<2>

Another change found on the DOS 3.3 System Master was in the method used to load the alternate BASIC. Since by this time the Language Card was available (which, as you should recall, was simply 16K more RAM to add in parallel to the Apple II ROM), there were two groups of users to service on bootup. For Apple II Plus owners, there was a file named "INTBASIC", which would load Integer BASIC onto the Language Card. For the older Apple II (non-Plus) users, the file "FPBASIC" would be loaded onto the Language Card when the DOS 3.3 disk was booted, making Applesoft available. The

last version of the DOS 3.3 Master disk, released with the Apple IIe, used a new utility to load these files which was significantly faster than the standard DOS BLOAD command.

DOS 3.3 - MISCELLANEOUS

A rumor expressed in a letter to Call-A.P.P.L.E. magazine in January 1982 suggested that up until Christmas of 1980 there never had been an assembly language source listing of DOS. The writer of the letter stated that changes made to DOS up until that time were done by patching it with the mini-assembler in the Monitor. However, during a phone interview in September 1991 with John Arkley at Apple, he said there always was a source code listing for DOS, as far back as DOS 3. He believes the writer of the letter may have been referring to the problem with the lost Autostart ROM source code (see Part 6 of this History). Arkley stated that the earliest versions of DOS were written using a cross-assembler on a Horizon microcomputer.<5>,<6> He also said that the only part of DOS 3.3 that was assembled from scratch was the new RWTS. The rest was merely attached to RWTS and "conditionally" assembled (a programmer's term; sorry). They made a few patches to fix bugs in the File Manager and Main DOS routines, but did so only in very specific places, to avoid moving undocumented entry points that programmers had been using up to that point.<3>,<4>,<7>,<8>

DOS 3.3 - LIMITATIONS

The major limit of DOS 3.3 was that it, like its predecessors, was designed specifically to support the Disk II drive. Hard disks, RAM disks, and 3.5 disks (like those used in the Macintosh when it was released in 1984) could not be directly used with DOS 3.3.<9>

PASCAL SYSTEM

The Pascal system was released in 1979, prior to the DOS 3.3 upgrade. It used the same hardware upgrade to the Disk II controller as was included with DOS 3.3. The method used by the Pascal disk system to store files was quite different from that used by DOS, however. Instead of the 256-byte "sectors" used with DOS 3.2 (and by 3.3), the Pascal system used 512-byte "blocks", using two sectors per block. Pascal used the larger 140K disks from the beginning, and its method of file naming was somewhat more limited. Instead of names that could be as long as 30 characters and could contain any ASCII character (as was the case with DOS 3.2 and 3.3), Pascal files could be only 15 characters long, and could contain only letters, numbers, or a period. It was designed with a little more flexibility in

the types of files that could be created, however. Instead of DOS 3.2's limit of eight different file types ("A", "I", "B", "T", and the other four little used ones), Pascal was designed to allow many more, and used a two-byte code to designate file types. A Pascal file entry also had space for a date when the file was created or updated. DOS 3.2 or 3.3 could not easily do this, even if a clock card was installed.<7>,<10>

Pascal disks differed also in being able to have a unique name to designate each disk. DOS 3.2 and 3.3 could be formatted to use up to 254 different volume "numbers", but this feature was seldom used and did not allow disks to be very unique. The Pascal disk name could be up to 7 characters in length, and had the same limits of character choice as did file names. Another feature of the Pascal disks that differed from the older DOS disks was how space was allocated on a disk for a particular file. Under DOS 3.2 and 3.3, space was used on the disk to identify which sectors were used and which were free. When a new file was created or an existing file was enlarged, this track/sector list was consulted by DOS to find where free space could be found, and the list was updated when a new sector was used. The advantage was that all space on the disk could be used as it was needed, but the disadvantage was that a file could be "fragmented", with the sectors that made up that file scattered throughout the disk.

Pascal disks did not have any map of free blocks. Instead, a Pascal file used only consecutive blocks on a disk, and a new file would be started following the end of the last file on the disk. The advantage of this system was faster access to disk files, since they were all on one continuous piece of the disk. The disadvantage was that if a file was deleted, the newly freed space could not be used unless Pascal's "Krunch" utility was used to move all files forward over the unused space.

The Pascal system also included some other built-in disk utilities, an assembler, and a compiler. As part of this system one could also purchase from Apple a compiler for FORTRAN programs and a few other computer languages.<10>

CP/M

With the release of the Microsoft CP/M Softcard, a disk system was needed to handle this foreign programming environment. (Recall from Part 12 of the History that the CP/M system gave Apple II users a Z-80-based computer inside their 6502 computer and, therefore, access to programs and utilities that were previously unavailable). CP/M disks were designed to use four 256-byte sectors as one "block" (twice as large as the Pascal "block"). Like DOS 3.2 and 3.3, the first three tracks on the disk were used for the CP/M operating system which was loaded into memory when booting the disk. Like Pascal, the CP/M directory was found at the start of the disk, instead of in the middle as DOS

was designed.

Apple II CP/M disks followed the standard CP/M file naming system. A file name consisted of 8 characters, followed by a period, and then a three character "extension". One interesting feature of CP/M files was that if a file was longer than 16 CP/M blocks (64 DOS sectors), a new directory entry would be made with the same file name. This entry had an extra byte set to show that this was a continuation of a previous file, instead of a new, separate file.<10>

SOS/PRODOS

The operating system designed for the Apple III computer was called "SOS". This title arose from the Apple III's code name, "Sara", which itself came from the name of engineer Disk Huston's daughter. Originally, then, SOS stood for "Sara's Operating System". The manuals released with the computer, however, used the more professional-sounding name "Sophisticated Operating System." SOS was the first operating system for a microcomputer to use the concept of "device drivers", which were programs taken from the startup disk and made part of the operating system. These drivers told the computer how to communicate with the various devices that were attached to it, from a variety of disk drives to the keyboard and monitor. This gave flexibility to the Apple III to use new technology as it became available.<9>

When Apple designed the Apple III, they were under constraints of maintaining some compatibility with the Apple II disk format. They used the same disk controller and the same capacity disks as the Pascal/DOS 3.3 systems: 35 tracks, of 16 sectors each. However, the engineers were free to make any changes they wanted in the way in which files were stored on the disk. They came up with something that was a hybrid between the DOS 3.3 and Pascal methods of file storage. From Pascal they took the concept of using 512-byte blocks as the basic unit of storage, a two-block "system loader" program at the start of the disk (this loader would locate a larger system file elsewhere on the disk to actually start the operating system), and a four-block main catalog (which they called a "directory"). From DOS 3.3 they used the concept of disk maps and block lists for each file, allowing parts of files to be stored anywhere on the disk (and eliminating the need for the Pascal "Krunch" function). The SOS filing system also continued the use of a byte to identify different filetypes, space for a date (and time) of file storage, and the 15 character file names using only letters, numbers, and a period. Because the Apple III was intended to be a business machine and had to be able to access larger disk devices than were allowed for the Apple II, they also added the ability to create and use different levels of file directories. A single four-block directory had space only

for 51 files; even if it was enlarged to allow more files, on a large disk it would soon be difficult to find a file in a list that got longer than a couple of hundred names.

The SOS disk file system also would allow files to be as large as 16 MB, and a single disk volume could be up to 32 MB in size. In 1981, when the 5 MB Profile hard disk was released by Apple for the III, this limit of 32 MB was considered to be more than adequate.

In 1984, when ProDOS was released for the Apple II as a "Professional Disk Operating System", the same file structure was used. In fact, the disks were so designed that a disk created by the Apple II ProDOS formatter installed an Apple III SOS loader segment in the second block on the disk. This made it possible to boot the same disk on either an Apple II or an Apple III, if the necessary system files unique to each computer were present on the disk. Also, files could be shared easily between the two computers. Even as late as 1992, when the Apple III has been out of production for eight years, disk formatted by Apple II System Utilities still have SOS boot information located on block 1. What may be even more amazing is that this disk system for the Apple III, released in 1980 (and probably designed in 1978 or 1979), is still flexible enough to be useful for Apple II's in 1992.<10>

PRODOS

The original DOS for the Apple II was designed primarily to support BASIC. If a programmer wanted to make use of the disk system for an assembly language program, he had to make use of undocumented, low level calls to the DOS File Manager, or possibly to some of the Main DOS Routines. This method was clumsy, and often made inefficient use of memory, as DOS expected that any calls made to it were done on behalf of BASIC. Moreover, this tied the hands of programmers at Apple in their ability to enhance DOS, since any changes they might make would most likely change internal addresses, and cause older software to malfunction if used with the revised DOS.

Another problem with DOS was speed. Since each byte read from the disk was copied between memory buffers three times, much of the disk access time was spent in moving things around in memory. Consequently, as hackers took DOS apart and found better ways to do things, several variations of DOS speed-up programs appeared by 1983, including Diversi-DOS, ProntoDOS, and David-DOS. Each of these programs were mutually incompatible in terms of the low-level calls they made, and had slightly different ways of speeding up DOS.

DOS was also limited since it was device dependent. It was designed to work quite well with the Disk II drive, but to make use of a hard disk or RAM disk (a pseudo-disk "drive" that was actually RAM memory, had no moving parts, and was therefore quite fast), DOS had to be patched. This

usually made it impossible to use different brands of hard disks together, or to use a hard disk and a RAM disk simultaneously.

Other problems with DOS included poor support for interrupt signals generated by various hardware devices, obstacles in designating memory areas as protected from being overwritten by DOS, and the difficulty in customizing DOS for special functions.

With the introduction of ProDOS, all of these weaknesses were addressed. ProDOS would run up to eight times faster than DOS in accessing 5.25 disks. It supported a standardized protocol for hardware-based devices, allowing reads, writes, status calls, and formatting (erasing). This allowed a large variety of disk devices to be used on an Apple II. Support was also included for a hardware clock, allowing date- and time-stamping of files. Hardware interrupts were supported, necessary system calls were placed in a standard location in memory (called a "global page"), and memory could be protected from being overwritten by the actions of ProDOS.

Because the functionality of this disk operating system was enhanced so much, its size grew as well. To specifically support Applesoft BASIC, a separate "SYSTEM" program was included that worked nearly the same as the older DOS 3.3 did. In addition, it included some further enhancements that had been requested for years by Applesoft programmers. The only disadvantage of the new ProDOS was that it did not support Apple's original Integer BASIC, since the ProDOS program loaded itself into high memory where Integer BASIC was loaded in an Apple II Plus. Since very little development of software had been done in Integer BASIC since the introduction of Applesoft, this was felt to be a reasonable trade-off. And if Integer BASIC was needed, it could still be run under DOS 3.3. At the time of this writing, there has been no release of a ProDOS system program that would support Integer BASIC (with the exception of an Integer BASIC compiler distributed by ByteWorks in late 1991 for instructional purposes).<1>

PRODOS 16

When Apple released the IIGS, with its considerably greater power compared to the older 8-bit Apple II's, changes were needed in the operating system to better manage that power. This had to be done with another goal, that of maintaining compatibility with older Apple II software. The new operating system was called ProDOS 16, and the operating system intended for use with 8-bit software (both on the IIGS and on the older Apple II's) was renamed ProDOS 8. But ProDOS 16 version 1.0 was somewhat of a temporary fix to the problem of disk access for 16-bit software. It was not written in 16-bit code, and it simply translated the new system calls defined for ProDOS 16 into ProDOS 8 calls to actually carry out disk activities. As such, it was slow

and cumbersome.<9>

GS/OS

With the experience of SOS, ProDOS, and the Macintosh Operating System to draw from, Apple engineers and programmers devised a yet more powerful and flexible disk operating system for the Apple IIGS. Written completely in 16-bit code, GS/OS was released in September 1988. It was more than a disk operating system, but a truly comprehensive operating system that also handled keyboard input, monitor output (text and graphics), mouse input, printers, modems, and more. In these respects it was just as powerful as the older SOS written for the Apple III back in 1980. But they also added a new concept.

Although GS/OS would allow an Apple IIGS to communicate with disk devices that had not been used on an Apple II before, there would still be the limits of having to know exactly how files were stored on that disk. ProDOS could only handle files stored in the specifically defined ProDOS/SOS format; DOS 3.3 could only handle files stored in that format; and so on. To make this new system as broad-based as possible, Apple programmers built into it the concept of a File System Translator (FST). With the appropriate FST teamed up with a suitable disk driver, GS/OS could theoretically be able to read any disk created by any computer. The FST simply translated the requests made by GS/OS into the language "spoken" by the disk it was trying to read. This task had never before been attempted by a computer company in designing a disk operating system. Apple, recognizing that the computers used in the real world would never be 100 percent Apple, made it possible to simplify transfer of data between different computers. The concept was first implemented in a limited fashion on the Macintosh, when the Apple File Exchange program was modified to be able to use MS-DOS disks. The Mac system is now also able to add its equivalent of an FST for the ProDOS and MS-DOS disk systems, but not as easily as has been implemented in GS/OS.

GS/OS was also made more flexible by removing the older Apple II method of identifying a disk by the slot where its disk controller was attached, and removing the limitation of only two disk devices per slot. The limits of maximum file and disk size built into ProDOS 8 were expanded. A GS/OS file or disk volume can be as large as 4 GB (gigabytes), or 4096 MB to be more specific. However, when GS/OS is dealing with ProDOS disk volumes, it still has to stay within the limits of ProDOS (files no bigger than 16 MB, and disk volumes no bigger than 32 MB).<9>

System Software 5.0 for the IIGS was introduced in May 1989. It added speed, speed, and more speed to many features of the IIGS, accomplishing this through more efficient software coding. There were patches to the IIGS ROM Toolbox to improve throughput in many of the built-in

capabilities of the machine. A new feature called "Expressload" was added, making it possible for certain program files to load from disk up to eight times faster. GS/OS was modified to be capable of staying in memory during a switch to ProDOS 8 applications, making the return to GS/OS significantly faster. The text-based control panel was supplemented by a new graphics-based one that was accessible in the same way as other 16-bit desk accessories. Access to 3.5 disks was accelerated by implementing a feature called "scatter read", which could take an entire track (rather than just a single block) of data from the disk at a time. An FST for AppleShare was added, allowing a IIGS attached to an AppleTalk network to access the file server as a disk. It also included an FST to allow access to CD-ROM drives, new utilities for disk partitioning, and it had an intelligent "Installer" program to make it easier to install system or application files.<11>,<12>

Because of further improvements in features, System Software 5.0.2 (an upgrade to 5.0) required a minimum of 512K memory, and worked best with 768K or more. Versions 5.0.3 and 5.0.4 needed a full megabyte of memory.<9> An improved "standard file dialog" was included in the system tools for 5.0.3, (making it possible to choose files more easily for loading into an application), as were improved drivers for the ImageWriter II and ImageWriter LQ printers. System 5.0.4 was released six weeks after 5.0.3 to fix some remaining important bugs discovered too late.<12>

GS/OS SYSTEM 6

Before System 5.0 was released, plans were already in store for further improvements to the system software. Apple IIGS "power" users were calling for the ability to use Macintosh HFS (Hierarchical Filing System) disks, as well as the older Apple II DOS 3.3 and Pascal formats. Although there were some simple third-party translation programs available that allowed transfer of files from Mac disks to ProDOS disks, they did not provide the same ease of use as did the direct access possible with ProDOS and CD-ROM files. Although it sounded to these users like a relatively straightforward proposition, the increased complexity of the Mac HFS directory structure complicated things. Not only did the Mac disks contain more information about each file than did ProDOS disks, but the names of files on Mac disks (as on DOS 3.3 disks) could contain characters that were not "legal" for ProDOS file names. To help with this problem, the new FSTs were designed to watch for potentially illegal filenames, and to make suggestions for alternate names that were legal.

Apple software engineers had always made it clear to programmers clamoring for additional FSTs that such changes were more than just dropping the new FST into the System/FST folder on a boot disk. Modifications were necessary throughout GS/OS to accommodate these new features, and the

time needed to make these changes was becoming longer than originally planned. To allow some improvements to be made available without waiting for them all, the system software engineers divided tasks during 1990, putting the features that could be programmed most quickly onto a fast track that would allow them to be released as Version 5.0.3 later that year.

The other half of the team worked on the rest of the planned enhancements for what would become System 6.0. When 5.0.4 was completed, the entire team again came together to continue work on this upgrade. After fourteen months of hard work, they were finally ready to release GS/OS System 6.0 in March 1992. In addition to FSTs for the Mac HFS disks, DOS 3.3, and Apple Pascal, device drivers were created to allow support of the Apple Scanner, the slot-based Apple II Memory Expansion card (which on the IIGS works primarily as a RAM disk), and the Apple Tape Drive. The SCSI drivers were enhanced, and the Apple 5.25 disk driver was made faster. A new printer driver was included, to support the Apple StyleWriter inkjet printer, and more large fonts were included to use with that and other printers. The Finder was re-designed almost from scratch by Andy Nicholas, the author of ShrinkIt and GS-ShrinkIt. Archiver (a disk backup utility) and Teach (a GS/OS-based text-editing program) were also included. Finally, ProDOS 8 v2.0.1 was released, allowing 8-bit programs access to as many as fourteen disk devices on a single slot. This made large, partitioned hard disks usable even to Apple IIc and enhanced IIe users (this version of ProDOS 8 required the opcodes of the 65c02 chip, although ProDOS 8 v1.9 was still available to run on the Apple II Plus or unenhanced IIe).<12>

At the 1992 KansasFest, Apple engineers announced that v6.0.1 of GS/OS would be out later in 1992 or early in 1993. Along with specific support of the Apple II Ethernet card, this version is expected to include bug fixes found in 6.0, and an MS-DOS FST (at least read-only, with write capability to come later).

++++++++++++++++++++++++++++

NEXT INSTALLMENT: Languages

++++++++++++++++++++++++++++

NOTES

<1> Worth, Don, and Lechner, Pieter. Quality Software, Beneath Apple DOS, Reseda, CA, 1984, pp. 2.1-2.9.

<2> -----. (ads), Call-A.P.P.L.E. In Depth #1,

1981, p. 106.

<3> Auricchio, Rick. (personal telephone call), Sep 4, 1991.

<4> Wozniak, Stephen. (personal telephone call), Sep 5, 1991.

<5> Roberts, Henry. "A.P.P.L.E. Doctor", Call-A.P.P.L.E., Jan 1982, p. 63.

<6> Arkley, John. (personal telephone call), Sep 9, 1991.

<7> Little, Gary. Addison-Wesley Publishing Company, Inc, Exploring Apple GS/OS And ProDOS 8, Reading, MA, 1988, pp. 2-4.

<8> Little, Gary. Brady Communications Co, Inside The Apple //c, Bowie, MD, 1985, pp. 1-7.

<9> Deatherage, Matt. "The Operating System", The Apple II Guide, Fall 1990, pp. 117-125.

<10> Hunter, Skillman. "Road Maps To Apple II Disks: DOS 3.3, CP/M, Pascal, and ProDOS", Call-A.P.P.L.E., Feb 1985, pp. 10-21.

<11> Weishaar, Tom. "Breaking the incompatibility barrier:An introduction to Apple's GS/OS", Open-Apple, Nov 1988, pp. 4.75-4.78.

<12> Deatherage, Matt. "The Operating System", The Apple II Guide, 1992, pp. 111-113.

APPLE II HISTORY

Compiled and written by Steven Weyhrich

(PART 16 -- LANGUAGES)
[v1.0 :: 22 Jan 92]

PROGRAMS "R" US

Nearly everyone reading this is already a programmer, on one level or another. Even if you don't know a "GOTO" from a "STA $C030", you already know how to program something. For the act of "programming" is nothing more than giving instructions to a non-human device to have it carry out what you want it to do. The device that most of you already know how to program is your automobile. The act of giving those instructions may not seem like programming to you; nevertheless in its strictest sense, programming it is. You want the car to go forward? Set the transmission to "D". Go in reverse? Use "R". Of course, the programming needed to operate an automobile is quite simple, and cannot be done in more than one step at a time. An example of a device that is more complicated to program but does let you store up several instructions in advance is a VCR. On the VCR you instruct it to record a television broadcast that starts at 7:00 pm and ends at 8:30 pm, on channel 6. The more sophisticated VCR's can have several programs set up in advance. If you can operate a VCR in this fashion (which is, admittedly, not always as easy as I have described), you are a programmer.

When it comes to the microcomputer, the process of programming (giving it instructions on how to carry out a task) is somewhat more complicated. This is primarily because the computer is far more flexible in its ability to accept instructions and carry them out than is an automobile or VCR. Devices attached to a computer can be manipulated by a program to do something useful (print a letter several times, or perhaps read the outside temperature and sound an alarm if it drops too low). This flexibility, plus the speed at which a computer can execute its instructions, makes it a powerful tool for doing things that have previously taken much more effort and time. And as a project becomes more sophisticated, so also must the programming acquire a similar level of sophistication. The rate at which computers, including the Apple II, have increased in capacity during the past fifteen years has made it possible to design programs that can do things that were not even dreamed possible back in the days of the 4K Integer BASIC machine.

An example of programming evolution on the Apple II was given during Kansasfest in July of 1991. To fully

appreciate this narrative, you need to know a little about an old Integer BASIC program, APPLEVISION. This was found on the DOS 3.2.1 System Master disk, and was a fun little display that showed off the use of hi-res graphics. It began by creating a simple line drawing of a room, with a picture on the wall ("HOME SWEET HOME") and a television set. On the screen of the TV appeared a man who danced to the tune of "Turkey In The Straw", which sounded on the built-in speaker. It ran repeatedly, until the user interrupted the program. It was fascinating at the time, since there was nothing in the program text that showed off exactly how the hi-res effects were accomplished. But things have gotten a bit more complex as time has gone by:

> "Roger Wagner's keynote address featured a history of hypermedia which Roger set into action and left to run as he wandered offstage. The history began with Bob Bishop's classic AppleVision, done in black and white on the original Apple II. Progressive screens enhanced the AppleVision image using subsequent incarnations of Apple II graphics (single hi-resolution, double hi-resolution, and the IIGS's Super Hi-Resolution modes). Finally, thanks to a laserdisc player under HyperStudio's control and a video overlay card, Roger's image appeared within the television's screen and spoke to the audience, completing the introduction before turning the presentation back to Roger (returning from offstage)."<1>

To follow the programming progress that has made such magic possible, we will begin with the first two built-in high-level languages for the Apple II, Integer BASIC and Applesoft, and move on to a briefer discussion of some of the other languages that have been available over the years. Next will be a summary of various 6502 and 68816 assemblers that Apple programmers have used over the years. Finally, I will present an introduction to "hyper-programming".

FUNDAMENTALS OF PROGRAMMING

A programming language has the standards to translate "what I want" into commands that the computer understands. To do so, it must take some human language and convert it into the binary dialect of the computer on which it is executed.

Computer languages usually come in one of two different types: "interpreted" and "compiled". A language that functions as an interpreter takes the text of the program and translates it at the time of execution into commands the computer can understand. A compiled program, on the other hand, has already had the program text

translated into executable code before it is run, usually including some extra code needed to carry out necessary functions of input, output, and calculations. As such, an interpreted program usually runs more slowly, but has the advantage of being easier to modify and re-run without the delay of first re-compiling. A compiled program will ordinarily run faster, but may use more memory than an equivalent interpreted program.

Languages are also given the designation of being "high-level" or "low-level", depending on how close they are to the base language of the computer on which they run. The lowest level of computer programming is at the level of the bytes understood as commands by the microprocessor. This "machine language" is typically not very understandable to humans. A low-level language more often used by programmers is "assembly language". This uses commands somewhat more understandable ("LDA $24" means "load the accumulator with the contents of memory location $24") and are then assembled (actually compiled) it into machine-readable code. Assembly language is very powerful, since it works on the byte level of the computer. However, as a low-level language it can be very complicated and requires an intimate understanding of the function of the computer.

As a language becomes more "high-level", it is easier for humans to read, but requires more effort from its interpreter or compiler to translate it into the native language of the computer.

INTEGER BASIC

This was the first language available for general use on the Apple II (aside from assembly, which will be dealt with later). Most of the details concerning its development have already been covered in Part 3 of this History. It was a quick, compact language, and its creation was an example of programming directly in machine language (since Steve Wozniak, the author, had no assembler available to use). Its disadvantage was the lack of easy access to floating point operations, and it lacked some string handling functions. Apple II users, especially those who wanted to produce programs that could be used in business applications, wanted something more powerful to use.

Despite its limitations, Integer BASIC was a language that had a fanatically loyal following. For those thousands who purchased Apple II's from June 1977 to June 1979, this was the only programming language available, and it took on a status similar to that of a beloved first-born child. Games, utilities, and even some simple business-use programs were written using Wozniak's hand-assembled masterpiece, and those who followed the pages of Call-A.P.P.L.E. magazine learned much about the internals of the language. With the disassembler built into the Monitor, people tore Integer BASIC apart to learn how it worked, and to make it work better. Val Golding, the editor of Call-A.P.P.L.E., even

wrote a series of columns in 1979 entitled "So Who Needs Applesoft?" These articles showed how to simulate some of the more advanced features of Applesoft in this older BASIC. A.P.P.L.E. even sold (under license agreement with Apple Computer) "Integer BASIC +", a relocatable RAM version of the original ROM BASIC. It had all the features of the original language, plus a "USER" command, the ability to easily do four direction scrolling on the text and lo-res screens, easy printing of ASCII characters, and improved error handling.<2>

Apple never released a comprehensive reference manual for Integer BASIC. The only manual available for it was primarily a tutorial (and a general introduction to using a computer). The "Apple II BASIC Programming Manual" didn't even call it "Integer BASIC", but referred to the language as "Apple BASIC". It gave most of its programming examples in the form of segments of a graphics and sound demo that created a lo-res ball bouncing off the sides of the screen.<3>

With the many programs available that were written in Integer BASIC, it was almost a necessity for Apple to offer a means for Apple II Plus users to be able to run the older software. The Integer Firmware card made this "backward compatibility" possible. This was especially important in the early days of the II Plus, when there was little new software available to use with Applesoft.

APPLESOFT I

Although Wozniak had written some floating point routines into the Integer Basic ROM, Apple II users needed a version of Basic that would make floating point math easier to do, particularly for business use (where the number to the right of the decimal point is as important as the one to left). Apple decided to license a 6502 version of a floating point BASIC from Microsoft Corporation. Back in 1977, Microsoft was producing BASIC interpreters for nearly every microcomputer that was produced. The version Apple purchased was almost identical to the MITS extended BASIC that Microsoft had previously written for the Altair 8800.<4>,<5>

This BASIC was named "Applesoft", and was released in November of 1977 on cassette. It was loaded as a 10K program that looked to the computer just like an Integer BASIC program, though only a small part of it really was. To make it easy to load and start from cassette, the Applesoft interpreter was attached to the end of a short Integer BASIC program. When the Integer program was run, it poked some values into memory and jumped to the start of the machine language section, which relocated the Applesoft interpreter to the lower part of memory (at $800), just after the memory that held the screen display.

Using this version of Applesoft (which later became known as Applesoft I) could be frustrating. It took several

minutes to load from the cassette tape, and it was not dependable. If the wrong key was pressed while entering or running an Applesoft program, the program that was being run could be wiped out, and the Applesoft interpreter itself would have to be reloaded from cassette. However, few users knew how to make use of the floating point routines that Wozniak had written into the Integer ROM, so this unreliable Applesoft BASIC became the only practical means of doing floating point math on the Apple II.

Aside from the reliability issue, another difficulty with Applesoft involved hi-resolution graphics. Although the Apple II was capable of displaying it, the Applesoft interpreter extended up into the memory used by the hi-res screen, and so prevented its use. Furthermore, this early version had no built-in commands to manage hi-res graphics.<5>

Applesoft I came with a manual that was 8 1/2 inches by 11 inches in size, and sported a blue cover with square glued binding.<6> This came to be known as the "blue book" (recall that the reference book for the computer itself was affectionately known as the "red book"). When starting the interpreter after loading it from the cassette, a screen was display announcing that Applesoft was copyright 1977 by Apple and Microsoft. It then asked the user for the memory size of his computer, and gave options of allowing either LET and REM statements or the use of lo-res graphics. The names of the lo-res graphics commands were very different from those that existed in Integer BASIC (and in the later versions of Applesoft). The commands were:

```
PLTG           = Go to lo-res graphics mode
TEX            = Go to text mode
PLTC N         = Set color to N (0-15)
PLTP X,Y       = Plot square at X,Y
PLTH X1,X2,Y   = Plot horizontal line from X1 to X2 at Y
PLTV Y1,Y2,X   = Plot vertical line from Y1 to Y2 at X
```

There was a note about these commands in the reference card included with Applesoft I that warned about using graphics coordinates only between 0 and 39, or a program could "self-destruct". Apparently it lacked the error checking that could prevent the plotting of lines from spilling over into the text of the Applesoft program itself.<6>,<7>

The A.P.P.L.E. user group published a patch in 1978 that allowed programmers to avoid the question about using LET and REM statements versus lo-res graphics, and use the graphics only. The author of the patch pointed out that the LET statements were not necessary ("A = 3" worked just as well as "LET A = 3"). The REMark statements could be simulated by putting them at the end of a GOTO line (where they were ignored by the interpreter), and the GOTO could just jump to the following line:

```
530 GOTO 540: REM LINE 540 SETS VARIABLE N.
```

```
540 N = 2
```

Additional patches were made available for some of the other bugs found in Applesoft I.<8>

APPLESOFT II

In spring 1978, Randy Wigginton and some others at Apple made some needed revisions to Applesoft. Using a cross-assembler running on a North Star Horizon (Z-80) microcomputer, they fixed the known bugs and added other commands to control features unique to the Apple II. These commands included the ones needed to draw and manipulate hi-res graphics. Also, the lo-res graphics commands were renamed to be more consistent with the equivalent commands in Integer BASIC (GR, HLIN, VLIN, etc.) This version was called "Applesoft II", and eventually it was available in five forms: Cassette RAM and Diskette RAM (which loaded to the same memory locations that interfered with hi-res graphics as did Applesoft I), Firmware card ROM, Language card RAM, and finally main board ROM (in the Apple II Plus).

When Applesoft II was started up from cassette or diskette versions, the display screen now showed a copyright date of 1978 by Apple Computer, Inc., and 1976 by Microsoft (which may be either their copyright date for the original Microsoft BASIC, or possibly for Microsoft's first 6502 version).<6> This RAM version of Applesoft II used memory from $800-$2FFF, and the Applesoft BASIC program itself was loaded beginning at $3000. When the versions that came on ROM and for the Language Card RAM were released, the BASIC program could load at $800, and much more memory was available for it. Some of this extra space (in high memory) was reclaimed by DOS when the Disk II was released, however.<5>

Applesoft in the original IIe was unchanged from the II Plus version. When the IIc was introduced in 1984, however, Apple programmers had cautiously made a few useful changes to the language:

- Input processing was changed to allow lowercase entry of Applesoft commands (they were translated into uppercase)
- Screen output commands (PRINT, TAB, HTAB, etc.) were modified to more properly handle the 80-column screen
- Program lines (when LISTed) were changed to begin in column 2, making screen editing easier
- All of the cassette tape routines (LOAD, SAVE, SHLOAD, STORE, and RECALL) were removed, since the hardware did not support cassette I/O. The keywords were still in the token table, but now pointed to the same memory vector as the ampersand ("&") command.
- Patches were made to the lo-res graphics commands (GR, HLIN, VLIN, PLOT, and SCRN) to work with double lo-res graphics. However, a bug was introduced that

allowed PLOTting vertically to areas outside of the double lo-res graphics screen, which would land right in the beginning of the $800 space where the Applesoft program text was located (similar to the "plot" bug in Applesoft I).

When the Apple IIe Enhanced ROMs were made available, Applesoft in those ROMs had undergone some similar modifications. All the above IIc changes were added, with the exception that double lo-res graphics capability was not added (lack of ROM space), and the cassette I/O commands were not removed (since the cassette input and output port was still present).

The version of Applesoft on the Apple IIGS closely resembled the Apple IIc variant, the only exception being a fix of the double lo-res PLOTting bug. However, a bug in the SCRN function that applied to double lo-res mode was not fixed. No changes to Applesoft from the IIc version appeared in the Apple IIc Plus.<9>

The manuals written for Applesoft II were far more comprehensive than either the older "Blue book" or the Integer BASIC manual. It gave not only programming examples for each of the commands, but included much more information about the various ways in which each Applesoft statement could be used. It also mentioned some of the differences between Applesoft and Integer (for those who wanted to convert their older programs), and gave a little information about the internals of Applesoft to aid in creating machine language additions to the language. Curiously, the manuals that have been reprinted even as late as 1990 by Addison-Wesley have included an odd cautionary note to programmers. In a section in the index about "reserved words" (words reserved as Applesoft commands), it advises against using "XPLOT" as a variable name, stating that "it is a reserved word that does not correspond to a current Applesoft statement." What is apparently meant by this comment is that at one time Apple intended to extend the language and add another command "XPLOT" to it, probably working with HPLOT in the same way that XDRAW complements DRAW in doing hi-res graphics. Examination of the command table within the Applesoft interpreter shows thereis no entry labeled "XPLOT", and a disassembly of the interpreter shows no preliminary code to support the command. Somehow this precaution persisted to the present day and has never been removed, even though it is extremely unlikely that Applesoft will ever be upgraded.<10>

Particularly helpful for programmers was the foresight to include a simple extension called the "ampersand hook". If Applesoft came across the "&" symbol while interpreting a line, it jumped to a known location in memory and left it to the programmer to insert the correct code to add a machine language extension to the language. With the publication of important information about the internals of Applesoft in 1980, assembly language programmers could now add statements to do things that could not be done with the language as it

was originally created. Music, extended graphics, IF-THEN-ELSE logic, and even the missing "XPLOT" command could be added to the language. The only limits were the author's imagination (and available memory).

The importance of Applesoft as an influence to productivity on the Apple II cannot be overstated. Since the release of the Apple II Plus in 1979, every variety of Apple II has contained Applesoft in virtually an unchanged form. This has made it possible for anybody to write programs that all other Apple II users will be able to use, since the language does not have to be purchased or added. If there were thousands of Integer BASIC programs from the two years when Integer Apple II's were produced exclusively, there are hundreds of thousands of Applesoft programs that appeared over that subsequent thirteen years. Even today, it is not uncommon for an applications program to include a configuration module written in Applesoft using the disk commands available with BASIC.SYSTEM in ProDOS. It is often faster to write such a program in BASIC, and the author knows without a doubt that his customer will be able to run it.

APPLESOFT 3 (?)

In 1979 there were rumors at the West Coast Computer Faire about an enhancement to Applesoft II that was in the works at Apple. It would possibly be called Applesoft 3, and would be as much of an enhancement over Applesoft II as that version was to Applesoft I. Supposedly it was intended to merge DOS and BASIC, and would include such powerful functions as IF-THEN-ELSE, PRINT USING, WINDOW, and VIEW PORT. It was predicted to be a RAM version only, and would be about 24K in size. Knowing the events that actually followed, this rumored BASIC was probably the "Business Basic" released with the Apple III, rather than an enhancement for the Apple II.<11>

++++++++++++++++++++++++++++++++++

NEXT INSTALLMENT: Languages, cont.

++++++++++++++++++++++++++++++++++

NOTES

<1> Doms, Dennis. "KansasFest 1991", A2-Central, Sep 1991, p. 7.57.

<2> -----. (ad), PEEKing At Call-A.P.P.L.E., Vol 2, 1979, p. 62.

<3> -----. Apple II BASIC Programming Manual, 1978,

1979, 1980, 1981.

<4> Chien, Philip. "The First Ten Years: A Look Back", The Apple II Review, Fall/Winter 1986, p. 12.

<5> Golding, Val J. "Applesoft From Bottom To Top", Call-A.P.P.L.E. In Depth #1, 1981, p. 8.

<6> Bernsten, Jeff. GEnie, A2 Roundtable, Apr 1991, Category 2, Topic 16.

<7> Arkley, John. (personal telephone call), Sep 9, 1991.

<8> -----. "Apple Patches", PEEKing At Call-A.P.P.L.E., Vol 1, 1978, p. 40.

<9> Weyhrich, Steven. "Applesoft Miscellaneous Information", Applesoft Concordance v1.0, Dec 1989.

<10> Kamins, Scott. "Appendix D Reserved Words", Applesoft BASIC Programmer's Reference Manual, 1982, 1983.

<11> Aldrich, Darrell. "The Computer Faire And The Apple", PEEKing At Call-A.P.P.L.E., Vol 2, 1979, p. 158.

APPLE II HISTORY

Compiled and written by Steven Weyhrich

(PART 17 -- LANGUAGES, CONT.)
[v1.0 :: 22 Jan 92]

APPLE PASCAL

Applesoft was easy to use because it was interactive. You entered a command, and could immediately try it out. The disadvantage was a lack of more powerful commands, and it could be difficult to create large and complex programs. Efforts were begun within Apple to develop a more comprehensive language for the II, one that could be updated and modified if necessary. Since Applesoft was in ROM, it was more expensive and difficult for the end-user to install any upgrades to that language.

In 1979 Apple Pascal and the Language System was released. It sold for the steep price of $495, and came on four 5.25 floppy disks (all in the format of the Pascal disk system, of course). It also included the ROMs to change 13 sector disk controllers into 16 sector controllers, and the Language Card to plug into slot 0. As discussed in previous segments of this History, the Language Card was a 16K RAM card that made an Apple II into a full 64K RAM computer. Because of the extra available RAM, the Pascal system could load into memory without having to avoid the space used by the Applesoft (or Integer BASIC) interpreter. And with some complicated bank switching, even routines in the Monitor could be used if needed.

Apple chose to use the Pascal standard defined by the University of California at San Diego (UCSD). To make portability between various different computers possible, UCSD Pascal programs were compiled into a specialized code called "P-code". This "P-code" program could then be executed on any computer that had a proper interpreter. An Apple Pascal program could, then, run a little faster than an Applesoft program (since it was compiled), but not as fast as assembly language. The extra power it provided made it an attractive choice for some programmers.

The earliest version of Apple Pascal got complaints from users because it would not support lowercase (for those who had modified their Apple to display lowercase), and it was so large that it was quite awkward to use by those who owned only one disk drive.

Since the original UCSD Pascal language was designed to work with a full 80 columns of text, this was somewhat of a problem for the 40-column Apple II. For those Apple II's that did not have an 80-column card, Apple Pascal would display half of the screen at a time. In the Pascal Editor,

entry of a line longer than 40 columns would cause the screen to scroll to the left. Using the arrow keys to move back to the left would scroll the screen back the other way. If needed, you could jump directly to the other half of the screen by pressing Ctrl-A.<1>

The limitation of Apple Pascal came from the need for a user to own the Language Card (or one of the later equivalent 16K RAM cards), and the fact that it was incompatible with the large library of DOS 3.2 programs and files that were already available. Eventually, with the proliferation of the 64K Apple IIe and 128K Apple IIc, a platform for Pascal applications was available. However, by that time the primary disk system being promoted by Apple for the II was ProDOS, and Apple never officially released a version of their original UCSD Pascal that would run under that operating system.

The Apple Pascal system has evolved up to version 1.3, which will support the more advanced features of the Apple IIe and IIc, but does not work as well with the IIGS as some would like. Instead, IIGS programmers now have versions of Pascal distributed by third party companies (like ORCA/Pascal from ByteWorks) created to take full advantage of that machine in 16-bit mode.

INSTANT PASCAL

This version of Pascal was written by Think Technologies, and Apple later bought the rights to sell it as a program for teaching Pascal. It would run only on the Apple IIc or on a 128K IIe because it used the double hi-res graphics mode, functioning much like a Macintosh "desktop" with multiple resizable windows. It had a mouse-based editor that checked program syntax as each line was entered (as did the older Integer BASIC) and automatically indented lines and boldfaced Pascal reserved words. Since it was intended for teaching, it also had a single-step trace function and the ability to modify the contents of variables while a program was running. Though good for learning the language, it was quite slow because of the overhead needed to display everything in graphics, and because it was an interpreted version of Pascal (instead of a compiled version).

Fans of the original Apple Pascal complained loudly after Apple introduced Instant Pascal. After this new Pascal came out, Apple didn't seem motivated to make any further upgrades to the older Pascal, which still used the original Pascal disk system format (Instant Pascal was made to run directly under ProDOS).<2>

FORTRAN

Released by Apple in 1980, Apple FORTRAN ran under the Pascal operating system. It cost $200 (over and above the

$495 needed to get the Language System). Programs written in FORTRAN for other computers could run with little modification under Apple FORTRAN (if a user needed that ability). As a compiled language, it ran faster than Applesoft, and probably also faster than Pascal, since FORTRAN wasn't translated into an intermediate "P-code". Apple's FORTRAN had many bugs in it, though, and since its introduction in 1980 it was never upgraded. By September 1986 it had disappeared from their product catalogs.

Another way for an Apple II user to get FORTRAN was to buy the Microsoft Z-80 Softcard for $345 and Microsoft FORTRAN for $200. This version of FORTRAN was more full-featured than Apple's, and offered some advantages in usability. It did not require changing to the 16 sector disk controller ROMs (if you didn't want to). Also, standard Microsoft BASIC (which was more advanced than Applesoft) was included in the Softcard package.<3>

In June of 1987 Pecan Software released FORTRAN for the IIGS. It ran under ProDOS 16 (GS/OS), but still used the UCSD format for its FORTRAN by creating a ProDOS file that acted as a UCSD volume.<3>

OTHER LANGUAGES

PILOT: Designed primarily for creating tutorial modules, this language allowed educators to design interactive programs to instruct students and test them on their responses during the process. One early version was written in Applesoft and was text-based. Apple later released their own version that ran under the Pascal system for $125.<4>

FORTH: This was a interesting language described as "extensible". It had a number of built-in commands, and new ones could be added as easily as typing its definition. These added commands could then be used in larger programs. Two versions sold in the late 1970's were "Apple Forth 1.6" (Cap'n Software) and "6502 Forth 1.2" (Programma International). Apple Forth 1.6 was a good package, but it used a unique disk system that was not compatible with DOS 3.2. Programma's Forth was more extensive, but also more complicated.<5>,<6>

LOGO: Developed from LISP (LISt Processing) language to be an aid for learning, Logo has been popular over the years in the school environment. Apple's first version of Logo (which operated under the Pascal system) could run on any 64K Apple II, while Apple Logo II (released in July 1984 for $100) ran under ProDOS on Apple II's with 128K memory.<7>

COBOL: This language has had limited availability for the Apple II. The only version I am aware of was from Microsoft. It sold for $599 and ran under the CP/M system with the Microsoft Z-80 Softcard.<8>

C: A language that is currently popular among "power" programmers. It has some of the structure of Pascal, but

also some of the low-level power of assembly language.

ASSEMBLERS

A large variety of Apple II assemblers have been available over the years. The earliest one, of course, was the mini-assembler that came with every Integer BASIC Apple II. The one was only good for immediate entry of assembly code; if changes were needed, much of the code would likely have to be re-entered from the beginning. Some other assemblers available in the early days include:

TED/ASM: Developed at Apple and smuggled out the doors around May 1978, this assembler had memory conflicts with DOS, so they couldn't be used together. The text editor module was written by Randy Wigginton, and the assembler was written by Gary Shannon. In the early days, it was the only assembler they had available that would run on an Apple II.<9>

RANDY'S WEEKEND ASSEMBLER: Also written by Randy Wigginton, this one slipped out of Apple in September 1978. The text editor was written mostly in SWEET-16 (Wozniak's 16-bit emulator in the Integer BASIC ROM), and was therefore slow. Unfortunately, it had its own set of bugs.<9>

MICROPRODUCTS ASSEMBLER: The first commercially available assembler for the Apple II, this was a "four character assembler", meaning that labels (a designation identifying a line or variable) could only be four characters long. Later it was expanded to work with six character labels. Despite some annoying bugs, it was inexpensive at only $39.95.<10>

SC-ASSEMBLER II: Probably the second Apple II assembler that was commercially distributed. Externally it was similar to the Microproducts assembler, but was better supported and regularly upgraded. It was very compact, and achieved that goal by making heavy use of SWEET-16 code. Consequently, it was slow when assembling. The author, Bob Sander-Cederlof, later started a popular newsletter called "Apple Assembly Lines" to both support his product and to be an information center for 6502 assembly language tips and techniques.<10>

BIG MAC/MERLIN: Sold originally by A.P.P.L.E. as "Big Mac", and later under the name "Merlin" by Southwestern Data Systems (later known as Roger Wagner Publishing). This assembler has been well supported over the years and has been extensively upgraded. It is one of the few remaining assemblers that have moved on to the 65816 GS/OS world, while retaining full compatibility with the previous 8-bit 6502 versions. Currently it is sold as Merlin 816 (including an 8-bit version) and Merlin 16+. The author, Glen Bredon, has also done many other programs and utilities for the Apple II.

ORCA/M: Sold by the ByteWorks, the current version was chosen by Apple Computer as the official assembler of

the APW (Apple Programmer's Workshop) programming environment on the IIGS. ByteWorks has since expanded its product line to include versions of Pascal, C, BASIC, and other IIGS languages.

APPLE EDASM: This was Apple's original "official" assembler for the II Plus and later 8-bit Apple II's. Though no longer actively supported (ORCA/M having supplanted it in the APW environment), the early versions for DOS 3.3 were included on the Apple Toolkit disk, which also had a hi-res character generator that could be interfaced into Applesoft programs. The early ProDOS versions of EDASM were sold with a 65c02 debugger called BUGBYTER.

UCSD PASCAL ASSEMBLER: Part of the Apple Pascal package, it was popular because it had macro capability, could do conditional assembly and create relocatable code, and had a good text editor. However, programs created with it could not be run on a standard (non-Language card) Apple, because there was no utility available early on to transfer the files to DOS 3.2. (Later, A.P.P.L.E. published transfer utilities called "HUFFIN" and "PUFFIN" for movement to and from DOS 3.3, named affectionately after Apple's "MUFFIN" utility for DOS 3.2 to 3.3 file transfers).

MISCELLANEOUS OTHER ASSEMBLERS: ASM/65, sold by Programma; "EAT" (Edit and Assemble Text) sold by Software Concepts, and written in Integer BASIC; and L.I.S.A., sold by Laser Systems.<10>

MACROS VS. SCRIPTS

With the increase in complexity of applications programs has also come a secondary level of programming. This extension has been called a "macro", meaning that a single step would accomplish several separate ones that would ordinarily take more effort. Early examples of this were available in some DOS 3.3 utilities, where pressing Ctrl-C from the keyboard (for example) might cause the word "CATALOG" to appear on the command line. In this example, a macro was used to save keystrokes and speed up repetitive activities. Similar macros were available for BASIC programmers, making a control key sequence print many of the common BASIC keywords, speeding program entry. (This type of macro was different from macros used in some assemblers, such as Big Mac/Merlin and the Pascal assembler. Here a "macro" was a new command that was defined to represent several standard assembly operation codes. This did not shorten the final resulting program, but made it possible to more easily enter repeated sequences of assembly codes).

Application programs began to take this concept and include a macro capability (either offered with the program or as a third-party add-on product). With time, some of these macro features have become so complex that they have become programming languages in their own right. In fact, many of them are being referred to as "scripting" languages,

since they "direct" the function of a program, as a director uses a script to film a movie. This has been most popular with telecommunications programs, where the process of logging on to a remote computer, downloading new messages, and uploading replies is automated with a script that analyzes the responses from the other computer and takes the appropriate action. It has also been popular in programs like Applewriter (WPL, Word Processing Language) and AppleWorks (UltraMacros), where each has had its own method of automating repetitive tasks.

A LEAP IN COMPLEXITY

The environment for writing, compiling, and debugging programs has evolved along with the applications created by those programs. Originally, the Apple II and other computers of the day were used in a "command-line interface" environment. This means that each command was typed one at a time, and sometimes "batched" together to simplify a repetitive process (as with EXEC files under Apple DOS). An example of this command-line interface can be found by starting up Applesoft (or by using MS-DOS on an IBM). Anything that is to be done with this language has to be started by typing the proper command from the keyboard. Misspell the word "LOAD", and an error message is printed and it will stubbornly refuse to do what you wanted. The same command line is used for entering the lines of a BASIC program, or RUNning the program. This method was used because it was what programmers of the day were accustomed to. Nearly every computer prior to the microcomputer revolution worked in the same way, even if it was done using punched cards instead of being typed at a keyboard.

Minor differences were used from time to time in different computer languages, but none really took effect and changed the way in which people used computers until the release of the Macintosh in 1984. Macintosh used a radically different method of operating a computer. Instead of typing each command, the user would point to something on the screen and "click" on it using the mouse pointing device. And Macintosh programmers extended this concept to every application released with it. This different environment has been called a "graphic user interface" (GUI), and uses the concept of objects rather than typed commands. To delete a file, you don't type "DELETE PROGRAM", but point to the picture (icon) representing the file and drag it onto a picture of a trash can. This "desktop" includes more complex commands chosen from menus that appear in boxes called "windows" that pull down like a window shade from command category names on a "menu bar".

As the command line disappeared, so did traditional methods of handling program data. Words were still typed into a document on a word processing program, but many of the features that set up margins, tabs, and page breaks were translated into graphic icons selected with the mouse.

Eventually this progressed into the world of the programmer. The text of computer program was entered much like any word processor text, and the command to compile it into an executable program was now selected from the menu bar at the top of the screen.

A step further along this path is the concept of "object-oriented programming" (OOP). In this method, the details of windows, menu bars, buttons, and other GUI standards are used to create other programs that use a consistent interface. Instead of having to laboriously define at the byte level how to create a window box, the computer already knows how to do this; the programmer just has to tell the computer how big it should be and where to place it on the screen. OOP programming allows smaller modules (called "objects") to be used to build a more complex final product. A language that works in an OOP environment is finally available on an Apple II, but before we get to it, a little more introduction is necessary.

HYPERTEXT

"Hypertext" is a term created by Computer Lib author Ted Nelson that refers to a method of allowing a user to move from one concept to another in a text by linking the two concepts together.<11> The first type of program that used "hypertext" was a simple text based one. Certain words in the text of a document being viewed were marked to indicate that other information about that word was available elsewhere. Moving a cursor to that word and pressing a key would jump to the additional facts. For example, in an article about the history of music, the word "sonata" might be highlighted. Selecting this word could jump to another article that discusses sonatas in greater detail. When finished, the user could jump back over this link to the place where he left off in the original article.

"Tutor-Tech" was the first comprehensive hypertext system available for the Apple II series. It worked on 8-bit Apple II's, and was designed primarily for use in a classroom setting. Entirely graphics-based, it defined certain parts of the screen as "buttons", and moving the pointer to that area could allow the program to move to a different screen or cause something else to happen. As with any graphic interface, icons that represented certain functions were used to designate commands (i.e., to exit the program, you point to a picture of door labelled "EXIT").

In 1986 a remarkable program became available on the Macintosh that was, for a time, included with each Mac sold. "HyperCard" was a comprehensive system that used the idea of hypertext, plus added a programming language that consisted of words and phrases as close to English as anything else previously available on a microcomputer. The HyperCard system took care of the details of how to draw boxes and buttons, and left it to the user to define where to put them and how to label them. And because of the language (which

Apple called "HyperTalk"), user actions could do more than just move to a different picture (called a "card" by the program). It was possible to design simple databases, games, and much more using this system. Because it called a single part of an application a "card", a collection of cards comprising an entire HyperCard application was called a "stack".

With the release of the IIGS, the power was finally available in the Apple II world to create a similar product. But it didn't come first from Apple Computer; instead, Roger Wagner Publishing introduced a product called "HyperStudio" in May of 1989. This program used the super hi-res graphics modes accessible on the IIGS to create its own type of stacks. Like HyperCard on the Macintosh, HyperStudio used buttons and objects on the screen to direct movement through a stack application. It also included a hardware card that made it possible to easily digitize sounds to use in stacks. Though more extensive than Tutor-Tech, it was not quite as flexible as HyperCard, since it lacked a true programming language.

In January 1991, Apple released HyperCard IIGS, a conversion of the Macintosh product. This finally made a fully programmable hypermedia environment possible on the IIGS. Later in the year Roger Wagner Publishing responded with an updated version of HyperStudio that also included a programming language similar to HyperText that afforded more control over that stacks that were created. Although neither of these products gives the user power over details of the computer system itself (as does "C" or assembly), it does make it possible for a beginner to create programs that have outstanding graphics and sound without having to know exactly how the hardware produces these effects. This, along with the flexibility possible with these products, has led Dennis Doms in an A2-Central feature article to suggest that HyperCard IIGS (and now also possibly HyperStudio) will become the "Applesoft" of the 1990's; that is, an Apple IIGS user with HyperCard IIGS can create programs as easily as the Applesoft programmer of 1980 could do, but with far more attractive results.<11>

++++++++++++++++++++++++++

NEXT INSTALLMENT: Software

++++++++++++++++++++++++++

NOTES

<1> Walls, Keith S. "The Fantastic New World Of Apple Pascal", PEEKing At Call-A.P.P.L.E., Vol 3, 1980, p. 237.

<2> Howerton, Christopher, and Purvis, Lee. "The

Apple IIGS Pascal Revue", Call-A.P.P.L.E., Apr 1988, pp. 12-17.

<3> Winston, Alan B. "The Multi Lingual Apple", PEEKing At Call-A.P.P.L.E., Vol 3, 1980, pp. 222-224.

<4> Vanderpool, Tom. GEnie, A2 Roundtable, Mar & Aug 1991, Category 2, Topic 16.

<5> Winston, Alan B. "The Multi-Lingual Apple: Languages", PEEKing At Call-A.P.P.L.E., Vol 2, 1979, pp. 183-190.

<6> Cap'n Software's version was written by John Draper, the legendary phone phreaker "Cap'n Crunch" who had worked at Apple in its early days. During his time at Apple he had designed one of the first peripheral cards for the Apple II: A telephone controlling device that also just happened to be capable of hacking into long distance telephone switching systems, and was therefore quite illegal.

<7> -----. Apple Computer, Inc, Apple IIc Memory Expansion Card Owner's Guide, Singapore, 1986, pp. 2-4.

<8> -----. (ads), Call-A.P.P.L.E. In Depth #1, 1981, p. 106.

<9> Hertzfeld, Andy. "A Consumer's Guide To Apple II Assemblers", PEEKing At Call-A.P.P.L.E., Vol 2, 1979, pp. 164-166.

<10> Hyde, Randall. "Assembler Maxi-Reviews", PEEKing At Call-A.P.P.L.E., Vol 3, 1980, pp. 240-246.

<11> Doms, Dennis. "An Applesoft for the 1990's", A2-Central, Mar 1991, p. 7.09-7.13.

APPLE II HISTORY

Compiled and written by Steven Weyhrich

(PART 18 -- SOFTWARE)
[v1.1 :: 15 Sep 92]

"WILL SOMEONE PLEASE TELL ME WHAT AN APPLE CAN DO?"

One of the most important features to a customer considering any computer is, "What can I do with it?" It might be an attractive-looking box, with incredible features and potential, but if all it can do is run demonstration programs, it won't be very useful. In the early years of the microcomputer era, most users had to either write their own software or use programs written by some other amateur. "Commercial" software written by "professionals" was unavailable, except possibly from the company that produced the computer. And unless the user knew assembly language and the internals of the computer intimately (which depended on the willingness of the manufacturer to divulge those secrets), the only application software available was likely to be written in BASIC. Anyone who has used the versions of BASIC available at that time are well aware of the quirks and limits placed on the programmer by that language and by the small memory sizes available (see discussion in Parts 16 and 17).

As we have already seen, the Apple II came with few intentional secrets; the primary limitation on information distributed with it was the time required for Apple to produce a printed manual. When the first manual finally did arrive, it included a commented source code listing for the entire Monitor and all its supporting routines. This openness had a lot to do with the early success of the Apple II. Other manufacturers, such as Atari (with their models 400 and 800, based on the same 6502 as the Apple II) and Texas Instruments (who made a 16-bit machine called the TI 99/4), kept everything very secret and thus tried to maintain some control over distribution of software. This may have been done to ensure that only high quality programs were released, but more likely they were concerned about controlling who received royalties on sales of the software. Unfortunately for them, it choked the development of amateur software authors (who may have later become professional authors).

As an example of this corporate secrecy, one early programmer named John Harris wanted to write games for the Atari, but could not get the company to release any information on how certain effects were achieved in their commercially released games. He was bright enough to eventually figure out the secrets himself, and became one of

the wealthy software "stars" of the late 1970's and early 1980's.<1> Computer producers of the time did not yet grasp the principal of the software/hardware loop: Available software stimulates sales of hardware (computers and peripherals), which further enlarges the software market, which sells more computers, and so on. The industry was too new to know how to do much more than make and sell new computers.

SOFTWARE ON THE APPLE II

In the Apple II part of the computer world, the first distribution of software came from home authors. These people were usually first-time computer buyers who were captivated by the excitement of owning their own computer, and then had to sit down to actually find something useful or fun to do with it. They often brought their first programming efforts to show off at the computer store where they had bought their machine. Since the store owners had very little software to offer to their potential customers, some of these authors ended up with the opportunity of having their programs duplicated and made available for sale. Ken and Roberta Williams started their company "On-Line Systems" (later Sierra On-Line) this way with a game called Mystery House, one of the first adventure games featuring hi-res graphics pictures.<2>

Other early software came from the first user groups. These usually developed out of the gatherings that inevitably took place at the computer stores, as mentioned above. Since the people who actually used these computers day in and day out at home had a better grasp of how they worked and what could be done to work around problems, the store owners often ended up referring their new customers to these groups for the detailed help they needed. Not only were there the older groups (like the Homebrew Computer Club), but many newer, more machine-specific groups developed. Names like A.P.P.L.E. (Apple PugetSound Program Library Exchange) and International Apple Core became known well beyond their local beginnings as they began to distribute their newsletters and magazines to a national audience. Later, they became major sources of informational articles, utilities, and application programs that were as yet unavailable anywhere else.

Many of the programs sold by A.P.P.L.E. were popular with Apple II owners. A.P.P.L.E. was designed as a club with dues to pay for the collection of programs, all considered to be public domain, but sold to members at a nominal price to cover the costs of duplication. A.P.P.L.E.'s programs were written by amateur home users who had a unique idea, were able to make it work, and found that they had a product that was useful to others as well. Originally collected on cassettes, and later on disks, some of the programs were eventually made available as commercial products by authors that knew they had something unique that

would be in demand by Apple owners hungry for something to use on their computer. A.P.P.L.E. sold many of these as GamePaks, which contained several games on the same tape.<3>

Understanding that a large variety of available programs would help encourage more sales for the Apple II, Apple took some steps to help software authors get their programs on the market. In 1980 Apple employee Mike Kane suggested that Apple help distribute programs that were good, but whose authors couldn't get a publisher to distribute them or didn't have access to computer stores that were willing to sell it for them. Kane formed a division within Apple, called it "Special Delivery Software", and promoted both third-party and Apple-sponsored programs under that label. Between 1979 and 1981 a number of different programs were sold through Special Delivery Software, sporting the Apple logo and displaying a standardized appearance (packages, manuals, etc.), all listed in a catalog that could be used by dealers for orders. Apple Writer was originally distributed in this fashion, as were other less well-known programs such as Tax Planner, Plan 80, Script II (for Pascal), and MBA (a spreadsheet). Apple also established the Apple Software Bank and used it for special programs through 1980. It was more clearly a set of Apple-sponsored programs than were those sold through Special Delivery Software, and some of them programs, such as Quick File and Apple Plot, achieved strong popularity and were moved more into the mainstream of sales for Apple.<4>,<5>

SOFTWARE EVOLUTION: THE COMMAND LINE INTERFACE

Some of the earliest programs available for the Apple II had a user interface that was quite similar to the ones available for use with time-sharing terminals on mainframe computers: A command was typed on a line, and the computer would execute that command and return with a prompt for the next command. This method was the necessary way of doing things, because video displays were expensive and not in common use. This was particularly true for those who used remote terminals, which usually consisted of a paper-based glorified typewriter connected by a phone line to a mainframe. This device was physically limited to allowing commands to be entered one line at a time. The concept of displaying things on the screen in any order desired, not necessarily going from top to bottom (as would be necessary if it was being typed on a piece of paper in an teletype) was difficult for many programmers of the time to grasp. Moreover, for design purposes, the software code built-in to a computer (like the Apple II) that handled a command line style of interface was much simpler (and shorter) than what would be needed for a more complex interface. With memory at a premium price, simple would have to do. Thus, the Apple II used the command line interface in both the Monitor and in Integer BASIC. These

could be used as building blocks to create more complicated software, once people figured out how to do it.

The command line interface, though simple to implement in a program, had the disadvantage of requiring the user to know (and correctly type) the names of the commands. For example, a word processing program might use the command "LOAD" to get a text file into memory, the command "EDIT" to begin to make changes to that file, and then the command "SAVE" to put a copy of the completed work back onto tape or disk. "SORT", with various pieces of modifying information called "parameters", might be the necessary command to arrange the information in a database file into the desired order. Other commands might be needed to search for a specific word, replace a word, and move lines around. In fact, early word processors were often quite similar to writing a program in BASIC: Each line had its own line number, and inserting new lines often meant having to renumber the lines to make a new line available between two existing ones. If extra text had to be added to a line in the process of editing, making it too long, the end of that line might have to be re-typed into the following line and deleted from the current one.

More sophisticated text editing programs eventually began to appear that took advantage of the fact that the user was not working with a typewriter and paper, but with a video screen. These "full-screen editors" would allow use of the arrow keys (or the IJKM "diamond" on the keyboard) to move the cursor around on the entire screen, and it made text entry and later editing easier. As they were further refined, these newer word processors even allowed what had previously been impossible: Text could be typed in the middle of a line, and the text to the right of the cursor would be magically pushed to the right (even "wrapping around" to the next line if needed) as things were typed. Deletions were just as easy. What was still cumbersome was the need to have specialized commands, often entered as combinations of the Control key and another letter, to carry out some of the functions of search and replace, copy, and so on. Moreover, these command keys were often different from one program to another, with Ctrl-F in one program being used to begin a "find" process, and in another program as a command to jump to the "first" line of the file. As the full-screen method of text editing became more standard, the command-line type of interface became less commonly used.

SOFTWARE EVOLUTION: MENUS

As mentioned above, one of the problems with the command-line method was the requirement for the user to have a good memory for the names of the various commands necessary for the program to function. If the command name was typed incorrectly, or if a specific parameter was omitted or given in the wrong order, an error message would

appear, causing great anxiety and hand-wringing to those who were still trying to overcome their fear of using a computer. As an alternative for certain functions in a program, the concept of "menus" became more popular (and was actually used as early as the Apple Color Demo program that came on cassette with the first Apple II's). A menu was simply a list of possible functions a program could carry out. It still often used a command style prompt ("Type choice") to allow entry of the desired item on the menu, but gave a little more ease-of-use since a specific command name did not have to be memorized. A further enhancement of this style of program construction was called a "magic menu", after a sample program written in BASIC and distributed by Apple. In this type of menu, the user had the option of typing the number of the desired menu entry at the prompt, or he could use the arrow keys to move a large inverse bar up and down the menu to that item. After selecting the item with the arrow key, it was executed by pressing the RETURN key. This came to be known as the "point and shoot" method of command selection.

AppleWorks (which will be discussed in detail later) took the "magic menu" interface to its highest form, adding the metaphor of "file cards". One menu appeared on the screen enclosed in a box, with a "tab" on the top left of that box. This box resembled a 3x5 file card. When a selection was made from the menu, another file card would appear on top of the previous one, slightly down and to the right, leaving the tab on the lower box still visible. This allowed stacking of menus, with a clear path identifying which menu led to the current menu. The ESC (escape) key was used to "back up" one level, erasing the menu card on top and re-drawing the menu card underneath it. Also, prompts were displayed on the top line of the screen that told where ESC would take you, and what function was currently being executed. Part of the success of AppleWorks stemmed from its ease of use in this respect. Not only were there no cryptic commands that had to be remembered and typed, but the use of special command keys was reserved for advanced use of the program. And when such special keys were needed, a standard "help" screen was available for quick reference. It was possible to do quite a bit in AppleWorks without the need of even opening the instruction manual.

SOFTWARE EVOLUTION: GRAPHIC USER INTERFACES

One thing necessary to make computers easier for people to use was to overcome both the fear problem and the frustration problem. Those who were inexperienced in the use of computers were often afraid that they would press a button that would cause something terrible to happen. If they overcame the fear problem, they still had to face the frustration of trying to decipher cryptic error messages ("*** TOO MANY PARENS" or "$27 Error"), or lack of success

in getting the computer program to do what they wanted it to do.

Adding familiar things to the screen, like the file card menus in AppleWorks, made the fear factor diminish. Making the keys that controlled certain features of that program work consistently from the word processor to the database to the spreadsheet decreased the frustration factor even further. But there were still barriers to overcome in making computers easier to use.

When Lisa appeared on the scene in 1983, and Macintosh in 1984, computer users were exposed to a radically new concept in computer software. These computers lacked the previous standard of typed command input to control programs. Instead, they used a bit-mapped graphics screen to represent a desktop, with pictures (called "icons") that represented a program to run or a file to load. It took the "point and shoot" interface to the limit; you used the mouse to move a pointer on the screen onto an icon representing that program, and then "click" on it to start the program! For more complex control, the Mac used a variation on the "magic menu" system: A "menu bar" at the top of the screen gave a list of command words, arranged horizontally on the same line. Pointing to one of the words and holding down the mouse button would cause a menu to "pull down" like a window shade, displaying several further options available. The desired choice on the menu could be highlighted by moving the mouse to that item (such as "Delete") and the command would be executed. This approach made use of the Lisa and Macintosh considerably easier for the novice computer user, although some commands were also given keyboard equivalents similar to the old "Ctrl" key commands, so a more experienced user could execute some of them without having to take his hands off the keyboard. If AppleWorks could be considered easy enough to use without opening the reference book, this graphic user interface (GUI) was even more so. It also provided a standard environment that all programs written for the Mac could use, making it easier to learn how to use a new program.

Although the 6502 processor did not have the horsepower of the 68000 in the Mac, some programs began to appear for the Apple II that tried to make use of the same concept of overlapping windows, pull-down menus, and a mouse (or joystick) driven pointer. Quark released a program selector called Catalyst that used a similar graphics-based desktop, icons for files, and the point-and-click method of file execution. It was included with some of the early UniDisk 3.5 drives, and on Quark's hard drives. Another company, VersionSoft (from France) had a program called MouseDesk, which was distributed in America by International Solutions. MouseDesk worked just a bit better than Catalyst, but did not do very well as a standalone product, especially with Catalyst being given away free with the new UniDisk. Eventually, International Solutions made MouseDesk available for only ten dollars via mail-order, hoping to get it into general enough use

that their other graphic- and mouse-based products would sell better. Although that did not happen, International Solutions did eventually sell the rights to distribution of MouseDesk over to Apple Computer. Apple then modified the program and included it with as a rudimentary desktop (modeled after the Macintosh Finder) for their first versions of ProDOS 16 System software for the Apple IIGS.

With the release of the IIGS, it became possible for better GUI software to be produced for the Apple II. The 65816 processor had a bit more power, and the IIGS provided a better quality graphics environment (via its super hi-res mode) and more available memory than was possible on the older 8-bit Apple II's.

SOFTWARE: APPLE'S GREATEST HITS

It is beyond the scope of this writing to go into much detail about the many programs released over the years, as the sheer volume of them since 1977 is enormous. Even a brief mention of them all could become a book in its own right, but Appendix A contains a listing (in moderate detail) of popular software released over the years. In this segment here I will address in a little more detail three programs that have been particularly influential in the Apple II world: VisiCalc, Apple Writer, and AppleWorks.

By 1980, the Apple II software market had fairly well established itself. This allowed users of the computer to no longer have to write their own programs, but instead move on to simply being able to use them. Softalk magazine, which began in that year, had started nearly from the beginning with an analysis of top selling software of the day. In their second issue (October 1980) their bestseller list first appeared, with the top thirty software programs ranked based on actual sales information obtained by polling retailers across the country. In that first list the top selling program was VisiCalc.

SOFTWARE: VISICALC

A major part of the answer to the question, "What can I do with this computer?" lies in whether or not the software program in question is so important or useful that it literally sells the computer. Robert X. Cringely, in his book "Accidental Empires", put it this way: "VisiCalc was a compelling application -- an application so important that it, alone justified the computer purchase. Such an application was the last element required to turn the microcomputer from a hobbyist's toy into a business machine. No matter how powerful and brilliantly designed, no computer can be successful without a compelling application. To the people who bought them, mainframes were really inventory machines or accounting machines, and minicomputers were

office automation machines. The Apple II was a VisiCalc machine."<6>

Visicalc was a way of using a computer that no one had ever thought of before, especially at the time when most computers were mainframes with limited access to the "average" user. VisiCalc was written by Dan Bricklin, a programmer that had decided to enter Harvard Business School in the fall of 1977 and learn a second profession. Because of his programming background, he saw ways in which some of his class work could be simplified through the use of computers. He wrote programs in BASIC on the college time-sharing system to do his financial calculations, but found it tedious to have to re-write the program to deal with each new type of problem.

In a class that dealt with business production, Bricklin learned that some companies used long blackboards (sometimes stretching across several rooms) that were divided into a matrix of rows and columns. Each row and column had a specific definition, and calculations were made based on the contents of each cell (the intersection of a row and a column). If the value of one cell changed, the values of any cell that made use of the first cell's value also had to be changed. Because this was all written on a blackboard, the results had to be checked and re-checked to make sure that something hadn't been missed when changes were made during a planning session. Bricklin conceived of a computerized approach to this production and planning matrix. Even though the computer could not display the entire matrix at once, the video screen could be used as a window on a part of the matrix, and this window could be moved at will to view any part of it. Best of all, the computer could keep track of all the calculations between the various cells, making sure that a change made in one place would be properly reflected in the result of a calculation in another place.

Over a single weekend he wrote a program in BASIC that demonstrated this concept. This demo program was rather slow and could only display a single screen of cells, but it was enough to illustrate the concept. Bricklin teamed up with a friend from MIT, Bob Frankston, and together they looked for a publisher for the program. They found Dan Fylstra, who had graduated from Harvard Business School a couple of years earlier and had started a small software company called Personal Software, which he ran out of his apartment. Fylstra's primary product at the time was a chess program for the Apple II, and he was preparing to release the first commercial version of the adventure game Zork. After he heard what Bricklin and Frankston had in mind, he agreed to help them out. Fylstra loaned an Apple II to them as a platform on which to develop a more full-featured (and faster) machine language version of Bricklin's program. During 1978 and 1979 they worked together, as time permitted, with Bricklin doing the program design and Frankston writing the code. (One design contribution made by Frankston was the idea of using

"lookup" tables, which he wanted so he could use the program to calculate his taxes). They did most of their development work on an Apple II emulator running on a minicomputer (much as Apple itself had used a local time-sharing computer for development of the original Apple II Monitor program). They named their program "VisiCalc", and by October 1979 it was ready for release.

At first, VisiCalc was not a big hit. When most customers at computer stores were shown what the program could do, they didn't really grasp the concept behind it well enough to appreciate its possibilities. When business customers who had some computer knowledge came in and saw the program, however, they immediately saw that it could simplify much of what they did. VisiCalc actually sold Apple II's to many customers, and these businessmen managed to sneak the new computers onto their desks (despite company policies that discouraged use of anything but the company's mainframe). The combination of the Apple II's ability to expand its memory up to 48K, and the new Disk II drive to use for quick and easy data storage and retrieval, made VisiCalc an ideal program to sell potential users on this new computer.

Although executives at Apple Computer had been shown a pre-release version of VisiCalc, they also did not really understand the potential of the program. Trip Hawkins, an Apple employee responsible for developing plans to help sell computers to small businesses, could see that this could become a major selling point for getting Apple II's into those businesses. He negotiated with Dan Fylstra about the possibility of Apple purchasing from Personal Software all rights to VisiCalc (thus locking up the market in Apple's favor). However, Apple's president, Mike Markkula, felt that the $1 million in Apple stock offered by Hawkins was too expensive and cancelled the deal. If his decision had been otherwise, the future of the microcomputer industry might have been quite different; however, Apple was headlong in their push to create their next product, the Apple III, and a million dollar investment in an untried program for this "aging" Apple II was not in their agenda at the time.

Bricklin and Frankston had themselves formed a company called Software Arts, and it was this company that had contracted with Fylstra's Personal Software. As part of their arrangement, they were obligated to create versions of VisiCalc for many other microcomputers, from the TRS-80 to the Commodore PET and eventually to the IBM PC. As sales of VisiCalc grew by leaps and bounds, Personal Software (and Software Arts) became quite wealthy. To more closely identify his company with his flagship product, Fylstra changed its name form Personal Software to VisiCorp. He also hired other programmers to write companion software to extend the usefulness of VisiCalc. These included VisiFile (a database system), VisiSchedule (capable of creating critical path PERT schedules), VisiCalc Business Forecasting Model (a set of business templates for VisiCalc), and VisiTrend/VisiPlot (graphs, trend

forecasting, and descriptive statistics).

But despite these additional products, VisiCalc continued to be VisiCorp's cash cow. This, ironically, led to the company's biggest problem, centering around a disagreement about money. VisiCorp's contract with Software Arts guaranteed Bricklin and Frankston a hefty 37.5 percent royalty on each copy of the program that VisiCorp sold. VisiCorp was responsible for marketing and distribution of the program, but it was Software Arts who owned the rights to it, and they had no motivation to change their contract to decrease the royalty percent to a number that was more typical for programmers.

The problem escalated when VisiCorp filed a lawsuit seeking damages because Software Arts was supposedly late in providing them upgrades to VisiCalc. Software Arts countersued, and demanded back the rights to distribute the product themselves. Further complicating matters was the fact that the name "VisiCalc" was a copyright of Software Arts, but a trademark of VisiCorp.<7>

By early 1985, things had worn on to the point where Bricklin decided to end the battle by selling the rights to VisiCalc -- but not to VisiCorp. Instead, Mitch Kapor, who ran the Lotus Development Corporation, purchased the program. Kapor had previously worked for VisiCorp, and had helped write VisiTrend/VisiPlot. After he sold the rights for those programs to VisiCorp, he began design on a spreadsheet program that would run specifically on the IBM PC, with the additional features of limited word processing and the ability to create graphs. His program, Lotus 1-2-3, worked as well on the IBM PC as the original VisiCalc had on the Apple II (the ports of VisiCalc to other machines had never been quite as good as the original), and Lotus eventually captured the spreadsheet market on the IBM. In fact, it became the "compelling application" that helped push that computer platform into prominence. It had, however, made a significant contribution to decreased sales of VisiCalc, and after Lotus succeeded in purchasing it from Software Arts, VisiCalc quietly disappeared from software store shelves.

SOFTWARE: APPLE WRITER

This was certainly not the first word processor for the Apple II, but it was one of the most popular. During the four years that Softalk magazine was in print, Apple Writer rarely (if ever) disappeared from their best selling software list. Even if it was not in the Top Thirty, it usually held some spot on their list of top Word Processors.

The original version was released in 1979. Apple Writer 1.0 had to deal with the limitations of the Apple II in the form of its uppercase-only keyboard and 40-column display. Clearly, a document produced on a computer could be uppercase only, but it was more valuable if it could look more like that produced on a typewriter. To achieve entry

of upper and lowercase characters, Apple Writer used inverse text to display uppercase, and normal text to display lowercase. When entering text, an uppercase letter was entered by pressing the ESC key once. This changed the usual cursor box to an inverse caret (^), and the next letter entered would be uppercase (displayed in inverse). If the ESC key were pressed twice in a row, the cursor changed into an inverse plus sign (+), and was now an editing cursor that could be moved through the text.<8> The IJKM diamond on the keyboard was used to move the cursor, just as it was used for moving the cursor for editing lines of BASIC programs. Although the box cursor used in Apple Writer looked just like the flashing box also used in Apple BASIC, this cursor "floated" through the text instead of sitting on top of a character. If you moved it through the word "AND", it would look like this as it went from left to right: *AND A*ND AN*D AND*.

This original version of Apple Writer actually consisted of two separate binary programs: TEDITOR and PRINTER. The first program was used to actually edit the text, and the second one would print the files created by the TEDITOR. In its first release, Apple Writer had two problems that bothered some early users of the program. One was that the files created by the program were Binary files (instead of Text files), apparently as a means to speed saving and loading files under Apple DOS. Although it worked fine for Apple Writer, the files could not be used by any other program. The other problem had to do with the way in which it used (or misused) the ASCII character set. The Apple II, you may recall, used the upper half ($80-$FF) of the ASCII set for its screen display of "normal" characters (much of the rest of the microcomputer world tended to use the lower half), and used the lower half ($00-$7F) for flashing and inverse characters. In the upper half, the characters from $80-$9F were designated as control characters (generated by pressing the "Ctrl" key with a letter key), $A0-$BF were special characters and numbers, $C0-$DF contained the uppercase alphabet and a few more special characters, and $E0-$FF repeated the characters from $A0-$BF (this is where the lowercase letters should have been, according to the ASCII standards). Since the lowercase ASCII characters were unavailable, the Apple II video routines translated any characters in the $E0-$FF range into characters in the $C0-$DF range, making them displayable on the uppercase-only screen. Apple Writer, for some reason, used the $C0-$DF range internally for display of uppercase letters (which was standard) and the $E0-$FF range for special characters and numbers (instead of using the $A0-$BF range). When some users began plugging different ROM characters chips (like the Paymar chip) into their Apple II Plus computer, they found that Apple Writer wouldn't display text properly. The number "3" appeared as a lowercase "s", and "%" as an "e". A special patch was soon developed to intercept Apple Writer's text output to the screen and make the correct translation to display

lowercase as lowercase, and numbers and special characters where they were supposed to be.<9>

Apple Writer 1.0 ran from 13-sector DOS 3.2 disks, and the binary files it produced had names that began with the prefix "TEXT." (a file named "LETTER" would appear on disk as "TEXT.LETTER"). Apple Writer 1.1 was released in 1980 when DOS 3.3 became available. It ran under the newer 16 sector format, and contained some minor bug fixes. This version also had available a companion spell checker called Goodspell.

The next version released was called Apple Writer][. This one came out in 1981, was copy-protected, and still ran on an Apple II Plus under DOS 3.3, but now produced standard Text files instead of the older Binary files, and could properly display 40-column lowercase characters when the character generator ROM was replaced. It also supported 80-column text if a Sup-R-Term card was plugged into slot 3. In 40-column mode, words would now "wrap" to the next line if they were too long to display on the current line (the older versions of Apple Writer appeared to split the word and continue it on the next line). The ESC key was still used as a pseudo shift key (one press) and to enter editing mode (two presses, displayed as an inverse "@" instead of the "+" in previous versions), but the keyboard SHIFT key could be used to enter uppercase characters if the "shift key mod" was performed (recall that this connected the shift key to the input for button 3 on the game paddles). Other new features included a glossary and the Word Processing Language (WPL). In modern terminology, WPL was a macro or scripting language, making it possible to automate nearly everything the program was capable of. A WPL program could create templates like form letters, or could be used for entry of repetitious text (such as your return name and address for correspondence).<8>

Apple Writer //e, also copy-protected, came next in 1983 with the Apple IIe. This took advantage of the features of the new IIe (such as the built-in 80 column display and full keyboard). It also included improvements in tabbing (since a TAB key was now available on the keyboard), could create larger text files (these could be larger than the size of memory, by loading just a segment of the file into memory at one time), could "print" text files to the disk, could directly connect the keyboard to the printer (to use like a typewriter), and had improvements in the WPL language. When the Apple IIc came out, users of this version of Apple Writer had some problems, as the inverse status line at the top of the screen displayed uppercase characters as MouseText; however, patches quickly appeared to remedy this situation.<10>

The first version to run under the ProDOS operating system was called Apple Writer 2.0. It came out in September 1984, was not copy-protected, and it fixed the MouseText problem. It also allowed the user to set right and left screen margins, giving a closer approximation of

the final appearance of the printed text. This version also had the capability of connecting the keyboard directly to the printer or to a modem, allowing it to be used as a rudimentary terminal program. This version had some problems with properly printing to certain third-party parallel printer cards (such as the Grappler).<11>

One annoying "feature" that was added to this version (and was also present in a couple of other Apple-distributed programs, AppleWorks 1.3 and Instant Pascal) was that it did not follow Apple's published protocols in properly handling slot 3 RAMdisks (or other disks). Since some programs used all 128K memory that could be present in a IIe or IIc, Apple had given guidelines in one of their Technotes on how to properly "disconnect" the 64K RAMdisk (which was designated as slot 3, drive 2) so all 128K would be available to the program. Apple Writer and the other two programs mentioned above had been written so that they disconnected any slot 3 disk device, whether a RAMdisk, hard disk, or a genuine Apple disk. It is not clear as to why this had been done, although it was suspected in publications at the time that someone at Apple had done this so memory cards not made by Apple would fail to work. Some of these memory cards had been made to also work in slot 3 but to not interfere with the official 128K of program memory. Their manufacturers had worked to follow Apple's published standards, and then had been bypassed by what appeared to be programming arrogance. Patches to make these programs work properly appeared when the problem was identified.<12>

Apple Writer 2.1 appeared in late 1985. It contained some minor bug fixes, including the above-mentioned problem with some parallel printer cards. The 2.0 version had printed characters as low-ASCII (values $00-$7F), which caused a problem with some kinds of interface cards and printers. Version 2.1 changed this so characters were printed as high-ASCII ($80-$FF), although files printed to a disk file were saved in the original low-ASCII format.<13> This version also was not copy-protected, making it possible to easily install on a 3.5 disk or hard disk.

When AppleWorks appeared on the scene, Apple Writer began to decrease in popularity; however, old time users did not like AppleWorks as well as Apple Writer, primarily because it put a layer of "protection" between the user and the program. This made it easier for the computer novice to immediately put the program to use, and less likely to do something that would "mess up" his printer or interface card internal settings. That same protection also made it harder to do specialized jobs. For example, where Apple Writer would allow entry of control characters (which allowed very specific control of printers and their interface cards), AppleWorks was much more restrictive in this sense, handling more of the details of printer control internally. Apple Writer's power made it possible to even create documents on Postscript laser printers (as

demonstrated by Don Lancaster in his Computer Shopper column, "Ask The Guru"), something that all the computer experts claimed was not possible on an Apple II. Where Apple Writer allowed an experienced user to use all features on a printer and interface card to the maximum, AppleWorks was more dependent on the printer and card already knowing how to be cooperative with it. The same thing that gave Apple Writer its power also made it harder to user for less skilled users, who probably found intimidating its nearly-blank screen with no prompts or instructions visible.

For several years, from around 1988 through 1992, Apple Writer was not very available except as a used program. The exact reason for this is not clear. One reason probably had to do with the better-selling AppleWorks, which had the additional features of a spreadsheet and database. But with its Word Processing Language, Apple Writer was still more suitable for certain jobs than was AppleWorks; and yet, Apple simply stopped upgrading, distributing, and supporting it. But in the summer of 1992, one of the Sysops on GEnie's Apple (A2) Roundtable, Tim Tobin, was successful in contacting Paul Lutus. Tobin was coordinating a project that A2 had started to try to locate and revive the availability of "Lost Classics", programs that had ceased publication (often because their distributor had gone out of business), and recovering Apple Writer was high on his list. Lutus agreed to make his program available on a "freeware" basis: It could be copied freely and given away, but could not be sold for a profit. (This arrangement was quite similar to an earlier program Lutus had written, FreeWriter. He had released this program as freeware in 1984. FreeWriter was very much like Apple Writer, except it did not have a built-in ability to print the documents it created, and it did not have WPL). This new, free distribution was possible because although Apple Computer held the copyright on the Apple Writer documentation, Lutus had retained the copyright on the program itself (Apple had held the copyright on versions 1.0 and 1.1 of the program). Although the program is based on older technology, and does not take advantage of the larger memory sizes frequently available in the Apple II's of today, it still is powerful and is a welcome addition to any software library.

++++++++++++++++++++++++++++

NEXT INSTALLMENT: AppleWorks

++++++++++++++++++++++++++++

NOTES

<1> Levy, Steven. Dell Publishing Co., Inc, Hackers: Heroes Of The Computer Revolution, New York, 1984, pp. 314-319.

<2> Levy, Steven. Dell Publishing Co., Inc, Hackers: Heroes Of The Computer Revolution, New York, 1984, pp. 298-300.

<3> -----. "A.P.P.L.E. Co-op Celebrates A Decade of Service", Call-A.P.P.L.E., Feb 1988, pp. 12-27.

<4> Espinosa, Chris. (personal telephone call), Feb 4, 1992.

<5> Pohlman, Taylor. (personal telephone call), Feb 14, 1992.

<6> Cringely, Robert X.. Addison-Wesley, Accidental Empires, Reading, Massachusetts, 1992, p. 64.

<7> Tommervik, Al. "The Double Hi-Res VisiSuit", Softalk, Apr 1984, pp. 28-29.

<8> Dubnoff, Jerry. (personal mail), GEnie, E-mail, Aug 1992.

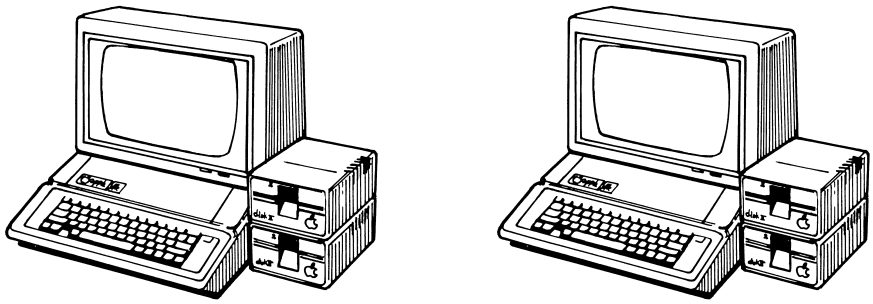
<9> Widnall, Sheila. "Lower Case For Apple Writer Using The Paymar Chip", PEEKing At Call-A.P.P.L.E., Vol 3, 1980, pp. 264-266.

<10> Lancaster, Don. Howard W. Sams & Co, Apple Writer Cookbook, 1986, pp. 29-30.

<11> Lancaster, Don. pp. 102-103, 111-112.

<12> Weishaar, Tom. "Ask Uncle DOS", Open-Apple, May 1987, p. 3.30.

<13> Weishaar, Tom. "Does Your Mother Love You?", Open-Apple, Jan 86, p. 1.97.



T H E E N D