CONDUCTED BY GREGG WILLIAMS AND ROB MOORE

# THE APPLE STORY

## PART 2: MORE HISTORY AND THE APPLE III

## *An interview with Steve Wozniak*

*Last month, Steve talked about his background, the evolution of the Apple I and II, and the early days of the company. In this part, the conversation switches to various aspects of the Apple II design, later personal history, and Steve's thoughts about the personal computing industry.*

### SWEET-16

BYTE: *One of the more interesting things in the Apple II ROM was your 16-bit pseudomachine called "Sweet-16." How did you come up with that?*

WOZNIAK: While I was writing my BASIC, I had been thinking about ways to save code. There were several places where I had to handle 16-bit pointers with an 8-bit processor, and that was pretty awkward.

So I decided to write a little emulator and implement a 16-bit machine that could interpret pseudocodes and implement registers 0 to 15 in the 6502 base page. It ran about 30 times as slow as 6502 assembly language, but it saved tons of code every time I used it in a program.

BYTE: *Did you actually use it in your Integer BASIC?*

WOZNIAK: No, I never had the time to reimplement the BASIC to use it. But I did use it in later years to write things like BASIC renumbering routines totally in Sweet-16. It was easy to mix Sweet-16 code with assembly language.

BYTE: *Isn't Sweet-16 still used in Apple DOS and ProDOS editor/assemblers?*

WOZNIAK: Yes, it's used in EDASM |the Apple Tool Kit 6502 editor/assembler|, mostly in the editor portion. Randy Wigginton wrote EDASM. He's worked here since before we even had a company. Lately he's written the Macintosh word processor—MacWrite. He's done a lot for the company, and he's used Sweet-16 in several things he's done.

### THE DISK DRIVE

BYTE: *Can you tell us a little about how you came up with the Apple II disk drive and how you ended up picking your form of group-coded recording?*

WOZNIAK: The disk design was my most incredible experience at Apple and the finest job I did. I never really knew what a disk controller was or what it had to do. But at Hewlett-Packard I had looked through a Shugart manual to see what signals were used and what they did. There were signals to make the head step in and out and signals to cause magnetic flux changes. It was similar to audio recording, and I knew about that. It was like a signal on a tape where you write it and then you read it back. So I figured out a simple little circuit to write signals at changing rates and read them back. I didn't know how disk controllers worked, so I assumed that I was doing something totally different. Maybe it wasn't as efficient, but at least I could write some data and read it back.

Well, Mike Markkula was annoyed because the cassette tape was too slow. He had a favorite checkbook program, and it took two minutes to read in the program and another two

(continued)

*Gregg Williams is a senior technical editor at BYTE. Rob Moore is a hardware designer and frequent contributor to BYTE. They can be contacted at POB 372, Hancock, NH 03449.*

PHOTOGRAPH COURTESY OF APPLE COMPUTER INC.

*'Hobbyists are a tiny part of our market, but they're faithful to the company.'*

minutes to read in his check files. He was complaining about this at a staff meeting, and I mentioned that I had this clever little five-chip circuit that could read and write a floppy disk. At the time, all the existing floppy-disk controllers were 40 or 50 chips, so I knew there must be something important that I wasn't doing.

I went off and tried to figure out what it was that I wasn't doing. One of our technicians had a North Star system, so I looked through their manuals. I read their schematics and figured out what every chip did. And I looked through their listings until I understood exactly what they were doing.

I was doing a lot more. I didn't even have to look at the sector holes, so I could use any disk drive, any floppy disk in the world. It was then that I new I had a really clever design.

The next week was Christmas vacation. Randy Wigginton and I spent the entire week, including the holidays, trying to get this disk reading and writing with a very simple operating system. We did the bottom levels of an operating system in that week. You could type R (for "read") followed by a program name like STARTREK, and it would load STARTREK into memory.

We were highly motivated because, at the end of the week, a show called the CES |Consumer Electronics Show| was starting in Las Vegas, and we wanted to go.

We worked all night the day before we had to show it |the disk drive| at CES. At about six in the morning it was ready to demonstrate. Randy thought we ought to back it up, so we copied the disk, track by track. When we were all done, he looked down at them in his hands and said, "Oh, no! I wrote on the wrong one!" We managed to recover it and actually demonstrated it at CES.

BYTE: *Had you been exposed to group-code recording before, or did you invent yours independently?*

WOZNIAK: The first version of the floppy-disk routines did not have group coding. I had followed the Shugart manual, which showed that you had alternate clock bits and data bits, so every other bit was wasted. I couldn't understand why it was necessary, but I started that way.

Then I came up with this idea to use coded recording. I knew the technical rule was that you could only have one or two zeros in a row. You could have either 4 or 8 microseconds between flux transitions. I didn't really know what group-coded recording was; I just knew that I could fit 13 sectors on a disk instead of 10. I had to write a program to take bytes off the disk, convert them to 5-bit chunks, and reassemble them into 8-bit data bytes.

It was a difficult routine to write. It was about a 20-hour job, and I'd work through the day for 10 or 12 hours and I wouldn't quite get there. The next day I'd come back and find out that I was starting exactly where I had the day before. This went on for almost a month. I was not quite getting the routines, and we were getting within a month of shipping the disk drives. Finally I stayed up all night until I got all five routines that had to work together done. So we were able to ship it the first time with the group-coded recording in place. Later, we changed the encoding method and stepped up to 16 sectors. That was DOS 3.3.

BYTE: *The Macintosh uses a custom chip called the IWM—Integrated Woz Machine—that does the same sort of recording. Can you tell us anything about that?*

WOZNIAK: My design was basically a little sequencer, or state machine. It used a PROM and a latch and cycled through various states depending on the input data coming off the floppy disk. The IWM takes that design and adds other features like the ability to go twice as fast—it can also do IBM format, double-density recordings.

BYTE: *It sounds like a fascinating part. Do you think we'll see Apple II owners benefit from it in any way?*

WOZNIAK: Well, it's our standard disk controller now, and it's cheaper than the older design. It's used in the Apple IIc.

BYTE: *Could you use it to get higher-density recording on an Apple II 5¼-inch disk?*

WOZNIAK: No, because the disk drives themselves aren't certified for double-density recording. You need heads with the proper gap, and they're more expensive.

BYTE: *You're a former hobbyist. Could a hobbyist buy one of these chips and a spec sheet and start playing with it?*

WOZNIAK: I don't think Apple would give out the spec sheet. I totally disagree with that policy because I'm very respectful to the hobbyists. They're a tiny part of our market, but they're loyal supporters and faithful people to the company. If they had a spec sheet, they could start playing with it and figure out a lot more incredible things that we never planned it to be used for—even using it as a communications channel from Apple to Apple, Macintosh to Macintosh. There are a lot of great tricks you could do with that little part. It's a beautiful random I/O device that has too many things that have not been taken anywhere.

## Personal Details

BYTE: *In 1981 you were in a plane crash and you left Apple for a while shortly after that. How long did it take you to recover from the accident?*

WOZNIAK: That was in February 1981. For about five weeks I had a type of amnesia that prevents you from forming any new long-term memories. After I recovered, people would show me pictures of myself in the hospital, playing games with my computer with my face all battered up. They would tell me stories of how I tried to sneak out of the hospital to visit my wife, Candy, or how I went to parties and rode my motorcycle. I didn't remem-

(continued)

ber any of that. I had all of my old memories, but I'd forget new things from one day to the next. Finally I came out of it one night, but I never got those memories back.

BYTE: *Why did you leave Apple?*

WOZNIAK: We had a hundred engineers at that point, and I was no longer really important to the company. I didn't want to be a manager; I was just an engineer, and I wasn't really needed there. But I didn't feel comfortable going to Steve Jobs or Mike Markkula and saying I wanted to take off. The plane crash was a good excuse. After five weeks of amnesia, I simply didn't go back. I decided that if I was going to take a year off I might as well finish college. It was the hardest year of my life.

BYTE: *We've heard that you went to UC–Berkeley and had some run-ins with your instructors. Could you tell us about that?*

WOZNIAK: I was going under an assumed name—Rocky Clark—so they didn't know who I was. I took computer science courses, economics, statistics, and a few other courses.

My computer science courses were interesting, but I have to criticize them a little because they taught only specific problems with specific solutions. You spent your time memorizing standard problems and solutions and then tried to recognize variations of them in the tests. You weren't supposed to explore new avenues or try things that nobody else was doing. You were only supposed to learn the proper answer. They thought that you could be trained to know all the problems and the standard solutions. Once you learned them all, you could solve them. It was wrong because they weren't really teaching you to solve problems—they taught you to identify them.

My economics course was interesting also. We had a socialist TA [teaching assistant] who taught us that companies made money by cheating the consumer. All the kids in the class thought that companies would make a lot of profit if they could figure out a way to cut the costs of a product down, to make it cheap and screw the consumer.

I contrast that with the way we did things at Apple. Every product design decision was based on what consumers wanted, what would compete the best, what they would buy. We tried to do what customers wanted, in our best judgment, and give them high-quality products.

So I would stand up in class and argue about what the TA was saying. After a while he started telling me to shut up, or that he would kick me out if I interrupted him again. Apple was the greatest business success in history, but I couldn't tell him who I was.

BYTE: *So you came back to Apple after about a year. What would you say is the most important thing that you've worked on since you came back?*

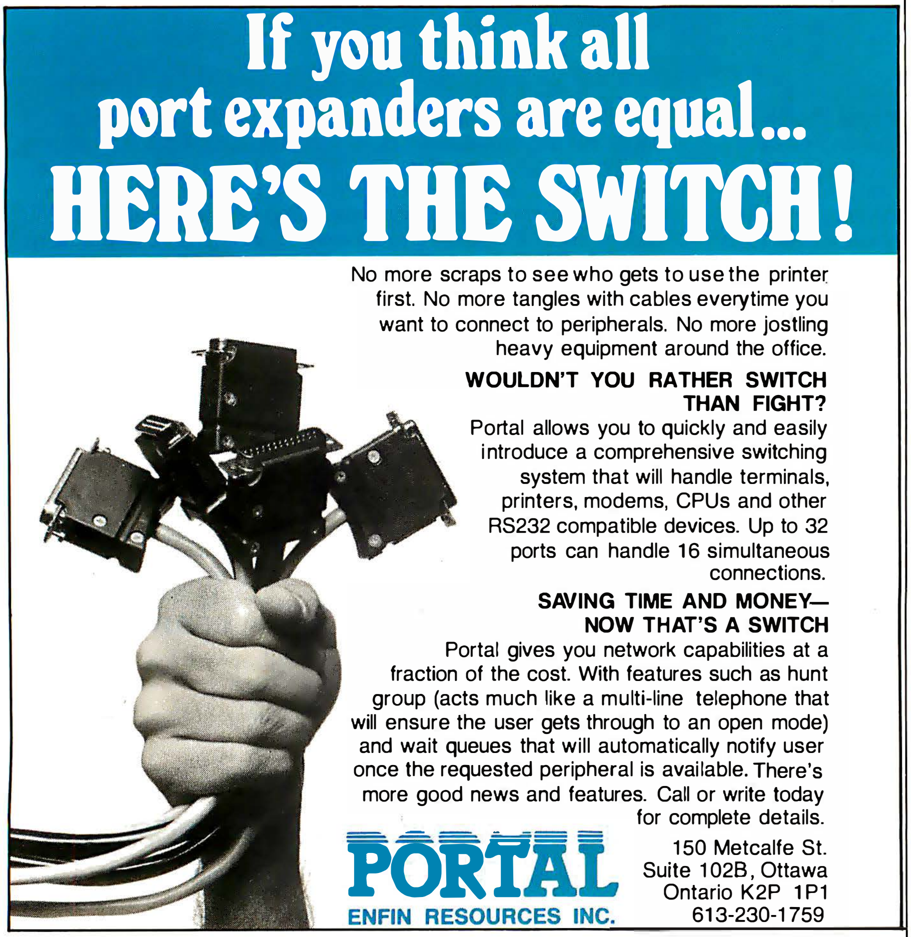
WOZNIAK: There isn't too much. When I came back I started getting a little bit involved with this division's management, but it was unofficial. Officially I took the title of Engineer. Mostly, I've stayed involved with the Apple II because that's where I've got the most to offer.

Because I'm a founder at Apple, I could take almost any role I want, but I've tried to avoid the newest, most far-reaching projects because there are other capable people to do them. I try to stick to small projects where I can sit down and handle them myself.

## Current Work

BYTE: *Do you have anything interesting going on now?*

WOZNIAK: There are not many individual projects in this whole business. The floppy disk may have been the last one for Apple. I have got a few projects that I'm working on now, but they're not all going at once.



The language Fifth is one of them, but I haven't written it yet. I'd like to combine the best features of BASIC, FORTH, and Pascal, and leave out the worst ones. You could have the formal structures of Pascal, the immediateness of BASIC, and the extensibility of FORTH. In FORTH or Logo, new words become part of the language, and you can use them immediately. It's really helpful for debugging. In Macintosh Pascal, you can define a procedure and run it immediately. I also want some level of globalness, like BASIC. I don't want to always have to declare a variable before I can use it. I like variables with scope, but I'd like undeclared variables to be totally global.

Another project is an operating system, like the one on the Macintosh, only a bit different and a lot more relational.

I also have a hardware project that I'd like to do, a personal computer based around consumer video sources like TVs, videotapes, and videodiscs. It would switch them around, synchronize them, and mix them, sort of like a little home editing studio. I think it's possible because memories are getting so cheap. You could hold a frame from each of your video sources in memory and let the software accommodate the sync variations. There are a lot of new chips available that do NTSC modulation and demodulation, so there ought to be a minimal chip solution.

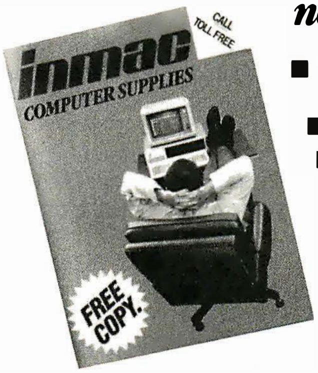
My main interest is still the Apple II—the home computer we started with. I'd like to see Apple do more with speech. There are some really inexpensive speech chips now, and that's the way the rest of the personal computer world is heading. I think we've been deficient in that area.

### SPECULATIONS

BYTE: *Are you thinking about using the new 65816 processor for anything?*

WOZNIAK: We're thinking about it and doing some R&D with it, but I don't know if we'll use it. Anything we do has to be compatible with the Apple II. If we found out that the 65816

*(continued)*



# THE SPREADSHEET

*In late 1982, A.P.P.L.E., a national Apple users group, offered an astounding value to its members. For $22.50, A.P.P.L.E. members could purchase a fully functional VisiCalc-like spreadsheet program called "THE Spreadsheet." At the time, various versions of VisiCalc and other popular spreadsheet programs were selling for between $200 and $300. THE Spreadsheet offered most of the features found in VisiCalc and added a few of its own. What's more, it was coauthored by the now legendary Steve Wozniak. Mysteriously, THE Spreadsheet was available for only one month, after which it was permanently discontinued. Since that time, original copies of THE Spreadsheet have become one of the few legitimate microcomputer collector's items. In the following segment, Steve describes how THE Spreadsheet came about.*

WOZNIAK: It started because Mike Scott was having difficulty negotiating with Personal Software [now VisiCorp] about VisiCalc. They just wouldn't do what he felt was right as far as giving Apple breaks on price or doing new enhancements to it. So he came by one day and he said to me, "Would you like to do a VisiCalc?" It wasn't even known as a spreadsheet then.

I was really scared. VisiCalc was the only spreadsheet for a long while, and none of us had really been in business before. Was it legitimate to go out and do your own spreadsheet? Or was it equivalent to just copying and ripping somebody off? I didn't want to get close to this whole thing because I didn't want it to look like I was copying somebody else.

Mike got Randy Wigginton and Randy said, "Sure, I'll do it." I said I'd do the arithmetic routines because they were general—something that could be used by any program—and I had some good algorithms that I had picked up at Hewlett-Packard that I wanted to implement.

So we started working on it and, a little ways in, Randy finished it. He wrote 4K of code in a couple of weeks, and I was shocked that he came in so quick; I hadn't even started mine. So, I was getting to work on my arithmetic routines and my first demo was almost ready but still a few days off.

At the time, everyone was trying to get the Apple III into production. Mike Scott was camped out in the Apple III building, forcing the things that had to get done to get it finished and delivered on schedule.

He was in a bad mood. Everywhere anyone ran into him in the company, he had a sour face—no laughs, no jokes. And because Randy finished his first part of the spreadsheet and I hadn't finished my first part, he'd always come to me. The way he was talking to me, I was afraid I was going to get fired. He was in a *bad* mood.

It was the scariest time of my life at Apple. I was really getting badly addressed just for not being on time with something. I had three more days to go, and I didn't dare run into him once more because I was already way overdue. So I said, "I've got to get him in a good mood for three days."

Well, he's a Star Wars fan. He gets the T-shirts the cast gets before they get them because he knows who prints them. So I had a friend of mine call his secretary saying, "This is George Lucas. Is Mike Scott in?" Of course he wouldn't leave a phone number, but he said, "I'll call back." It actually worked. Mike was in a good mood for a few days, and I got my routines done.

Anyway, we finished the spreadsheet, and Apple started looking it over. Originally it had been defined to be as accurate as VisiCalc, and we were so much faster than VisiCalc it was ridiculous. The exponential routines that I really wanted to write for so long, because I had a good algorithm, were 30 times faster than VisiCalc's.

All of a sudden we started hiring experts on precision and things like that, and they started redefining the project from what Mike Scott had originally defined. They added a lot of features that Randy had to implement. This made life very difficult here at Apple, so we finally decided, "No, it won't be a Special Delivery Software product [a line of software offered by Apple a few years ago]."

After a long, long time, Randy went to *Call-A.P.P.L.E.* [the magazine put out by A.P.P.L.E.]. One night we were out to dinner and he said to Steve, "Well, I'm going to sell that to *Call-A.P.P.L.E.*" or "I'm going to give this spreadsheet to *Call-A.P.P.L.E.*" Steve said, "Well, that's good" or something like that. It was not official or formal. Randy went ahead and hit on it and gave it to *Call-A.P.P.L.E.* It was sold for $22.50—really cheap. We didn't want any money out of it. It was just the old Homebrew Computer club spirit of "Give to help others."

It was a good product, and it had been delayed well over a year. If Apple didn't want it, they should have just gotten rid of it, and it would have been out a year sooner. As soon as it popped up from *Call-A.P.P.L.E.* and was advertised, Apple got in a real huff about it. They forced *Call-A.P.P.L.E.* to only sell it to members, and only one copy per member. So Apple basically put it out of business—only allowed it to be sold for one more month.

It's really funny, because how did Apple get started? I designed the Apple while I was at Hewlett-Packard. Hewlett-Packard has a policy of deciding about these things very quickly. They would check with their legal department and other divisions and decide very quickly. So they gave me a formal release on it quickly because they didn't want it. But here was Apple, which didn't want it but wasn't going to release it.

If you use much Apple II software, you'll be surprised when you take the spreadsheet that Randy and I wrote and boot it in. It boots in so fast, you can't believe what's going on.

BYTE: *Is that because there's no copy-protection on it?*

WOZNIAK: No. Believe it or not, I used those routines that I did for Apple II Pascal.

BYTE: *You, Randy Wigginton, and Guil Banks were listed as the authors. Who is Guil Banks?*

WOZNIAK: Guil is right here in this building. I did the boot stuff, and he had written a lot of the other fast disk routines in there. He did all the routines that managed the disk and read and wrote DOS files appropriately without using DOS.

This was basically an all-volunteer project that we did all on our own time. And it was really neat, so we decided that we'd give it to the rest of the world.

wasn't, it would be a serious question. It's too new a part right now.

BYTE: *How is its performance compared to the 68000?*

WOZNIAK: It should be available soon in an 8-MHz version that will beat the pants off a 68000 in most applications, and in graphics applications it comes pretty close. Some of the Macintosh people might disagree with me, but there are ways around most of the problems they see. An 8-MHz 65816 is about equivalent to a 16-MHz 68000 in speed, and a 16-MHz 68000 doesn't exist.

BYTE: *The Apple II family has been a great success, and many innovations have come along to extend its life. What things do you think were responsible for its success?*

WOZNIAK: For three years we were one of the biggest business successes in history. We had three years of fantastic business success, and lately we've had three years of sort of dismal business. We've grown, but any additional revenues have just replaced the stock dilution, and the price remains about the same.

During that three years there were two main factors that led to our success—our floppy disk and VisiCalc. Out of the original home computers, which included the TRS-80 and the Commodore PET, ours was the only one that had enough memory to run VisiCalc. VisiCalc and the floppy disk sent this company into the number-one position.

We were also very faithful to our users—we tried to support everybody. When we changed over to floppy disks, we still supported cassettes heavily. When we moved up to the Apple II+ with floating-point BASIC built in, we still supported the original Apple II.

Lately our strategies seem to be changing. When we come up with a new enhancement, we start moving away from the prior version much more quickly. This could be harmful to our good relationships with a lot of our faithful users.

The Apple IIe is a good example. Most new software that really uses the features of the Apple IIe won't run properly on an Apple II or II+. A lot of good software for the Apple II+ won't run on the Apple IIe. It's an incompatible world.

BYTE: *How do you think the Apple II family will be extended and improved in the future?*

WOZNIAK: There are obvious areas. We're always trying to come up with better combinations of features and still reduce the cost. We're looking into improved processors like the 65816 we discussed before. Video resolution is always improving. We're trying to increase speed and the amount of memory in the machine because it's critical to certain applications.

The IBM PC is very successful, and it had no competition from Apple for the last three years because we made sure that it didn't. We would not allow the Apple II to compete in that market for three years.

PCSD is the most difficult division in the company. [Steve is referring to the Personal Computer Systems Division, responsible for the Apple II and Apple III products.] We're hamstrung by the need to be compatible; Macintosh isn't.

So we will continue to make improvements and produce new machines, but they'll always be compatible. With the Apple IIe we went to 128K, 80-column display, and double-resolution graphics. We came out with the IIc portable.

## APPLE VERSUS IBM

BYTE: *What did you mean when you said that the Apple II was not allowed to compete in the IBM market?*

WOZNIAK: Apple has never really supported the Apple II in the business market. If you walk around a trade show and look at the software running on the IBM PC, you'll see that most of it is a step above what's possible on today's Apple II. They have more RAM that's easily addressed and better access to hard-disk drives. Programs like 1-2-3 cannot be easily implemented on a 128K Apple II, but IBM has a capable machine for that level of software. Our machine has to be able to address more memory and

(continued)

*'I have positive and negative feelings about things Apple has done.'*

handle larger disk drives before we can really start to compete with equivalent software because IBM beats us in capability today. We need better screen resolution, more memory, and better speed. A 16-bit processor would help, although "16-bit" really doesn't mean that much.

Whatever we do will be compatible, because we don't want to alienate our existing customer base. If we can come up with the right machine, then we can start to talk about some really good software and compete well with IBM. Even if we do, the new features won't really be used right away.

BYTE: *Have you said all that you meant to say about where Apple is vis-à-vis IBM and where you personally feel that Apple has made its mistakes?*

WOZNIAK: I have both positive and negative feelings about things Apple has done, but I'm always honest. There is one real mistake that Apple made, in my opinion, and this is very subjective. It is symbolic of what could happen to IBM with its PCjr.

We had become a huge business success in 1979. We had really made it with our floppy disks and VisiCalc, and it looked like we were going a long way. So we decided it was time to start putting together a real company, a big company. We needed to start staffing up and hire a lot more engineers. So we set the Apple III project in motion.

The executive staff felt it understood the Apple II market. After VisiCalc, it was perceived that 90 percent of all Apple IIs sold were going to small businesses. Only 10 percent were going into this home hobby market that we originally thought was going to grow to be billions. Originally we were a home hobby computer. Now, suddenly, small businesses were buying Apple IIs, and they wanted more features—an 80-column display, lowercase characters, maybe more graphics modes and colors, and more memory. These were all the things that one product, VisiCalc, led to.

According to any research we could dig up, many people were buying the Apple II for small business because it had a disk drive and it could run VisiCalc. These weren't the people I was closest to, so I kept my mouth shut because I was only one out of the staff of 15. So we started staffing

*(continued)*

up and building an organization and a management structure based around doing this new product—the Apple III.

## THE APPLE III

We had some problems getting things done on time with the Apple III because of (a) our lack of experience as a group working together, and (b) not being able to predict project lengths well enough.

We started hiring intermediate levels of managers. Sometimes the managers were getting hired at a rate that added a completion date to the project faster than it was getting completed.

Around this time we started developing a perception of market separations—good strong separations between products so they don't overlap. You don't want to design a product that competes heavily with your own existing product. I claim that's untrue. What you really don't want to do is design a product that doesn't offer any more than your existing product.

So we started setting up strong boundaries. The Apple III would be our 80-column business machine and have 90 percent of our market. The Apple II would be our 40-column home/school machine and have 10 percent of our market. The entire executive staff was sure that once the Apple III was out, the Apple II would stop selling in six months. I felt really down, because this 10 percent were my friends—the hobbyists and home users.

The Apple III hurt Apple in many ways, but it was a very well conceived product. And because we were so successful with the Apple II, we decided to build in an Apple II emulation mode to take advantage of all the software that was out there. The emulation mode did get built in, but, because of our concept of market separations, it was a very limited emulation. While our Apple II customers were adding 80-column cards and 16K RAM cards to their machines, we actually added chips to the Apple III to prevent access to many of its features during emulation mode. In emulation mode you only had access to 48K of memory—you couldn't use the 80-column display, the extra graphics modes, or the extra memory. The emulation mode wouldn't even run much of the existing Apple II business software, and there wasn't much Apple III software available.

Originally, we planned to deliver four applications with the Apple III—word processing, a spreadsheet, business graphics, and a database program. Steve Jobs's thinking at the time was, "People don't really want to buy a computer. They don't want to know about microprocessors or cards or buses. They want to buy VisiCalc—they want to buy a solution." So we were going to provide the four major solutions. But because we were having problems managing the Apple III project while we were building our management structure, we were only able to deliver our operating system—SOS—and VisiCalc, which was done by Personal Software [which later became VisiCorp].

The Apple III shipped very late and had 100 percent hardware failures. This is very subjective, and some people might disagree with me, but I think we were trying to be too pure. We wanted to do it on one PC board, not two, and it didn't fit on one PC board. So we got a company that could put three traces between two IC pins, had them do the PC board, and 100 percent of the Apple IIIs failed.

The Apple III is really very good, but we spent three solid years keeping the Apple II down, and now it's finally being allowed to grow in that direction. We've come out with ProDOS, which is a major improvement, and the ProFile [hard disk] is available for the Apple II now. It's a good start. I think they are going to find out that allowing the Apple II to get there will improve the whole Apple image.

The III will do very well in its established vertical markets forever, but it really won't make the huge success we thought it would. It may have the best chance it's ever had. ProDOS is the best way to get someone closer

*(continued)*

*'We did SOS 3 years ago, and the rest of the world hasn't caught up.'*

to SOS. We did SOS three years ago, and the rest of the world hasn't caught up or come close to it. Macintosh went in a different direction, so I can't compare the two systems.

BYTE: *Is SOS really that good?*
WOZNIAK: I think it's the finest operating system on any microcomputer ever. It's the greatest thing in the world, but I wish we gave out listings of it.

BYTE: *Wasn't it the first commercial system that actually had installable device drivers?*
WOZNIAK: Yes, the Apple II uses device drivers in ROM, on the I/O cards, but they won't work on the III because they depend on specific memory locations. On the III, you load a device driver into RAM for any device you add to a system, so it's infinitely flexible. You can always change it, correct bugs, and it's clean. But it's more difficult for outside manufacturers, because they have to supply a disk and instructions on how to update your system. It would have been the best of both worlds if you could plug in a card with a ROM on it and if the first byte in the ROM were a 12, for example, the system would recognize that the ROM holds a device driver and link to it automatically. That would have made it much easier for the card manufacturer. It would have been really easy to allow both techniques, but the Apple III engineering group didn't want to do anything the way it was done on the II. Marketing just didn't have enough leadership and control.

Anyway, having the Apple III as a failure didn't really hurt the company much. The Apple II was still very healthy, but for the next three years the Apple III hurt the company tremendously because everyone in Apple knows how great a machine the Apple III really is. It's a very clean machine, it's easy to use, and it's really been organized right with the operating system.

Unfortunately, we made it very difficult for anyone to get access to the insides of the machine. We had hired some very bright people who figured, "This is the right way it should be done. So we'll give out enough information to do this and we won't give them any more, because they might try to do something they're not supposed to do." The right way for one person is not the right way for another. We closed that machine up to where somebody could have a very difficult time finding out how to add their own I/O drivers. We did not make it easy for the outside world. We thought we wanted all of the markets for ourselves.

You have to let the end users develop their own standards. You've got to give them the freedom to discover how they're going to use an operating system, what sort of things they're going to buy. And if you're really right and have provided a good solution, that's where they're going to settle. The thinking on the III was very much like a religion in that it could only be done one way—our way. We made it very difficult for outside developers, instead of providing all the information as we did with the Apple II.

BYTE: *Has that attitude changed now?*
WOZNIAK: No. It's still the most negative thing in our whole company, and it will be for years.

I think that when a new market

(continued)

evolves, like personal computers did, there's a period of time when you've got to let the world go in all random directions, and eventually it will subside because it wants standardization. Then, once it's obvious what the standards are, they should be heavily supported by the manufacturer. You can't try to dictate a standard.

### THE APPLE II AND THE APPLE III

When we came out with the Apple III, the engineering staff canceled every Apple II engineering program that was ongoing, in expectation of the Apple III's success. Every single one was canceled. 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.

There is a lot to somebody's perception or image of a machine and how good it is. How many of my friends have them? How many people in the world have them? The Apple III was a failure the first year as a product—it had a bad image. When you give a bad first impression, you can go for five years trying to overcome it.

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 on a lower level than on the III. Only now are we discovering that good solutions can be implemented on the II.

The Apple II was kept out of the business market to keep the III going, to give our users only one choice. We wanted to make the III the success it hadn't become and that it deserved.

Unfortunately, we made sure that the Apple II was nowhere close to the market that the [IBM] PC took. 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 up 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.

At one point during the Apple III development, I wrote some fast disk routines for our Pascal system on the II. And I got a lot of flak from Apple III engineers who felt that they shouldn't go on the Apple II. Nothing on the II should be allowed to run as fast or faster than the Apple III. That was the thinking that stuck with the company for three solid years.

It was unfortunate the way things worked out, because we probably put $100 million in advertising, promotion, and research and development into a product that was 3 percent of our revenues. In that same time frame, think what we could have done to improve the Apple II, or how much could have been done by Apple to give us products in IBM's market.

BYTE: *Are you putting more resources to work in that direction now?*

WOZNIAK: Yes, but things don't change in six months. In the Apple III we had a beautiful machine, and we spent a lot of money to try and emphasize that. We were trying to force the world to finally accept the machine because we knew just how good it was.

The PCjr had a poor initial reception like the Apple III did. It came out in the wrong month—the month when Macintosh was going to be perceived as the leader. The PCjr was perceived as an uninteresting product.

They may try for three years to overcome its bad first impression. They may put a lot of their corporate resources into trying to promote the PCjr and lose sight of the PC. The PC will keep selling because it's been accepted and there are a lot of aftermarket companies out there selling software and hardware for it. But if IBM neglects it enough, we may have a chance to turn it over on them in the next few years if we have a better product. ■