TROUBLE-SHOOTERS' CORNER

Ralph Wells

In an article titled "Cooperation" in the June 1978 *Interface Age,* John Craig (former editor of *Kilobaud*) quoted Bob Jones (publisher of *Interface Age*) as saying, "If we all work together at this thing, then we're all going to be successful together." Last month in this column, I tried to emphasize the importance of just this type of cooperation in the field of troubleshooting and used the phrase *synergistic synectics* to describe it.

At about the same time I read John's quote, I also received my current (May 20) copy of *EDN* magazine, containing an article by John Conway entitled "Glitches Can Turn Your 'Simple' Interface Task Into a Nightmare." For anyone interested in troubleshooting microcomputers, this *EDN* article should be required reading. It recounts the debugging of a glitch in the Apple II and reads like a Sherlock Holmes mystery. It's well illustrated and fascinating reading, but at the end I felt that I'd been on a witch-hunt. There was a haunting feeling that the *EDN* group might prefer to see the Apple II, its mentor, Steve Wozniak and *all* 6502 micros in the stocks on a town square in New England.

Glitches—Part II

I'd been contemplating designing in an Apple II as part of a system development for Inmarco (where I work). The requirements called for it to be interfaced to read and write through the 6522 VIA (versatile interface adapter). The gist of the *EDN* article was that the difficulties in using an Apple II for this purpose were so serious that *EDN* was considering discontinuing one of the best series of *practical* development (Apple-Indecomp project) articles I've had the pleasure of following.

After I mulled it over for a couple of days, I bought an Apple. It arrived last Wednesday afternoon. By Friday night we had it working through a kludged VIA. What happened in between is very much like the second episode of a multipart serial TV drama, and just the sort of situation I'd been looking for to illustrate some basic troubleshooting principles.

When last we left this stirring tale . . . Some background is in order. The *EDN* staff developed and beautifully documented an I/O port using a 6820 PIA (peripheral interface adapter). It worked on a 6800 system and on a KIM, but not on Apple II. After some hair-raising adventures (including a fall down an elevator shaft by one of the staff), the villain was tracked down.

In the last scene, our heroes are provided with a potential solution to the basic problem. In the closing scene the *EDN* staff is left with several multiple-choice imponderables: Should they (a) redesign their "simple" interface to include a delay-producing (expensive) buffer, (b) give up on everything or (c) something in between?

The villain turned out to be a Ø0 (phase zero) clock line masquerading as Ø2, aided and abetted by the $\overline{Ø0}$ (inverted Ø0) line passing itself off as Ø1. The *real* Ø2 never even got to the user bus as proclaimed.

As we resume our story, we find . . . The scene shifts to the Sunset Strip in glamorous Hollywood, where we pick up the action as an Apple II is being delivered to the Inmarco engineering department by a couple of Apple lovers. These gentlemen are convinced that the *EDN* article is a gross miscarriage of justice and that, even if there *is* something amiss, it can be easily remedied—not to worry. Billy Shatto (my partner in this adventure) had kludged a VIA using Vector Slit 'N Wrap so that we could easily change the design, and change it we did.

The well-documented *EDN* article provided some schematics and clues. Also helpful was that Apple was preceded by three earlier 6502 micros in my personal collection: Jolt, KIM and PET.

An Inventory of "Tools"

Inmarco has a resident Apple freak named Dave Gordon. If you tear away his accounting-executive facade, you'll find the most turned-on Apple polisher that ever lived. If you read last month's column, you can recall that I feel one of the first steps in practical troubleshooting is to inventory your *tools*, or assets. I also had a lot to say about the "most valuable tool" being synergistic, synectic friendship of the type I referred to in the beginning of this month's column.

Dave is another excellent example of what I'm referring to. He'd be the first to admit that, technically, as an engineer or programmer, he's a great accountant, and this was certainly a technical troubleshooting problem. Even so, I'd rate Dave as the top asset we had at our disposal. We also had experience with interfacing I/O to 6502 microprocessors, since both the Jolt and KIM used PIAs, and the PET used VIAs. We had also interfaced an SWTP printer (PR-40) to both the PET and Dave's Apple, and then used his Apple to write the machine-language software to drive the PET's VIA (through the user plug) from BASIC. The most valuable experience had been with the Apple-printer interface because it used a 6520 PIA and 2708 PROM, both addressed with the bogus Ø2.

For the most part the printer worked, but the uneasy aspect of this experience was that it did *not* do all the things we expected it to do—more on that later. We didn't have the Hewlett-Packard #1615A logic analyzer ($6800) that *EDN* used, but we did have a four-channel Tektronics #549 storage scope and a Paratronics analyzer (which we didn't have to use). We also had some logic probes, DVMs, etc. This inventory of assets seemed to warrant taking the risk, so here we were with an Apple.

And We're Off . . .

After unpacking the Apple II, we ran through a few routine checks with Dave's help. We decided we'd go for broke and try the VIA configuration as shown in Fig. 2. I was pretty sure that the problem could be solved by delaying the Ø0 (used as Ø2 in the Apple). Putting Ø0 through two stages of CMOS should delay it at least as much as the delay caused by the CPU (see Fig. 1) and gate

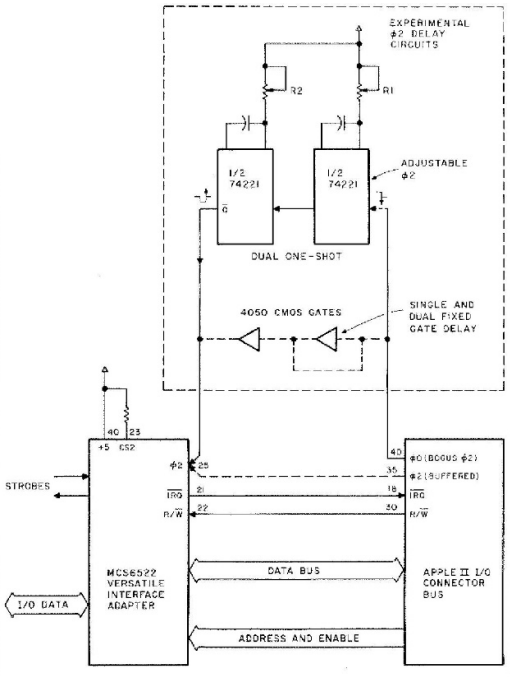

Fig. 1. Experimental versatile interface adapter for Apple II using various methods to provide delay for Ø2 input to pin 25 on the 6522. A buffered Ø2 from the CPU was distributed on unused bus #25 in the last analysis.

B11 (the latter wasn't mentioned in the *EDN* article).

Billy wrote a BASIC program to POKE in the VIA initialization data followed by a WRITE loop. Then he fixed it up. Nothing came out. Was *EDN* right? Had we goofed? (We had.)

Failures like this are a daily occurrence for Billy and me, but Dave was suffering real pain . . . something akin to an expectant father at delivery time. We juggled the software around and looked at some waveforms. The double CMOS gate delay was a lot longer than I had expected, so we cut it down to one—still no output. Then we substituted the PR-40 printer interface. It worked with its own PROM program, but when Billy tried it with the BASIC driver (suitably modified), nothing came out and the day was over.

We changed to the variable design of Fig. 2 so that we could adjust both the leading and trailing edges of an "adjustable" Ø2 coming out of the 74122, and Billy rewired it Thursday morning. By the time I arrived at work, he had discovered that the PIA could be initialized and driven by a machine-language program or from the Apple monitor, but *not* from a POKE command in BASIC. The scope showed two access pulses where we expected to see one. The first was coincident with the proper data pulse—the second wasn't. It was turning itself off about as fast as it was turning on!

We felt that the major problem was still the hardware, and it would be driven by machine language anyway, so Billy rewrote the test program in machine code, and now the PIA worked! The *EDN* article quoted Steve Wozniak (the chief Apple grower) as saying that the 6502 had an odd habit of generating a READ before a WRITE during store operation. My experience with the Jolt and PET didn't jibe with that. I don't like loose ends like this floating around, but for the time being we decided to ignore BASIC and stick with machine code. (Days later, we found a bug in our program.)

Billy changed the program back to double-delayed Ø0 VIA, and we tried to write with it. Success! We could move the critical edges back and forth by tweaking R1 and R2. When the rising edge of this "adjustable" Ø2 was aligned with the Ø0 (pin 40), the VIA would fail. When we moved it to coincide with the real Ø2 on pin 39 of the 6502 or as much as 200 ns beyond that time, it worked! So much for "writing" out with the VIA.

Wozniak was also quoted as saying that the PIA was probably only useful as a write-out-to device. Our project called for read-after-write on a 9-track tape deck at 25 ips. That meant that we would have to both *read* and write, synchronize and index in less than 40 usec—could we do it with the Apple?

Our 6800 had been too slow. Billy rewrote the program to read and display in a high-speed loop. It worked! I was relieved, Billy was happy and Dave was ecstatic. His faith in his beloved Apple was vindicated. When we told him that it wasn't over yet and that we had *no* intention of putting adjustable pulses on every I/O board with a PIA or VIA, it cooled him off a little, but he walked off into the sunset muttering something about "that being only a *little* problem."

A Can of Green Worms

In last month's column I said that the second step in troubleshooting was to define the problem and that the third step was to "fix" it. By now we felt we had the fundamental problem fairly well defined and turned our attention toward what to do about it. We knew that the real Ø2 coming from pin 39 on the 6502 was the correct phase and formed to do what the chip manufacturers (MOS Technology, Synertek and Rockwell) said it should do, so why not just use it instead of the Ø0?

Fourteen MHz works out to have around a 35 ns pulse width, and the Ø2 phase error was almost that much. Would replacing the Ø0 with a Ø2 for the whole system be the answer? If the feedback circuit didn't cause a race condition, we might just get away with it. We decided to try.

Billy cut the etch so that only the gate (B11) was driven from B1 (74S175 D flip-flop wired to toggle). The real Ø2 from pin 39 on the CPU was buffered by a TTL chip we installed on the kludge section of the Apple (a premeditated plus in the Apple design).

The result was a disaster. Attempts to trace the problem through the feedback loops produced a jumble of unintelligible green squiggles on the four-channel scope. We had opened a can of worms—green ones. We quickly clapped the lid back on and restored the original Ø0 wiring . . . and sought a practical compromise.

Our past experience with memories, both static and dynamic, indicated that the phase error probably wouldn't bother them either way; so when it came right down to it, the only critical devices were most likely those on the interface bus. An examination showed that there were two unused lines (three if you count the user). One of them (pin 35) was adjacent to the other clock pulses, so Billy reconnected the original drive circuit and then ran a line from the real Ø2 down through the buffer (see Fig. 1) and back to pin 35 on the interface bus.

Now when we tried our VIA kludge (without a delay on board), it worked when driven from the "new Ø2" and failed when driven from the original line. This means that if we use VIAs, PIAs or the unbuffered PR-40 driver in our own designs, we will use the "new" *real* Ø2 line on pin 35. If we use a buffered design, such as the Apple PROM burner described in *EDN*, we can have a choice. Either line *should* work, but to play it safe I'd use the real (buffered) Ø2.

Later we tried the Ø2 right out of the CPU (no buffer or buffer delay). If we routed the wire *exactly* right (and the moon was full), then it would work, but it needed the extra delay as a safety margin.

Will the Real Ø1 Please Stand Up!

So far we've been discussing the use, misuse and misnomering of Ø0 and Ø2 (see Fig. 3). What about Ø1? We did a lot of worrying about Ø1, probably unnecessarily. To start with, the waveform generated by our Apple 6502 Ø1 doesn't match the specifications shown in the MOS Technology book or the *EDN* article. Ø1 is supposed to be shorter than Ø2 on both ends. The leading edge was OK, but the trailing edge was coincident (as best we could read it on our scope), or maybe a few nanoseconds longer than Ø2.

On Friday afternoon after we got Ø2 straightened out, we checked the KIM and our just-arrived VIM 1, which used a Synertek 6502. The KIM looked just like the Apple, but the VIM 1 looked as the textbook said it should. With the exception of a clamp diode, the crystal clock circuits of the KIM and VIM are

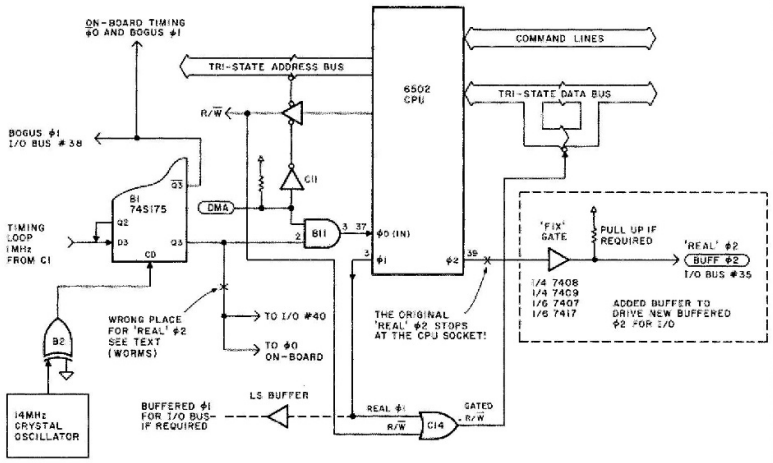

Fig. 2. Apple II clock circuitry showing "fix" buffer and waveform generation described in text.

(continued on page 116)

TROUBLE-SHOOTERS' CORNER

(from page 11)

practically identical. The VIM drives (successfully) the same VIA we've been discussing.

Now let's take a closer look at Steve Wozniak's use of Ø1. There are *two* Ø1s, and they're *not* the same. The real Ø1 comes out of pin 3 on the 6502, goes through gate C14, and gates the buffer amplifiers for the data bus—a critical timing function. This is a good straightforward design.

The other (bogus) line labeled Ø1 comes out of the $\overline{Q}$ (B1, pin 6) of the same flip-flop that drives the bogus Ø2 (really Ø0 and the gate delay of B11). This is the bogus Ø1 which appears on the interface bus at pin 38. It is really the inverse of the bogus Ø2 (pin 40) and is closer to being $\overline{Ø0}$ than the real Ø1.

I haven't checked all the places where this bogus Ø1 is used, but it *is* operating successfully on the main board and video, so it's probably OK. Since we have no intention of using it with the VIAs or PIAs, we'll let it off with a warning: *It has been determined that the use of Ø1 on the user bus may be dangerous to the health of any I/O design.* Since gate C14 is an LS device, the real Ø1 should be able to drive one or more additional buffers, if a new design should prove critical in this area. For now, we'll leave it—as is.

Some Troubleshooting Fundamentals

I said earlier that I had been looking for something like this with which to illustrate some of the theory of troubleshooting. Let's get into the fundamentals of "defining the problem." There are probably as many ways of defining a problem as there are problems—but the majority of techniques for troubleshooting hardware bugs fall into three methods and the combinations formed thereof.

The fastest and usually most practical is *substitution*. The next is *signal tracing*. These two are indigenous to nearly all electronic devices. A third (peculiar to computers) is hardware/software trade-off. Eventually, I expect to dedicate a whole column to each, but for now we'll take a little "preview of coming attractions."

Substitution. Substituting is even more effective in troubleshooting than on the playing field. It requires a minimum of knowledge and training, and can usually isolate a problem faster than any other method. It's my first choice whenever practical.

Substitution has two major drawbacks. The most common is that there is often no spare available to substitute. The second is best illustrated by the example of replacing a fuse. If the fault is still present, the "substitute" fuse is destroyed. This is a sobering thought when expensive or hard-to-get devices may become "fuses."

If I intend to *swap* chips or boards, I try always to check the supply voltages as close to the suspected fault as possible, *before* swapping. A solder bridge or loose bit of metal lodged between two device leads can zap a $50 IC in a split second. If the supply voltages are OK and the circuit was OK at some earlier time, then burnouts due to swapping on 5 V TTL/MOS systems are rare. On the other hand, if you've been fooling around with a soldering iron or reconnecting cables, then *beware!*

Signal Tracing. If substitution isn't practical, then your next best bet is signal tracing. If you've ever tried to follow a carnival con-man with his three-card monte or shell-game movements, then you have some idea of what it was like trying to trace out the results of substituting the real Ø2 for the Ø0 in the foregoing example.

Signal tracing usually falls into one of three general categories: signal detection (using scopes, pulse catchers, voltmeters, etc.), signal injection (using pulse injectors, oscillators, waveform generators, etc.), and a relatively new system of *signature analysis*. Signature analysis is too new (and expensive) for me to use, but as devices become more complex it may become the best practical way to accomplish in-field signal tracing. The other two techniques were used extensively in the example at hand.

The most powerful tool is the Hewlett-Packard #1615A logic analyzer used by *EDN*. I doubt whether many personal computing fans can afford one, or even have access to one (certainly I don't). On the other hand, a dual-trace oscilloscope is usually available in a metropolitan area if you have enough friends. A few years ago a voltmeter, VTVM (vacuum tube voltmeter) or DVM (digital voltmeter) was an absolute necessity for any kind of electronic troubleshooting, and when it comes to power supply and analog problems, they're still a *must*.

The advent of TTL and CMOS technology produced some new devices: the pulse-probes. They are rapidly becoming a must, and I hope we'll be doing a special article on them in the near future. If you have access to a good dc scope, it can usually duplicate the functions of a meter and/or a pulse probe.

The most common method of signal tracing is to put a normal signal into an input and follow (or trace) it from one device to the next using one of these tracers, until you find a place where it misbehaves. This can get rather complex, as the *EDN* article testifies. A not-so-often-used method is to inject a signal into the system and watch what comes out. We did this in testing the VIAs by building up a counter circuit that would provide a known pattern on the VIA input bus.

Hardware or Software?

Many troubleshooting aids can be implemented in either hardware *or* software. Most large corporate-development efforts (*EDN* included) are polarized into hardware- and software-type personnel.

Personal computing is breaking down this barrier, and one of the advantages of mixing these two disciplines is that the trade-offs between them can be used for troubleshooting. On the one hand, I mentioned the *hardware* counter we used to provide a known digital input. On the *software* side, we wrote loop programs that generated an alternate AA (1010 1010) and 55 (0101 0101) pattern so that we could trace through hardware with the scope.

The diagnostic system built into the PET is a giant step in the direction of using software to troubleshoot hardware. When people get tired of writing programs for games, maybe they'll put some effort into writing diagnostic programs directed towards isolating faults. Of course, the basic fallacy occurs when you try to use a faulty system to troubleshoot a faulty system.

Let me point out the spots where we used some of the specific, fundamental troubleshooting methods I've just described. By *substituting* Dave's PR-40 printer interface with its PIA, we were able to identify the problem of using POKEs from BASIC (more on that later). By generating test patterns with *software* loops, we were able to *trace* the bogus Ø2 waveforms and compare them using an oscilloscope. By *swapping* back and forth between the VIA and PIA and *signal tracing*, we established that, with a marginal

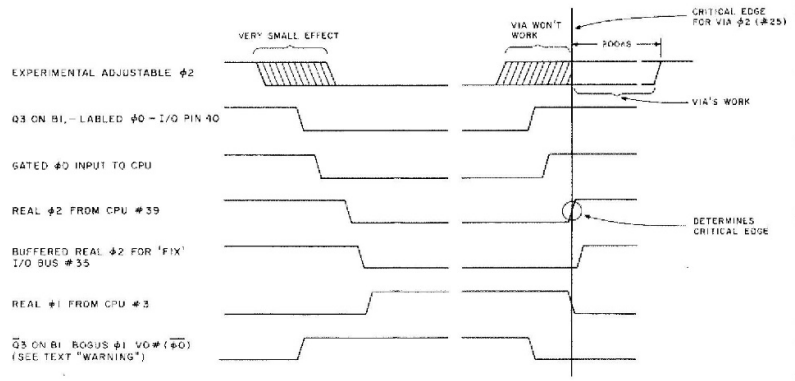

Fig. 3. One MHz clock timing diagram for Apple II illustrating errors uncovered by EDN and the Inmarco "fix."

system:

1. It's easier to write than read.
2. The PIA was less critical than the VIA.
3. Both worked if the rising edge of the driving pulse was delayed with respect to the real Ø2.
4. The 2708 memory would function on the same card as the PIA, even when the card was driven by different Ø2s.
5. The chances are very good that any other member of the 68XX or 65XX I/O devices can be expected to work on a modified Apple II, but we have no intention of *proving* it.
6. The "fix" for our Apple II consists of one TTL gate delay costing less than 25¢ (remember what I said about the *defining* being tough and the *fix* being easy?).

Things That Go Bump in the Night

Haunted houses, cemeteries and strange noises late at night are all part of a basic fear in mankind—the fear of the unknown—the *what-if* syndrome. Last weekend I was plagued with many fears. This weekend they're (nearly) all gone. There was still the problem of why we couldn't POKE into the PIA or VIA from BASIC.

After modifying the board for the real Ø2 buffer, we tried the BASIC program experiment again, with the same negative result. The VIA would work with machine code but not with BASIC, and the same double pulse showed up where there should only have been one. It really wasn't a problem for us because we expected to program in machine code, but it bothered me.

Synergistic Synectics

You may recall that I rated the synectic relationship of Dave Gordon as a top priority "asset." As it worked out, we didn't need this "tool" until Friday, but it was there and probably cut a full day off the debugging. As I said, Dave is neither an engineer nor a programmer, but he has assets that make these two skills redundant.

Dave is a gregarious, natural-born "horse trader" in the best American tradition. Personal computing is a fertile field for bartering, and Dave revels in it. His collection of programs and accessories for the Apple II is probably the largest in existence, but he "collects" something else with even more vigor: friends. He's up half the night with them, running new programs, expanding horizons, swapping "things" and information—all with an infectious, unbridled enthusiasm. If he doesn't meet enough people through the various computer clubs he belongs to, he'll have a barbecue at his home with the top brains in the Apple Corps.

This is what synergistic synectics is all about, and when an article like the *EDN* piece threatens their rock-solid belief that the Apple II is the best-thing-that-ever-happened-to-humanity, things happen.

The day after we started debugging, I found myself on the phone being introduced to Dave's friends, who were eager to help out. It turns out that Craig Vaughn and Sandy Tiedman had encountered the same problem and "fixed" it by using CMOS delays on the PIA boards just as we had done the first day—but they had *fixed the bug long before the* EDN *article was published.*

While discussing with Sandy what we'd found on Friday afternoon, I described the bug we'd observed in using the BASIC POKE command. He called back later to report that his Apple II worked fine with BASIC. On Saturday I had a long talk with Craig, and he agreed to try the same experiment. When we got back to software, we found our

own bug, and now our BASIC is home free. The point to be made is, how would you tackle a problem like this—without friends? How long would it take to solve it? How much manpower? How much equipment?

A Rebuttal

A "box" in the *EDN* article recounted all the shortcomings of the 6502 programming language. At the end, Mr. Jack Hemenway asked the rhetorical question, "Of course programmers can get around all of the above problems, but why should they have to?" Now let's be fair! He obviously preferred to program the 6800—so do I, but there *are* reasons why the 6502 is designed into so many computers.

Since I have eleven 6800 micros, an excellent 6800 program development system and a personal preference for 6800 language, why did I buy an Apple (6502) . . . even after reading the *EDN* article? Probably the most important reason is speed. A single byte register is faster than a double byte; it's a trade-off between hardware and software. If you want the speed, you write more code.

Hemenway made no mention of some of the 6502's pluses, such as the three-way branch capability after a bit test, the decimal-binary shift and the indirect address capabilities. No mention was made of the "sweet sixteen" ROM interpreter, which gives the Apple more double-byte power than any 6800 system I'm aware of. If you want to prove the speed difference to yourself, try benchmarking a program in BASIC (such as a bubble sort) on a PET (6502) and then on an SWTP, Altair 680 or Sphere (6800s).

I personally bought all of these machines. For my money, the 6502 can outperform the 6800 when it comes to speed, but it's not as easy to program. Do you design a microcomputer to please the end user or the programmer? I've been reading *EDN* ever since its inception, and I can't recall its ever being as editorially unfair as it was in this article.

Fan-atical Behaviors

If you've ever witnessed the spectacular behavior of a Dodger fan at World Series time, you've seen histrionics and heard exaggerations (from otherwise "normal" people) that make a used-car salesman look like a saint. In Europe, soccer matches can lead to bloodshed. If you've ever watched a PET lover battle it out toe-to-toe with a TRS-80 devotee, you've seen elements of the same phenomenon—but when the shouting's over, even the most dedicated *fan* knows that in Mudville even the mighty Casey *can* strike out.

I enjoy observing Dave and the members of the Apple Corps. I know that Dave realizes that the Apple has a design error in it. Even if it can be fixed for 25¢, any bug that requires the amount of troubleshooting effort that this one did is *serious*. He may verbally refer to it as a "little one," but he knows better, and (in private) admits it.

Rationality through Diversity

If *EDN* were to scrap the Apple-Indecomp project because of a "two-bit" defect, I, for one, would consider it grounds for serious criticism. *EDN* might well examine the factors that have made the Apple one of the largest selling micros in history! Can all those people be *that* wrong?

In delineating the shortcomings of the 6502, Jack Hemenway cries out to "move on" to a different microprocessor (probably 6800). I agree that *EDN* should move on, *but not yet!* The 6809 and 8086 *double*-byte CPUs are just around the corner, and I know of at least two houses that intend to put them into computers as soon as they're commercially possible. I heartily agree with *EDN*'s original choice of CPUs. In my opinion the Apple II was, and is, the best one for the Indecomp project (until the doubles appear).

Personal bias is one thing; responsible editorial bias in a major publication is something else. I repeat Bob Jones's statement: "If we all work together on this thing—we can all be successful, *together.*" Synergism!

If you have interesting problems, solutions, comments, or indeed, anything you believe can help others, or that we can help you with in this area, drop a line to: Troubleshooters' Corner, *Kilobaud* Magazine, Peterborough NH 03458.