SOFTALK

Margot Comstock Tommervik — Editor
Bill Depew — Technical Editor
William V. R. Smith — Advertising Manager
Al Tommervik — Marketing and Production
Kurt A. Wahlner — Art Director
John Mitchell — Assistant Production Manager
Robert Koehler — Circulation Manager
Matthew Yuen — Editorial Assistant
Kirin Tommervik — Editorial Assistant
Paul Essick — Circulation

 Editorial offices at 10432 Burbank Boulevard, North Hollywood, CA 91601; (213) 980-5074.

September fulfillment courtesy of Apple Computer Inc., Cupertino, California.

Apple is a trademark of Apple Computer Inc., Cupertino, California.

Composition by Photographics, Hollywood, California.

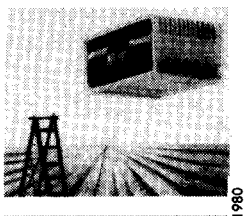
Printing by California Offset Printers, Glendale, California.

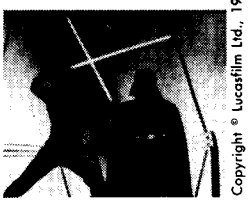
SOFTALK is published monthly for $12.00 per year by Softalk Publishing Inc., 10432 Burbank Boulevard, North Hollywood, CA 91601. Controlled circulation pending at Burbank, CA.

Postmaster: Send address changes to Softalk, 10432 Burbank Boulevard, North Hollywood, CA 91601.

ADVERTISER'S INDEX

TABLE OF CONTENTS

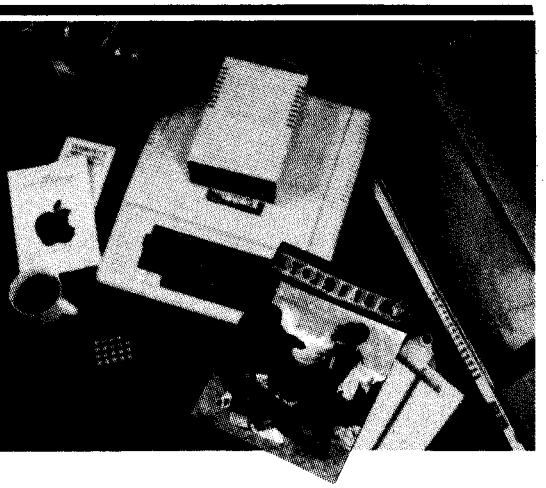




FEATURES

PREVIEWS

Whispering about October . . . can we stay afloat in the wake of the software pirate? . . . Apples offer independence to New Jersey handicapped . . . bestsellers . . . part one of assembly language for the novice . . . Exec Personal . . . the man who refused to buy a disk drive . . . Halloween contest: bobbing for apples . . . and more.



WIN A PRIZE FOR YOUR APPLE!

There's speculation aplenty about what programs will head *Softalk*'s software bestseller list, which begins next month. At least fifteen companies believe one or more of their programs will be in the top five. They can't all be right, and here's your chance to match your wits against theirs.

To the person who most accurately predicts the actual results of our first poll, we'll award any one product manufactured by any advertiser in this (September) issue of *Softalk*, up to a retail value of $150. If you choose something more expensive than that, we'll apply $150 toward its purchase.

To enter, predict which five programs, nationwide, will head the bestseller list, and write these into the coupon in order. Points will be awarded on this basis: six points for correctly naming the leading seller, five points for predicting the correct second-place bestseller, four points for the third, three points for fourth, and two points for fifth. One point will be awarded for each program correctly named in the top five but not correctly placed. Entry with the most accumulated points wins. In the case of ties, Apple's random number generator will be used to select the winner.

Entries must be postmarked no later than September 15; the winner will be announced in the November *Softalk*. Your prize will be purchased by *Softalk* from your retailer, and you'll be able to pick it up at that store.

(Contest is open to all Apple owners and their immediate families except those associated in any way with *Softalk*. Use of computers in deriving answers to this contest is strictly encouraged. Multiple entries are acceptable, but you may not enter more than one set of predictions on a coupon. Photocopies of the coupon are okay, as are handmade copies on a plain sheet of paper, the back of an envelope, the margin of your daily newspaper, whatever, as long as they're legible and reproduce the entire coupon.)

Mail coupon to: Softalk Circulation, 10432 Burbank Blvd., North Hollywood, CA 91601.

My predictions are: 1. _____ 2. _____
3. _____ 4. _____ 5. _____

Name _____ Address _____
City _____ State _____ Zip _____

I have examined the advertisements in this issue of *Softalk*. If I win, the prize I'd like is _____
I understand that my selection must be a product of one of *Softalk*'s advertisers, although it need not be the particular product advertised. I accept *Softalk*'s bestseller list as the standard of judgment for all entries, and recognize that in case of a tie, the decision of Apple's random generator is final.

_____
(your retailer) (your autograph)

SIGN UP FOR SOFTALK

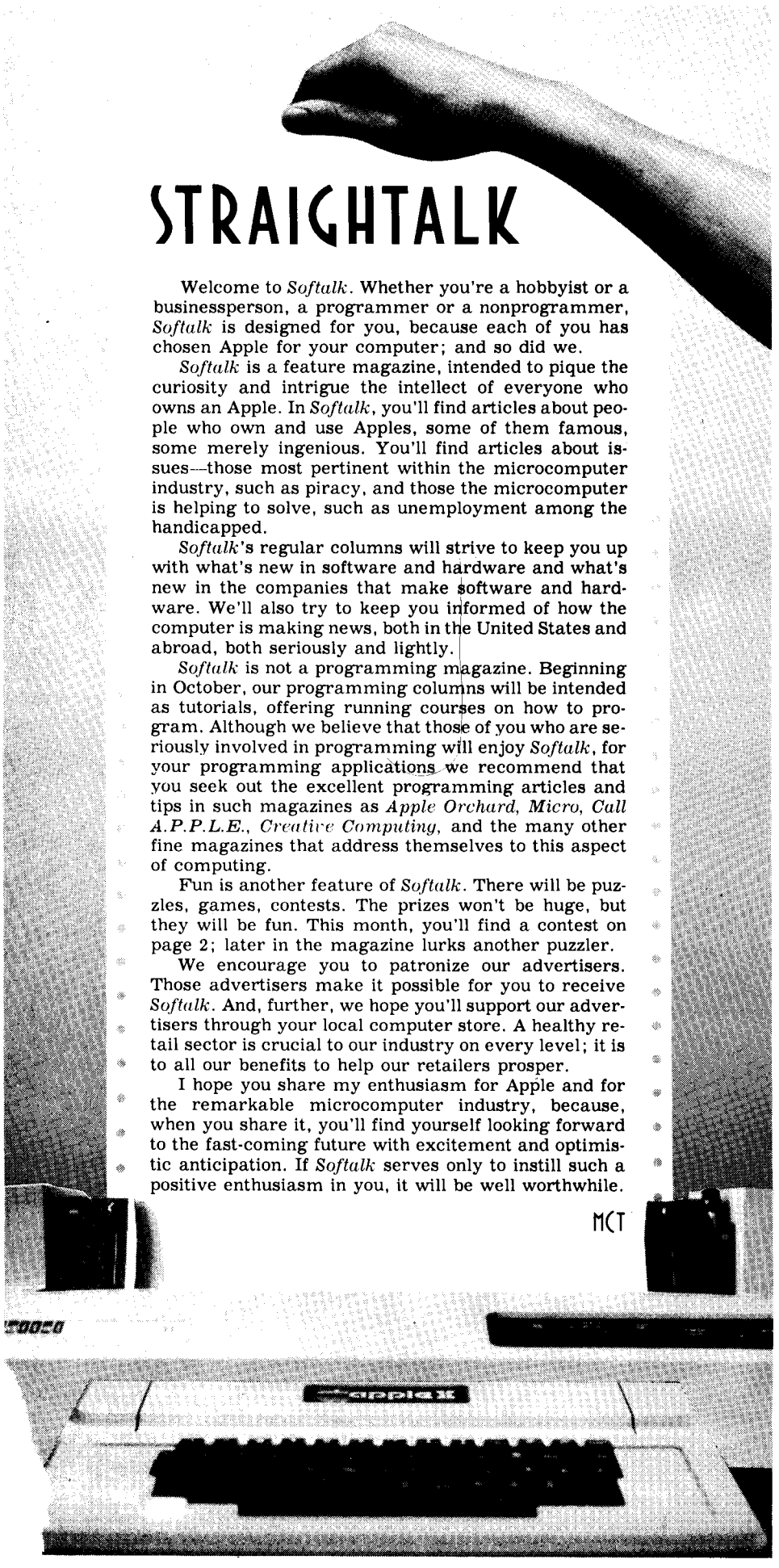
To ensure receiving *Softalk* free each month, fill out coupon and mail to: Softalk Circulation, 10432 Burbank Boulevard, North Hollywood, CA 91601.

Name _____
Address _____
City _____ State _____
Zip _____ Apple Ser. No. _____

This *Softalk* was mailed via Apple Computer's mailing list. *Softalk*'s own mailing list lags behind Apple's; consequently, you could miss several issues.

To prevent that, complete the coupon, or affix to it the mailing label from this *Softalk*, and send it in.

Include your Apple's serial number. You can find it above your name on *Softalk*'s label or on the bottom of your Apple.



STRAIGHTALK

Welcome to *Softalk*. Whether you're a hobbyist or a businessperson, a programmer or a nonprogrammer, *Softalk* is designed for you, because each of you has chosen Apple for your computer; and so did we.

Softalk is a feature magazine, intended to pique the curiosity and intrigue the intellect of everyone who owns an Apple. In *Softalk*, you'll find articles about people who own and use Apples, some of them famous, some merely ingenious. You'll find articles about issues—those most pertinent within the microcomputer industry, such as piracy, and those the microcomputer is helping to solve, such as unemployment among the handicapped.

Softalk's regular columns will strive to keep you up with what's new in software and hardware and what's new in the companies that make software and hardware. We'll also try to keep you informed of how the computer is making news, both in the United States and abroad, both seriously and lightly.

Softalk is not a programming magazine. Beginning in October, our programming columns will be intended as tutorials, offering running courses on how to program. Although we believe that those of you who are seriously involved in programming will enjoy *Softalk*, for your programming applications we recommend that you seek out the excellent programming articles and tips in such magazines as *Apple Orchard*, *Micro*, *Call A.P.P.L.E.*, *Creative Computing*, and the many other fine magazines that address themselves to this aspect of computing.

Fun is another feature of *Softalk*. There will be puzzles, games, contests. The prizes won't be huge, but they will be fun. This month, you'll find a contest on page 2; later in the magazine lurks another puzzler.

We encourage you to patronize our advertisers. Those advertisers make it possible for you to receive *Softalk*. And, further, we hope you'll support our advertisers through your local computer store. A healthy retail sector is crucial to our industry on every level; it is to all our benefits to help our retailers prosper.

I hope you share my enthusiasm for Apple and for the remarkable microcomputer industry, because, when you share it, you'll find yourself looking forward to the fast-coming future with excitement and optimistic anticipation. If *Softalk* serves only to instill such a positive enthusiasm in you, it will be well worthwhile.

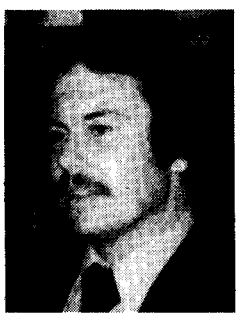
MCT

TRADETALK

☐ In a major realignment of the executive suite at **Personal Software,** Sunnyvale, California, **Terry Opdendyk** has been named president and chief operating officer.

Founder and former president **Dan Fylstra** has assumed the position of chairman of the board and chief executive officer. Founder **Peter Jennings** has assumed the position of vice-president of advanced research and development.

Opdendyk joins Personal from Intel, where he was on the corporate staff in charge of human resources. Previously at Intel he had headed the commercial systems division and had managed microcomputer software development activities. Opdendyk joined Intel from Hewlett-Packard, where he had been a software engineer and project manager.



(Above) Among the many Apple Computer Inc. representatives at Computerland's vendor fair were Jean Richardson, manager of marketing services, and Phil Roybal, product marketing manager. (Below) Ken Silverman, president of International Apple Corps, and John Wilbur (left) of M&R Enterprises.

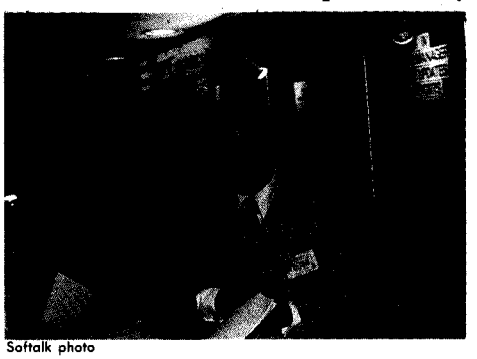
Softalk photos

☐ **Microage Computer Stores,** headquartered in Tempe, Arizona, have announced the awarding of the first two franchises in an expansion drive that the company projects will result in 100 retail franchises in the next two years. Marketing targets for the outlets are the small business and the professional user.

☐ **On-Line Systems** has relocated from the hectic environs of the Los Angeles suburb of Simi to the more contemplative surroundings of Coarsegold, California, just outside of Yosemite Park.

☐ **Computerland,** the nationwide chain of retail computer stores, held their 1980 Computerland International Conference in San Francisco, California, July 18-21.

Approximately one hundred Computerland retail stores were represented by



Softalk photo

Milton Nelson, western regional sales manager of Eaton Corporation's Count Control/Systems Division, expounds on the merits of an Eaton printer at Computerland's vendor fair, held in conjunction with the 1980 Computerland International Conference for the company's franchises. Fifty-three vendors displayed their wares at the fair. More than 100 retail stores were represented at the conference.

more than twice that many attendees at the series of seminars and meetings.

Marian Murphy, products manager at Computerland corporate, arranged a vendor fair for July 20 at which vendors of Computerland-carried product could display their wares. Among the fifty-three companies that accepted the invitation, Apple Computer Inc. and various vendors who support Apple products were prominent.

☐ Representative of the explosive growth in the microcomputer industry is **D. C. Hayes Associates.** The micromodem manufacturer just announced its third move in less than as many years. **Dennis Hayes**, president of the company, said the move was necessitated by a threefold growth in manufacturing employees in the last twelve months. Company's new address is 5835 Peachtree Corners East, Norcross, GA 30092.

☐ Also indicative of industry's growth is the name change at **Mountain Computer,** nee Mountain Hardware. The manufacturer of Apple peripherals has always been heavily into software development as it pertained to its own products. Now, according to **Avery Dee**, the firm will capitalize on its in-house programming capability to create specialized software packages independent of its hardware. First product may yet be seen this year.

GOTO 27

EXEC APPLE:

John Couch's Software Vision

BY ALLAN TOMMERVIK

If John Couch's vision of the software future comes to fruition, *datagramming* will replace *programming* as the operative word for microcomputers in the current decade.

Any conversation with John Couch about microcomputers centers on vision—on what will be, not what is. But unlike Mohammed, Moses, and Crazy Horse, who went to the mountain to get their visions, John Couch has his. And rather than the mountain, he's gone to the gulch.

In this case, the gulch is silicon gulch, and it's the Santa Clara Valley south of San Francisco. Specifically, Couch toils away at actualizing his vision as vice-president of software for Apple Computer in Cupertino, California.

The ABD and HP. Couch's education and wealth of software development experience leave him with few peers as a seer. He majored in computer sciences in the late 1960s at the University of California, Berkeley, when that was the only school in the country offering a major in that curriculum. He continued through his master of science degree and eighteen months of doctoral work before the pressures of supporting his family and the challenges of the industrial world lured him out of the university environment. He jocularly claims one of the most widely held of all degrees, the ABD—*a*ll *b*ut *d*issertation.

Prior to joining Apple, Couch spent seven years in the software development group at Hewlett-Packard. His initial assignment was assisting in building the FORTRAN and BASIC compilers for the HP-3000. At the time of his departure, he was in charge of all software development for that system and had been responsible for the architecture of and languages developed for the HP-300.

The Time Has Come. In its most fundamental form, datagramming entails entering the data to be operated on, setting specifications that will inform the computer of the operations to be performed, and then letting the computer do the programming (see page 24). Datagramming will allow the nonprogrammer the access to and ability to manipulate data that had been available previously only to programmers.

It's a concept whose time has come—Couch points to Personal Software's *VisiCalc* as the first software package to pioneer the field. The easiest way to come to grips with the concept is to understand what led Couch to the conclusion that this was the path of the future.

His thought processes stemmed from a general realization that most computer systems have historically followed the same development path—which has yet to lead to anything other than a new system down the road—and from his experience in setting up a microcomputer for his father's health spa business while he was still at Hewlett-Packard.

Couch's view of historical computer system development is not radical or revisionist in the least. His differences with traditionalist views are not in what has happened, but in whether what has happened should continue to happen in a self-perpetuating cycle.

The Common Course. Traditionally, a new computer system has been generated by hardware engineers with little regard for software support; the completed prototype is turned over to the software department for them to do with what they will. Software developers either adapt an existing language or search for a new one that will match well the uses that are foreseen for the hardware.

What happens, of course, is that as soon as the system—now complete with advanced hardware and appropriate software—hits the market, the users find the limitations annoying. User complaints growing out of such limitations have been promptly addressed by the software developers. But, rather than building tools for the nonprogrammer, they've just continued to make the programmer's tools more sophisticated. As Couch outlines the process:

"Their first solution is always to add syntax to the existing language to make it a more powerful tool for the programmer. But that's a limited avenue that eventually runs into the law of diminishing returns. At that point, a new, more powerful language is usually introduced to address still more programming problems."

The new language is then updated with additional syntax until a more powerful system is introduced to the marketplace, which is exactly where this historical overview started.

Couch is quick to point out that Apple Computer has, until now, taken this traditional path. "We've gone from Integer Basic to Applesoft floating point to Pascal, each in an effort to harness a more powerful language and provide better tools for the application programmer. But personal computers are for nonprogrammers as well as for application programmers, so now our development is going a different direction."

It Springs from a Spa. *Why* that different direction is best il-

lustrated by Couch's own initial experience with a microcomputer. Couch's father, managing a health spa, complained that the business was getting away from him—that he didn't have the right data to know how and why things were happening to the enterprise.

"The microcomputer will be to the eighties what the calculator was to the seventies."

Couch induced him to buy a microcomputer. With his father's data requirements in hand, Couch *fils* programmed the system to give Couch *pere* the data he needed. But it was never enough.

"Every time I'd go home, he'd ask if I couldn't just add this or that to the program to give him more insight into his business. It got to the point where I almost didn't want to go home, because going home entailed writing more programs for Dad's computer."

Through his dad, Couch began to realize the limitations of traditional programming, no matter how powerful or versatile the language.

"What my father needed was a computer environment where he could specify the parameters of what he needed, and the computer would then perform the programming functions to give him the data in the form he wanted."

From this experience, the concept of datagramming formed and grew.

Little Fish Spawned Quality. It's a bit of an overstatement to say that Couch's vision of a new software approach became an obsession, but the record does show that he soon thereafter jumped from big fish HP to the embryonic Apple company in pursuit of the concept.

"I took this datagramming concept to HP and made a proposal. They liked the idea, but they wanted to mount it on a one-hundred-fifty-thousand-dollar machine. My father could only afford a personal computer, and I felt that the most valuable application for the concept was in the personal computing area, serving people like him.

"The microcomputer will be to the eighties what the calculator was to the seventies. Every home, professional user, and most small businesses will have one. The small businessperson and the hobbyist, who can't afford custom software support, need datagramming to get custom results."

Apple Profit Center Profits Apple Users

Nobody's accused Apple of being just another company, and a look at how its executives have structured their software lab reveals a telling reason for its success in the microcomputer industry.

What John Couch, software development and publications topper, has as his domain is a separate profit center within the company. The implications of that structure are immense for the Apple owner. As Couch explains it:

"In the normal company, once a system gets on the market, sixty to seventy percent of the programmers' time is taken up with maintenance and improvement of the system, and very little is left for the development of new concepts and ideas. And if the software department attempts to expand or to give itself more research and development capability, managers of other departments are likely to question the wisdom of those expenditures in relation to what the investment could achieve in their departments.

"Furthermore, hardware development is usually done independent of the software lab. The harmonious interaction of the two labs to achieve the most comprehensive and functional system is missing."

Partners in Growth and Profit. "The first difference at Apple, of course, is that software development goes hand-in-hand with the development of new hardware, and in most cases drives the hardware design.

"The second major difference is that I have the freedom and flexibility to structure research and development groups as long as I can justify it on the bottom line."

The fact that the software lab has expanded tenfold during Couch's eighteen months as vice-president testifies to the health of his bottom line.

Couch lays claim to the best software group in the Santa Clara Valley, no mean boast to those who know that the area is so rife with electronics firms of all pedigrees that it's affectionately known as silicon gulch.

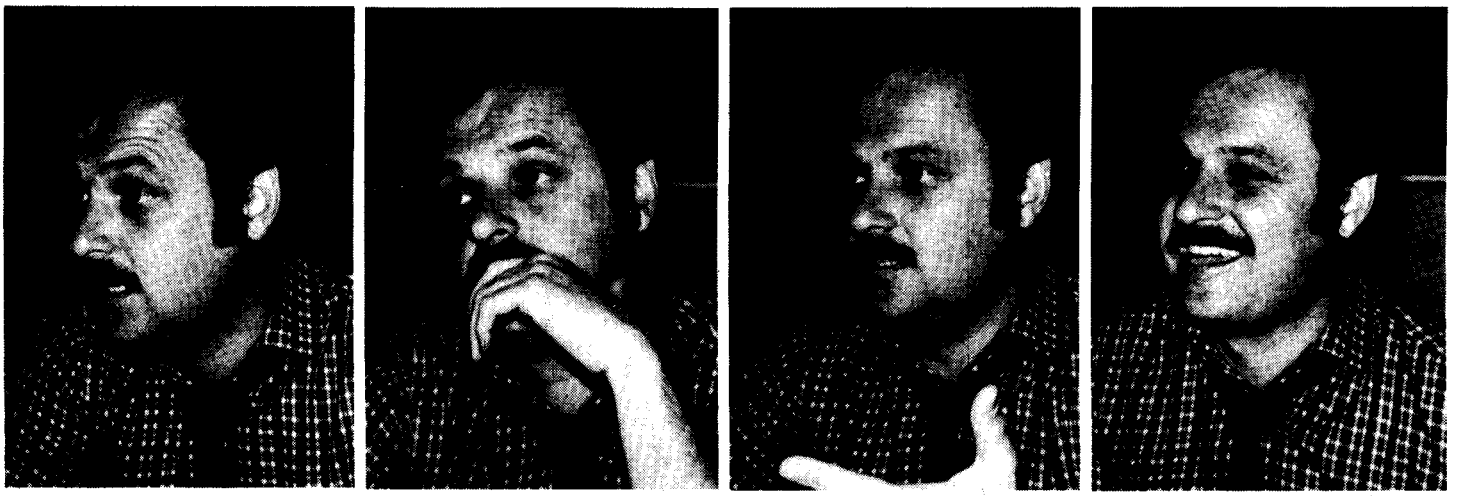
Sparing himself the charge of hyperbole, Couch attributes that statement to an executive from a larger, noncompeting company.

There's little doubt that Apple's software lab is among the most comprehensive.

One programming group is active in research and development projects that are entirely hardware and product-line independent—seeking new programming fundamentals.

Another group functions as liaison with independent software vendors who support the Apple hardware. This group

GOTO 24



Softalk photos

So it came to pass that Couch reversed the usual executive search paradigm. Instead of a company screening him for suitability, he began screening the microcomputing companies for a match with his philosophy. Steve Jobs and Mike Markkula, two of Apple's founders, convinced Couch that Apple was the right company with two telling arguments:

First, they showed Couch that his software vision was shared by Apple; and second, they agreed that Apple hardware engineers should—and would—develop new products in complete liaison with the software lab to ensure system compatibility and usefulness.

"I was also attracted by Apple's dedication to a quality product. I had been prepared to try to start my own company, but my vision and Apple's vision were the same."

Apple's Couch Vision. Couch now oversees approximately one hundred persons working on software development and documentation manuals. "My management style is management by walking around. Since I'm only as good and effective as the people in the group, it's important that our communication lines be wide open. There's no better way to find out what's happening than to see it firsthand.

"That doesn't sound as scientific as some of the fancy names theorists have put on management systems, but I see my function as making sure that the Apple vision filters down to everyone.

"I'll walk around and stop at someone's desk. We might not even discuss his particular project. What we'll talk about is the software strategy and maintaining the enthusiasm, esprit de corps, and quality that are elements of the Apple vision.

"If that vision permeates the group, our individual efforts will succeed. That's our main challenge as a company, to keep the founders' vision alive and clear to everyone as we grow."

It's vision that makes John Couch run. Datagramming is the prize that he pursues. And all Apple owners will be the winners when he crosses the finish line.

"My management style is management by walking around."

Doing an Apple

The hiring of John Couch exemplifies one significant result of Apple's vision. Even in its infancy, Apple attracted top talent—people who often took significant cuts in pay and responsibility to share in the realization of Apple's potential.

The result has been that the company has not outgrown its management, as so many expanding companies have done in the past. Rather, Apple has grown up to the capabilities of the executives already in place.

Couch credits his family, wife Diana and two of his three children, Tiffany, now age nine, and Kristopher, now five, for standing beside him when he decided to take the same cuts to join Apple. His youngest son, Jonathan, joined the family since the move.

Apple has had such notable success in hiring powerful executives into presumably lesser jobs that other companies are now attempting to follow the same course. And when those other companies achieve this, they call it "doing an Apple."

DEALING WITH DOS from Assembly Language

BY WILLIAM DEPEW

When you leave Basic, you leave the world of DOS-supported languages.

Many people, like me, prefer to program in assembly language. The extra effort required by doing everything myself is abundantly rewarded by the freedom gained in talking directly to the microprocessor. When I want fast, efficient, compact programs and don't really need floating point numbers and trig functions, I use assembly language. The Apple II computer is a particularly enjoyable system to code because of the many useful routines already coded in the system monitor ROM.

The DOS (disk operating system) is also relatively easy to interface and use from assembly language. DOS receives commands via the monitor I/O routines, making it *almost* language independent. I emphasize the word *almost*, because it isn't enough. Totally independent is even more desirable; and that is the subject of this article.

Imagine this sequence of events. You have recently completed a fast, efficient machine language program. It is a beautiful hi-res epic space adventure. You have designed it to BSAVE the program variables to continue your game at a later date. Being very proud of your latest work, you invite some friends over to play it.

As a friend sits at the keyboard, he is requested to enter his last name, which will be frequently preceded by "Captain" during the game. He types his name, Brunner, and presses return. DOS interprets this as "BRUNner" and the disk starts up. Soon you find yourself in Basic with a conspicuous FILE NOT FOUND message displayed on the screen. After your friend asks if you have been keeping tabs on him, you reload your program and advise him to enter "Sir" this time, instead of his surname.

Several hours of play later, the enemy forces have not accepted defeat. Your friend has to leave and is delighted when you advise him to save his game so he can finish it later. Unfortunately, he isn't very experienced at the keyboard and presses reset instead of return. Back in Basic, you've got more problems. Now anything you type causes a SYNTAX ERR. There is no way to get back to the game in progress!

Close to despair, you bid your friend good night and repower your computer. You remember that, in a Basic program, your friend typed in his surname and the disk remained silent. You try that again, and, to your amazement, the computer accepts anything you type, including DOS commands. Why does DOS work properly with Basic and not with machine language?

Leave Your Forwarding Address. Experiences like this fictitious example happen to many programmers, whether fledglings or veterans. The problem occurs because DOS was designed for the Basic languages. When you leave Basic, you leave the world of DOS-supported languages. Although DOS automatically redirects its communications from one Basic to the other when you do, it doesn't recognize your going into assembly language. So, unless you inform DOS that you're running an assembly language program and actively enlist its error-finding and informing capabilities by leaving your forwarding address, it will continue to leave its messages in Basic's mailbox. Thus, to avoid an operating snafu condition, you must tell the operating system where to find you. DOS keeps track of this information in tables of jump addresses (vectors).

Two vector tables are permanently assigned, one for Applesoft and one for Integer Basic. When one Basic is required, DOS moves that Basic's permanent vector table to the system jump vector area that it uses to interface with that Basic. This table holds five addresses in standard low-order-first format. A sample vector table is shown in figure 1.

The table in figure 1 is in a form that can be entered directly into your assembler to define your entry points for DOS. Of course, you must have actual routines for each label. Some assemblers use the pseudo-op ".WORD" instead of "DA" for the define-address function.

The table is ten bytes long, holding five two-byte addresses, low-order byte first. On a DOS boot, one Basic is loaded or enabled, and its vector table is moved to the system vector area. Your program must move your table into the system vector area while DOS believes Basic is still running.

```
SYSTBL  DA  CHAIN   ; execute with data intact
        DA  RUN     ; clear data then execute
        DA  ERROR   ; error handler routine
        DA  COLD    ; initialize system and clear all
                      data
        DA  WARM    ; reenter system leaving data in-
                      tact
```

Figure 1. System language vector table.

Where is the system vector area? The easiest way to find the answer is to ask DOS. DOS provides the answer at hex location $3D0, which is a JMP instruction supplied by DOS for reentry. Where DOS operates depends on memory size, but it is always aligned on 4K byte boundaries. Knowing this, you can add the JMP address to a fixed offset to determine where to put the table. This article assumes you are using DOS version 3.2. The fixed offset is version dependent and is shown for 3.2. The code in figure 2 will move the table to the correct place for DOS to use it.

```
MOVEIT  LDA  $3D2     ; get high jump address
        STA  $3D      ; high byte of temporary
                        indirect address
        LDA  #$56     ; fixed low-order offset
        STA  $3C      ; low-order indirect ad-
                        dress
        LDY  #$09     ; set index register and
                        counter for 9->0.
MOVLUP  LDA  SYSTBL,Y ; get table byte indexed by
                        Y reg
        STA  ($3C),Y  ; store indirect post—in-
                        dexed by Y reg
        DEY           ; decrement for next byte
        BPL  MOVLUP   ; ten bytes counted 9 to 0
        RTS           ; formally introduced
```

Figure 2. Table move routine.

Coding System Reentry Routines. The complexity of your DOS interface depends on your needs. If you were to write a high-level language such as a Basic interpreter, you would require a complete interface. On the other hand, an arcade game may only need reset recovery. In the simple case, one reentry routine is sufficient for full operation system interface. This routine is the warm-start point of your program. A warm start generally does all initialization except that of data pointers. The warm start for a text editor might initialize the screen and jump to a menu, but it certainly would not reset the text file pointers.

Here is some advice on coding system reentry routines.

Chain and run routines. If DOS knocks here, a Basic program was just successfully loaded. Unless you need to use Basic files as a storage mechanism, DOS should not execute the RUN or CHAIN commands. The BRUNner syndrome illustrates this kind of disaster as allowed by a misinformed operating system.

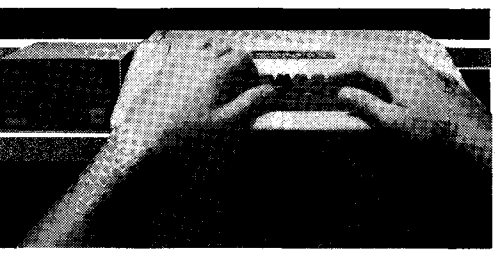
DOS can receive commands from either the input channel or the output channel. If the carriage return (CR) and the end of transmission (EOT, CTRL-D) are received via system output, DOS will interpret the subsequent characters as a possible directive. It is very hard to print accidentally a properly prefixed Basic RUN statement; most ill-advised commands come from computer operators rather than from tested computer programs. DOS will examine the keyboard data when Basic indicates direct execution mode. Typing CATALOG illustrates this. If deferred execution mode is indicated, DOS will not interpret the keyboard. Also, and of more importance, DOS only allows text-file commands when in deferred mode.

Fortunately, changing between direct and deferred operation is simple. Each Basic has its own mode flag. For Integer Basic, the sign bit (most significant bit, or Msb) of location $D9 defines mode. Applesoft uses the high-order byte (most significant byte, or MSB) of the currently executing Basic line number in $76. You need not keep track of which Basic is enabled if you indicate for DOS that both Basics are running at the same time. Figure 3 shows the switch.

GOTO 25

MARKETALK

Reviews



Real-Time Football. The simulation of sports has been almost nonexistent in software for the Apple. What sports programs are on the market are actually arcade games of which the subject is a sport—they have no real relationship to the actual strategy of sports.

Real-Time Football proves that sports strategy simulation is possible, and that the result can be an excellent game. Football fans, take heed; this game is your swig of Gatorade.

Real Time Football is actually three games in one: *Semi-Pro, NFL Professional,* and *Solitaire. NFL Professional* is the ultimate game. Each player generates a team through a draft; each automatically has $2,700,000 to spend, although provision is made to raise or lower that figure depending on your record. Players are drafted by position; each position is shown with high, low, and average salaries, and you get what you pay for.

Each player is shown lists of thirty-six offensive and thirty-one defensive plays, which can be numbered in any order so one player won't be able to tell what the other is doing.

The graphics in *Football* aren't graphics at all, but text-created representations of the football field with down markers and a moving cursor for the ball, the scoreboard, and the lineups. Players are represented by Xs and Os just as they might be in a coach's playbook. Paddle dials move numbers on screen to call out offensive and defensive plays. A thirty-second clock appears before each play. The offense is required to spend five seconds in the huddle; after lining up, the paddles can be used to change the play number—the Xs will follow suit—to represent audibles. Three pushes of the paddle button without a play change executes the play. Meanwhile, the defense can be changing its strategy according to what it perceives the offense is doing; but if the defense is caught changing, it's caught off guard and the offensive play has a greater chance of success. Allowing the thirty-second clock to run out causes penalty for delay of game. Other penalties occur at the computer's throw of a flag, and the opposite team can accept or reject them.

Each team is allowed three timeouts per half; these are also timed. There is even a two-minute warning timeout, but, in this case, players can elect to continue the game without waiting the full timeout period. Game statistics are given at the end of each half.

Is this beginning to sound a lot like a description of real football? If it is, you're getting the picture. This is one heck of a program for the football fan or player. In strategy, alternatives, realism, it doesn't equal the boxed simulation games, it surpasses them. And it moves a whole lot faster—as fast, precisely, as real football.

There are no sound effects in the *Real-Time Football* program. When you hear the sounds of the crowd cheering or groaning it will be you and your opponent—and you *will* hear them!

Semi-Pro is the same football game in a condensed version. Teams are stock (no drafting), as are the plays—eighteen for offense and fourteen for defense. Otherwise, play is the same. The clock in *Semi-Pro* still keeps real time, but you and your opponent choose the length of the quarters; thus you can opt for a short but complete game that you can finish in your lunch hour.

Real-Time Football is by the same company that makes *Sink the Bismarck* and *Ambush,* so it isn't surprising this program smacks of quality, complexity, and care. MCT

Real-Time Football, by Dan Bunte. Strategic Simulations. 48K Applesoft. Disk. $39.95.

Monty Plays Monopoly. You don't play Monty—you play *with* him! Monty is a person, albeit a lo-res person, who plays a darn good game of Monopoly with you and as many others as the Parker Brothers game will allow. He buys or rejects real estate, bids on unbought lots, builds houses and hotels, wheels and deals when he can, and will try to take you to the cleaners when you offer a deal to him.

In personality, Monty's the ideal opponent. He's sad when he's sent to jail, gloating when he lands Boardwalk, ruthless when he gets a monopoly, and philosophical when he goes bankrupt.

The only change Monty requires in your regular game is that you roll computer dice instead of regular ones most of the time. This is how Monty keeps up with what's going on. In exchange, he keeps track of whose turn it is, and won't let anyone forget to take salary.

Either paddle button keeps the game rolling, and Monty provides a musical identification of each board location. Recalling the name of each song is a fun game in itself. Monty keeps track of his own money; and, while he never forgets to collect a rent, he never forgets to pay one either.

Monty is a godsend for single Monopoly freaks, a boon to two people who want to avoid the onesidedness of a two-player game, and a great addition to a group.

Monty Plays Monopoly is well worth the price as it brightens and enlivens what remains by far the most popular board game ever. MCT

Monty Plays Monopoly. Ritam, $29.00; 16K, Apple II/II Plus.

APPLE HELPS THE EMPIRE STRIKE BACK

BY RICHARD KNUDSEN

Not long ago, in a studio not far away, a microcomputer helped make a megabucks movie.

The movie was *The Empire Strikes Back*. The microcomputer was the Apple. The location was the home of Industrial Light and Magic, George Lucas's special effects facility. ILM sits just outside San Rafael, in Marin County, California. The company was charged with responsibility for all the special effects on both *Empire* and *Star Wars*.

One difference in ILM's procedures from *Star Wars* to *Empire* was the harnessing of the Apple to improve accuracy and relieve tedium in the process of tracking more than eight thousand pieces of film that went into the special photographic effects of *Empire*.

The scope of Apple's task was barely less than the scope of the task assumed by George Lucas when he set out to make a sequel to *Star Wars*.

Sequels = Box-Office Poison. It is conventional wisdom in Hollywood that sequels are almost surely doomed to failure—financially if not artistically. The formula used in the film industry is that a sequel can hope to do no more than 40 percent of the box office of the original film. Few films can afford that kind of public disinterest and still be qualified as successful.

One need look no further than the sequel to *The Exorcist* or the prequel to *Butch Cassidy and the Sundance Kid* to understand Hollywood's thinking. Both dropped out of sight faster than Spiro Agnew.

Lucas, however, is an uncompromising taskmaster to himself; and he has sworn that he'll begin no trilogy that he won't complete. For the millions who have flocked to *Star Wars* and *Empire*, that's good news, because they can anticipate *Return of the Jedi*, which will complete the middle trilogy of the nine-part *Star Wars* epic.

A Measure of Success. But Lucas's promise meant that he saddled himself with the unenviable task of trying to outdo *Star Wars*, the most successful movie of all time. The success of *Star Wars* is almost incomprehensible. It has approximately doubled the box-office success of the previous topper, *Jaws*. It took in ten times the receipts of the average successful movie and five times the receipts of the average box-office hit. Its $200 million in box-office rentals—which is the amount of money returned to the studio, not the amount of money taken in at the box office—represents a pinnacle of success roughly comparable to hitting a hundred home runs in one season or finding a cure for the common cold.

So, along with the critical acclaim and financial rewards, what else did *Star Wars* reap for Lucas? It won him the monumental challenge of making the sequel better.

Incredibly Believable. Further complicating that task was the nature of Lucas's films. They depend heavily on the success of special effects to promote what critics are currently fond of calling "the suspension of disbelief," which is a fancy way of describing a movie's ability to engross the audience and hold it captive.

Poor special effects are the fastest way to lose credibility on the screen. Spaceships that look like Revell plastic models, jerky motion, shadows around composites—any number of pitfalls await the filmmaker who stakes his success on special effects. And anything other than 100 percent believability in execution of the special effects usually means limited success or total failure for the movie.

ILM delivered for Lucas on *Star Wars*, but could it do so again, without being trite and repetitive, on *Empire*? Four months after its opening, the verdict on *Empire* is favorable. Within a few days of your receiving *Softalk*, *Empire* will become the second most successful film of all time and will be setting out in pursuit of its predecessor.

An Unsung Star. And an Apple was there. ILM prepared 407 special effects shots for *Empire*. These shots were composed from 835 color elements and 7,534 black-and-white elements that had been prepared. Just keeping track of all these elements was a herculean task, and ILM put an Apple to work.

But the Apple did far more. As explained by Bruce Nichol-

APPLE HELPS THE EMPIRE STRIKE BACK

son, optical photography supervisor, "Take a simple shot of three spaceships. Each of those ships is photographed separately. We photograph the space background separately, and, if there's a planet in the frame, it's also shot individually. That makes five pieces of film. Many of our shots had more than a dozen pieces of film.

"Each piece of film must be frame-counted and sync-marked. The film must be separated into three black-and-white positives, representing the three primary colors. It's very tedious to explain to the operator the start point and the end point on each of these pieces of film.

"David Berry and Jim Veilleux programmed the Apple so that when a piece of film came in, we could enter a series of code numbers. Then, when the positive separations were made, the projector operator could just call up that piece of film on the Apple and the computer would indicate the start frame that was applicable."

That same data was needed later in the production process when the composites, representing the final shot, were made. The Apple was programmed to indicate exactly which elements were needed for the final shot and onto which frames they were to be printed.

In addition, ILM filmed its effects in VistaVision, which has fewer frames per foot than other formats in standard use. Apple was used to calculate the conversions from footage to frame counts.

Apple's Moving Up with Jedi Knights. "In the past," explains Nicholson, "we had to figure out the frame counts by hand and enter them onto count sheets. The hand addition and manual entry onto the count sheets made this a laborious pro-

Meet the Man Who Mastered the Apple To Conquer

Alan Gornick photo

BY ALLAN TOMMERVIK

No architect has begun a career by designing an edifice more stunning than the Eiffel Tower. No sculptor has contrived to outdo *The Thinker* in a maiden effort. No composer has exceeded the beauty of *The Messiah* in a first draft. No engineer has surpassed the grandeur of the Grand Coulee Dam in his first project.

Likewise, no one would expect a rank amateur to create a sophisticated number-crunching program with a multitude of variables and adapt a complete microcomputer business system within six months of his first hands-on experience. Alan Gornick did, and his experience is not only illustrative of the potential of the Apple in the hands of a novice, but also of the interdependence of hardware and software vendors.

His Work's All Wet. By trade, Gornick is a cinematographer. His specialty is underwater photography, although he's equally adept with a camera on terra firma. His most recent release is *The Black Stallion*, for which he did the underwater sequences. The film caused quite a stir in Hollywood when it was not nominated for an Oscar for cinematography.

He also shot sequences for *Gray Lady Down, Foes, Stunt Rock,* and *The Towering Inferno*. Films soon to be released to which he contributed underwater sequences include *Oceanic Opera* and *Virus,* the twenty-four-million-dollar Japanese-financed epic. He's scheduled to shoot parts of *Never Cry Wolf* for Disney Studios.

Gornick has also been director of photography on segments of such television series as "Charlie's Angels," "Fantasy Island," "S.W.A.T.," and "Barnaby Jones."

The Business of Art. By its very nature, cinematography is ruled by the laws of physics, and successful cinematographers

a Contract in Three Months

are those who clearly grasp those laws. But the general Hollywood consensus is that the director of photography is also the artist who paints on film what the writer and director have conceived and captures what the actors have wrought.

There's also a business side to cinematography, and it was this aspect of his work that led Gornick to Apple.

Late last year, he took a crew of seven on a twelve-day shoot for *Oceanic Opera* at Truk Lagoon in Micronesia. Even in Hollywood's overinflated jargon, those six thousand miles from home qualify Truk as a distant location.

So cut off were Gornick and his crew that their paperwork, including payroll, couldn't be handled until their return to the States. And so complicated are the union contracts governing the film industry that it took Gornick three days to complete his payroll computations and paperwork when he returned home.

That expenditure of time made him seek out a better way.

His search led him to microcomputers and convinced him that the languages available to him through the Apple computer were most compatible with his projected uses. Perhaps even more important to him was the modular nature of the Apple, which provides for expansion of the system as needed. In January 1980, he made his initial purchase.

Today, his system consists of an Apple II Plus with a 48K memory, an Integer card, three Apple disk drives, an Integral Data Systems 440 printer, and a D. C. Hayes micromodem.

Goliath Was a Contract. But, in January, what Gornick had was an Apple computer that he didn't know how to program and perhaps the most complex labor contract in the country that he wanted programmed. The Hollywood basic labor agreement, negotiated by the International Alliance of Theatrical and Stage Employees, covers every craft and job that goes into the behind-the-scenes production of a movie. That in itself would make the contract more detailed than most.

GOTO 22

An Apple II poses with David Berry, optical printer operator at ILM. Berry contributed to programming the Apple to handle several tasks previously done manually. With Jim Veilleux, he developed the software that enabled the Apple to generate optical composite count sheets. Count sheets track the thousands of individual pieces of film that go into optical effects composites, listing the shot to which they belong and in which frames of the shot they are to be inserted. Apple was additionally used to compute the frame-to-footage conversion necessitated by ILM's use of VistaVision, which has fewer frames per foot than standard formats.

EMPIRE

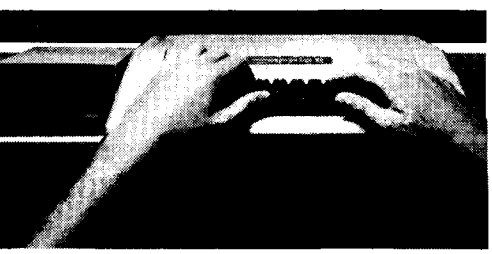
ess. With the Apple, we achieved tremendous savings in time and in the accuracy of the computations."

Nicholson has high praise for the Apple. "It saved us time and we'll continue to use it. We're now looking into applying it in other areas."

ILM has little time to bask in the glory of their *Empire* accomplishment. *Return of the Jedi* beckons, probably for 1983 completion. And it figures that the Apple will play an even bigger role then in smoothing the production path for the special effects wizards at ILM.

MARKETALK

News



☐ The **Software Publishing Company** (Palo Alto, California) debuts with the *Personal Filing System* (PFS). PFS is a free-form data base program with many of the datagramming capabilities expounded by Apple software vice-president John Couch. *Softalk* will present a full review of PFS in October. The reviewer's initial reaction is that PFS has the right combination of ease-of-use and power to help both the nonprogrammer and the programmer. Requires 48K, 16-sector disk-based Apple system. $80.00.

☐ **Southwestern Data System's** (Santee, California) newest offering is *The Correspondent.* Primarily a word processor, with upper and lower case, 80-column capacity, and four-way scrolling, it also serves as a data base and a programming utility. Capabilities include tab, justify, center. Permits editing of random or sequential text files, even exec files. Enables user to put bidirectional scrolling in own programs. Requires 32K with Applesoft in ROM, 48K otherwise; disk only. $34.95.

☐ **Advanced Business Technology Inc.**, Sarasota, California, is releasing this month the BarWand, a product that puts the Apple in the middle of the coming revolution at retail checkout counters.

The product harnesses the Hewlett-Packard HEDS-3000 bar-code data-entry wand to the Apple. The BarWand will read most retail product bar codes, paper byte codes, and ABT's own label codes. A bonus package of ABT software permits the printing of label codes with the IDS 4400 Paper Tiger printer.

With the BarWand and label code software, low-cost bar-code systems can be implemented for use at the point of sale, for inventory, for production control, for security identification, and for other purposes.

Available this month, the BarWand sells for $195.00.

☐ **Syntauri Ltd.**'s (Palo Alto, CA) *alphaSyntauri* digital musical instrument is a standard sixty-one note organ keyboard that inputs directly to the Apple and outputs six voices at once through a sound synthesizer board and user's stereo system. User controls envelope, octave range, note percussion, vibrato, and sustain. System simultaneously displays real-time color display of note position; alternate display for training and analysis gives note names as keys are played. Instrument-generated music can be saved and played back at variable speeds without change of pitch. Requires 48K Apple II/II Plus, disk drive, color CRT, and two Apple Music Synthesizer boards. $1295.00.

☐ **Meca** (Yucca Valley, CA) has developed a tape drive for the Apple II that behaves much like a disk. Called Tape II, the system offers random access to 500K of data that can be RUN by name and CATALOGed. The tape drive boots through a card via PR#n and responds to BLOAD, BSAVE, DELETE, and VERIFY; it changes languages upon INT and FP commands. Nor are files ignored. Tape II will OPEN, CLOSE, READ from, and WRITE to data files on tape. Normally LOADing seven times faster than a cassette recorder, Tape II allows 500,000 bytes of on-line storage with one tape drive. $499. Back-up drives cost $310.

☐ *Softalk* welcomes an old board-game favorite, **Avalon Hill**, into the world of Apple. Through its new subsidiary, Microcomputer Games, Inc., Avalon Hill announces five wargames for the Apple II for fall release. All are cassette only, require Applesoft plus 16K, and will retail for $15.

☐ **Muse** (Baltimore, MD) announces a form-letter generator module for their Super-Text word processor. It's scheduled for release in October. Muse's *Voice*, which was due to arrive this summer, is capable of a vocabulary of eighty words or phrases per data disk in high quality sound. Access is through the cassette recorder and it's easy to edit. Transfers with normal PRINT commands into Basic programs. Requires 48K Apple II/II Plus. $39.95.

☐ **Novation's** (Tarzana, CA) new D-CAT direct modem is intended as a complement to that company's popular CAT acoustic modem. The D-CAT's slick lines fit easily under a telephone and, according to Novation, it can be installed in seconds. Power pack plugs directly into wall socket. Modem offers special hold function and complete self-test. $199.00.

☐ **Artsci** (North Hollywood, CA) recently introduced the *Magic Window*, a word processor. The *Magic Window* allows the user to type as on a typewriter, able to see exactly where what's being typed will fall on the printed page. This feature is especially useful for forms or tabular matter and is helpful in letter formatting. Editing is handled entirely by control characters, which allow the user to delete letters, to delete and recover lines, to insert spaces, and to cut and paste to achieve the best possible layout. *Magic Window* comes in a looseleaf binder with a wall card showing the Apple keyboard marked for the editing control characters. Also included is a Webster's word-break dictionary. Requires 48K and disk. $99.95.

Armed with a basic understanding of the internal structure of program lines in Integer Basic, you can perform a number of feats in your programs that are, according to the book, illegal. A type of program that often benefits from illegal commands is the demonstration program: one that incorporates a game or display with a user's option to examine various parts of the program or that calls forth a program line for modification by the user. Also, only through illegal commands can you search Apple's memory for all instances of a partcular variable in your program and LIST the lines containing those occurrences.

Both of these applications require an indirect LIST command, and this is the command I'll focus on; several others are possible (see box), and the principle is the same.

Integer Basic has one capability that the generally more powerful Applesoft lacks: the use of the indirect GOTO, such as

```
100 GOTO LINE
```

where LINE is a variable set to a line number. This command is called indirect because the line number is acquired from a variable. The direct version of the same command might be

```
100 GOTO 75
```

But, even in Integer Basic, the indirect form is limited. For example, within a program you can say

```
1000 LIST 75
```

or

```
1000 LIST 75,100
```

but you can never include indirect LIST statements, such as

```
1000 LIST LINE
```

or

```
1000 LIST L1,L2
```

where L1 and L2 express a range of line numbers you want your program to LIST to the display screen. If you try to type these lines into your Integer Basic program, Apple's Basic interpreter objects strenuously with beeps, raspberries, and SYNTAX ERRs. It's illegal, says your Apple; you can't do it.

Or can you?

What if you could input such statements without Integer Basic realizing it? In other words, what if you could *con* your Apple into thinking you were using perfectly legal commands?

Sleight of Mind. It's the old now-you-see-it, now-you-don't routine, and, if you're willing to go to a little extra trouble for the setup, you can con your Apple into allowing such functions in your Integer Basic program.

Before you start looking in your mirror and seeing Henry Gondorff or Christine Vole,* you need to know some elementary facts about the internal structure of Basic program lines as they're represented in memory. It's assumed that you know what the hexadecimal number system is and that you understand at least something about how to use monitor commands to inspect and modify memory. (For more detailed discussion of the representation of Basic program lines in memory, see Sandy Greenfarb, "Internal Structure of Integer Basic," *Call A.P.P.L.E.*, November-December 1979; and Frank Kirschner, "The Integer Basic Token System in the Apple II," *Micro*, May 1979.)

A Basic program line is represented in Apple's memory as a series of bytes, or addressable locations. The hexadecimal values contained in these bytes define at least three types of data: numeric constants, ASCII character codes, and tokens that represent Basic keywords.

Casing the Joint. Examining the way Apple holds a legal program line should make this clear:

```
95F6—0A  E9  03  62  CC  B1  49  CC
          B2  01
```

Here is the same sequence, broken into its components and labeled:

```
0A     /E9 03 /62      /CC B1/49/CC B2/01
Length/Line #/PRINT/L1    /, /L2     /EOL
```

The first hex value, 0A, is the length in bytes of the entire line. A count of the number of hex values in the list will yield the decimal number 10, which is equivalent to 0A in hex.

The next two bytes, E8 and 03, contain the hex equivalent of the line number, which is decimal 1000. Note that 03E8 is the full four-digit hex equivalent of decimal 1000, and, in storing this value in two consecutive bytes, the low-order byte comes first. That is, the rightmost two hex digits come before the leftmost two hex digits. This is the normal way of representing integer numbers: low byte followed by high-order byte.

The fourth byte, 62, is the hex token for the keyword PRINT in Basic. All Basic's keywords—such as DIM, GOTO, FOR, NEXT—are represented by a single byte token. Other syntactical components of the language—such as punctuation signs, arithmetical signs, and statement separators—are also represented by tokens.

The fifth and sixth bytes, CC and B1, are the ASCII characters L and 1 (which together comprise the variable name, L1). The seventh byte, 49, is the token for a comma contained in a PRINT statement (other types of statements containing commas usually have other token values for the comma). The eighth and ninth bytes, CC and B2, are the ASCII characters L and 2 (the variable name L2). And, finally, the tenth byte, 01, is the end-of-statement token.

The Setup. If you've any cunning in you, your devious mind is beginning to suspect how you might go about putting

```
1000 LIST L1,L2
```

into an Integer Basic program without Apple's noticing. This line differs from the sample legal line only in the substitution of the command LIST for the command PRINT. To make this substitution in a program, you need only modify the memory location containing the PRINT token to contain the token for the LIST statement.

Here's how the Basic con job works.

1. Boldly enter the target statement first—to make finding it in memory easier:

```
1000 PRINT L1,L2
```

2. Slip stealthily into the monitor by typing:

```
CALL—151
```

*Can you fully identify these two conpeople? *Softalk* will award a disk of Stoneware's *Bloody Murder* by Arthur Wells to the reader whose correct identification has the earliest postmark (through drawing in case of tie), and another to the lucky slowpoke whose answer is drawn from all other entries received prior to October 31. Send your ID's to Softalk Conpeople, 10432 Burbank Boulevard, North Hollywood, CA 91601. Include the name of your regular computer store; in case you win, you'll be able to pick up your prize from your retailer.

3. Find the innocent statement in memory. The beginning of a Basic program—in this case, line 1000—resides at a spot whose address is recorded in hex locations in $CA and $CB. Read the address from these locations by typing the monitor command:

CA.CB

Monitor will answer by displaying a line just like or very similar to:

00CA— F6 95

This address, expressed in two bytes, shows the low byte followed by the high byte. The value of the high byte depends on the amount of memory in your system and on whether DOS is resident; the value shown is for a 48K DOS system. Now you know that the memory address of line 1000 is $95F6.

4. You know that line 1000 in its entirety is represented in 10 bytes. To display all 10 bytes, type

95F6.95FF

The result is the same sequence of bytes used to illustrate how a program line sits in memory.

The Switch. 5. With this display, you're on the right street; you need only count over to find the home of the PRINT statement, 62. In the example, it's $95F9. Now comes the real scam. Because you're fooling around in the system monitor, Integer Basic won't have noticed you're there. So you warm up your hands, and sneakily oust the PRINT statement from its home by installing the token for LIST—which has the hex value $74. Use the monitor command

95F9:74

to modify memory location $95F9 from PRINT to LIST.

But don't do your quick disappearing act yet!

6. The token for a comma contained in a PRINT statement is not the same as that for a comma in a LIST statement. The comma resides on the same street at a different spot; look for the PRINT comma's token, 62, between the tokens for L1 and L2. That's location $95FC. Make the quick switch with the monitor command:

95FC:75

because 75 is the token for a comma contained in a LIST statement.

7. Now you can make that getaway—only you don't need to. Your Integer Basic program is still oblivious to your presence. With a coolness worthy of Professor Harold Hill, you can return to your Basic program via the CTRL-C sequence, type in LIST 1000, and see your indirect LIST statement appear without a single protest from Integer Basic!

CON MAN LOADs NEW AUTO, RUNs CLeaR to DELaware to SAVE skin . . .

This may not be much of a news item as crimes go, but it contains every "crime," other than LIST, HIMEM:, and LOMEM:, possible in Integer Basic deferred mode.

Why are these commands illegal in deferred mode? They aren't allowed in deferred operation for the same reason that multiple statement lines aren't allowed in direct operation. They alter the 6502 registers that keep track of the next statement to be executed in the line. Ergo, when you con your Apple into accepting one of these commands, be sure it's the only statement on the line.

More formally, the illegally usable commands and their tokens are shown in figure 1.

LOAD	$04	Tape LOAD
SAVE	$05	Tape SAVE
CON	$06	
RUN	$07	Start at lowest line
RUN	$08	Start at specific line number
DEL	$09	Delete a line or range of lines
NEW	$0B	
CLR	$0C	
AUTO	$0D	
MAN	$0F	
HIMEM:	$10	
LOMEM:	$11	

Figure 1. Illegal deferred mode commands.

Conquering a Contract

from page 17

Further complicating the provisions of the contract is the fact that filmmaking is not a nine-to-five affair. An individual setup may take eighteen hours to shoot with an early call for the next morning. The IATSE contract dictates down to the last nanosecond what protections, penalties, and pay obtain during these long shooting days.

Soft Ammo Beat Big Guns. Whole batteries of attorneys and union business agents make their livings monitoring and debating the provisions of this contract. And here was one optimistic new Apple owner who proposed to program it all on his microcomputer. And did.

Gornick fortified himself for the fray with the provided Apple documentation; *Basic Primer*, written by Mitch Waite and Michael Pardee; and the two-disk Applesoft tutorial, *Step by Step*, by Program Design Incorporated.

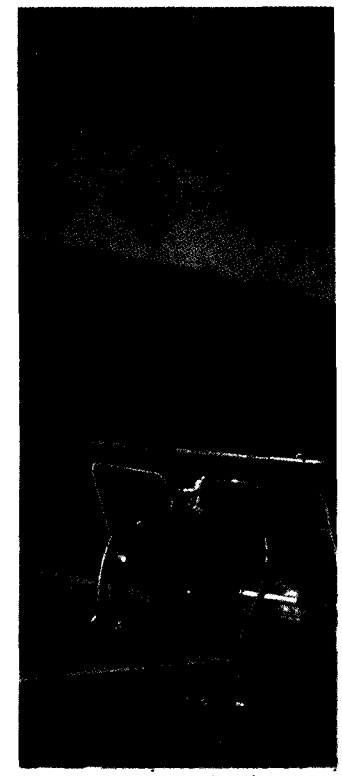
So armed, he began programming modules for what would become his *IA Day Pay* program. When contract terms stumped him, he turned to Gerald Smith, business agent of his local, or Joseph Bernay, international vice-president of the IATSE, to get a definitive reading on meaning.

The final result, accomplished essentially in two man-weeks of effort spread over a three-month period, is a program that will do in one hour what had taken Gornick three days to accomplish. Gornick found *Apple-Doc* from Southwestern Data Systems an invaluable tool for modifying variables during his creative stage and used Programma's *Applesoft Optimizer* to pack the program once it was complete.

Mainframes Will Catch Up "Soon." A final modification to the program allows him to check his actual labor expenditures against his projected expenditures daily. It is interesting to note that a computer service bureau, using a large mainframe computer, has just announced development of such a system for Hollywood studios and projects its completion "within a year."

Having leveled the Everest he set out to conquer, Gornick sought other challenges. He adapted Personal Software's *VisiCalc* to his bidding requirements. Producers, be they for movies, television shows, or commercials, take an intense interest in what a shoot will cost them.

By presetting *VisiCalc* with all possible categories, Gornick is able merely to plug in the requirements for a particular job and have an answer in minutes. And if an alternative to cut expenses is proposed by the producer, Gornick can integrate that and have the new result in seconds. His response time to a request for cost estimates is several hours faster than any of the major studios can manage with their battalions of accountants and mainframe computers.



Cinematographer Alan Gornick and his wife Martha provision the good ship **Enmare** preparatory to an underwater filmmaking session. Gornick is unusual among cameramen in that he owns all his own equipment, rather than renting the requisite equipment for each shoot. In the right foreground is the water-tight housing that protects his camera underwater.

The Art of Business. Then Gornick realized he could use his Apple to drum up business. Directors of photography function essentially as independent contractors, even though their working environment is rigidly controlled by the union contract, and therefore find it necessary to keep reminding production managers and directors of their availability.

Gornick implemented *Apple Post* for this task, entering the pertinent names and addresses with codes and sorting according to the codes of those he wanted to receive a particular promotional piece.

Apple Post is also used to store the names and addresses of his crew members.

But names and addresses aren't the only data Gornick needs to know about potential crew members, once a shoot has been proposed. He also needs information on job classifications, union memberships, salaries, and phone numbers, as well as references to special skills any potential member of the crew may have.

For this task, he adapted Synergistic's *Modifiable Data Base.*

Apple Cuts Red Tape. Gornick also uses the data base to store his inventory of equipment. He owns thousands of pieces of camera equipment, lists of which must be fully documented for insurance purposes and for customs if the shoot is outside the United States.

Pertinent information is make and model, country of origin, value, weight, and serial number. Before Apple, this information had to be prepared by sorting, by hand, through eight lists and spending four to five hours typing the forms. Now it only takes a few keystrokes and twenty minutes.

Gornick's next modification to this program will be to use the data base program to assign a shipping case to each piece of equipment and have the program calculate the exact weight and value of equipment in each shipping case.

Credits Where Credits' Due. Gornick is currently adapting Information Unlimited's *Easy Writer* to the construction of his credits list. Gornick has always felt strongly that his most recent credit should read first on his resume. But that always necessitated retyping and bearing new reproduction costs every few months for copies that became outmoded before they were all distributed.

With *Easy Writer,* he can not only insert his most recent credits at the front, but he can select only those credits interesting to a particular producer. A commercial producer need no longer wade through Gornick's feature credits nor a feature producer through his commercial background.

Alan Gornick photo

What about the D. C. Hayes micromodem? Gornick bought it in anticipation of linking to a computer that provides international weather reports. Weather is the vital variable in cinematography. Often a cinematographer will be in Los Angeles Monday, but scheduled to shoot on distant location Wednesday. The micromodem should give Gornick the means to gather better weather data on which to base his preparations than he has access to currently.

He also plans eventually to use the micromodem to access airline schedules and make travel arrangements for his crews.

How Stars Are Born. For someone who didn't know PRINT from RUN in January, Gornick's adaptation of the Apple to cinematography is a story that parallels Apple's own success.

Apple and Gornick both earn kudos on that account. But it's equally interesting to note the important contributions made by other microcomputer product companies, from publishers through peripheral manufacturers to software vendors. Without the support of their products, Gornick might still be struggling with his first application.

Instead, we may be looking to the time when Alan Gornick will be announcing that his Apple has supplanted him as director of photography.

How Hard Was It?

The complexity of the *IA Day Pay* package can best be understood by following Gornick's program through a run.

The menu first calls up the four basic shooting conditions recognized in the contract: studio, studio zone, nearby location, and distant location. Each of these may have as many as seven subdivisions.

Once the subdivision has been chosen, the time card data is entered. This consists of job category, pay rate, start time, meal break times, and end time.

The program then computes straight-time earnings, overtime earnings, golden-pay earnings (overtime earnings at higher than one-and-one-half times base pay), 10 percent night differential, 20 percent night differential, meal penalties to be paid if applicable, and contributions to health and welfare plans. It also provides for hazard pay and equipment damage allowances.

That may seem straightforward enough until you understand that the program works for all members of the crew on the set and that the basic contract provides differently for each job classification. As an example, some personnel are hired on an eight-hour guaranteed workday basis and others on the basis of twelve hours guaranteed as the normal workday.

Start with the four basic conditions recognized by the contract. Figure an average of five subdivisions for each. Assume nine compensation categories. Then, for easy figuring, assume the program to handle only ten job classifications.

So, a minimum of eighteen hundred variables faced programming novice Alan Gornick when he set out to tame the IATSE contract.

What in the World Is Datagramming?

John Couch believes the software revolution has yet to begin, although he concedes *VisiCalc* as a pioneering step in the right direction. He calls the revolutionary tool *datagramming*, and this is his explanation of the difference between it and programming:

"Traditionally, system software has been considered to comprise the tools for the application programmer, and the application software the solution to the programmer's problems. If you study an application program, you'll find at least four basic components: computation capability, data input and verification, data output (reports or graphics), and a set of data structures for storing and moving the data around within the application.

"Traditional procedure-oriented languages have tried to address each of these functions for the application programmer. Solving a user's problem via a procedure-oriented language is called *programming*.

"Datagramming is solving a problem without using a procedure-oriented language. Instead, datagramming employs a set of interactive tools that are specification-oriented, thereby allowing the nonprogrammer to specify his data input, the appropriate verification checks, his output, the form it should take, and the calculation on this data."

Couch believes that traditional procedure-oriented languages will continue to flourish for the large programming industry, but that the real market for the personal computer industry is the nonprogrammer. To reach this market, specification-oriented software is a must, and the eighties will breed a whole new phenomenon: datagrammers.

Apple Profit Center Profits Apple Users

from page 7

provides utilities, information, licensing agreements, and protection techniques to the outside software houses.

Self-Documenting Disks. Couch's publications group has already set new standards in the industry and intends to maintain its leadership position with a bold plan that will change the thrust of program documentation.

"Past documentation has always been passive, but we're shooting for interactive manuals," explains Couch. Interactive manuals allow the user to participate actively in the learning process by requiring the user to respond. They are interactive teaching programs supplied on a disk, programmed to provide necessary reference information for experienced users and more detailed introductory material for beginners.

Planned for marketing next year, the interactive manual is a concept that will stave off many of the frustrations felt by users who must wade through multipaged manuals before loading a program, when all they really want to know is the aging of their accounts payable.

But the main thrust of the group is new software development. Couch's new software products will emphasize "the integration of tools. The user should be able to carry his data from one of our programs to the next, so he gets maximum use."

Softalk photo

The Apple Software Lab's exec team: Bruce Daniels, manager of systems software for advanced systems; John Couch, vice-president; Jack MacDonald, manager of systems software for the Apple II and ///, and Susan Wells, Couch's executive assistant.

Programs tentatively scheduled for release yet this year for the Apple II and Apple II Plus are *Appleplot*, *Fortran*, and *Pilot*, a *Dow-Jones News and Quotes* reporter, a *Symbolic Debugger*, *Apple Music Theory*, and *Education II* and *III*. For the Apple /// will arrive expanded versions of *Business Basic*, *Fortran*, *Pascal*, *VisiCalc* ///, and a mailing list.

As exciting as these programs are, expectations are even higher down the road. Couch expects the next year to be the most fruitful and exciting the software lab has experienced.

Pascal Programs for All. Apple has made the commitment to using Pascal as its fundamental programming language of the future. For those who might suspect this to be just an underhanded means of selling prior Apple owners an extra language card, Couch is quick to explain, "We've developed a Pascal run-time program that permits most of our software programmed in Pascal to run on earlier Apples without the Pascal card. I would like to make that utility available to software vendors for inclusion with their Pascal applications."

In March 1979, Apple's executive staff pinpointed software as the weakness in the company and set Couch on the path to rectify that problem. A 4000 percent boost in sales reflects his success, but the implementation of the datagramming concept during the next year will represent the culmination of Couch's early efforts.

DOS

from page 11

```
INT       EQU  $D9   ; Integer run flag (Msb)
FP        EQU  $76   ; Applesoft line number (MSB)
DIRECT    LSR  INT   ; plus status
          LDA  #$FF
          STA  FP    ; big line number (too big!)
          RTS
DEFERRED  LSR  FP    ; smaller line number
          SEC
          ROR  INT   ; minus status
          RTS
```

Figure 3. Direct/deferred mode selection.

Error Reentry Routines. DOS reenters here if a disk error has occurred. Before reentry, an appropriate error message is printed to the current output device. Control is then transferred to your error handler with the system error number in the X register. The possible causes for each type of DOS error are fully treated in the DOS 3.2 reference manual; check there for further information. Figure 4 comprises a summary of errors by system error number.

Err#	Error Message	Most Common Cause
1	LANGUAGE NOT AVAILABLE	Applesoft not on diskette
2,3	RANGE ERROR	Command parameter too large
4	WRITE PROTECT	Write to write-protected diskette
5	END OF DATA	Reading past end of text file
6	FILE NOT FOUND	File name misspelled or not on disk
7	VOLUME MISMATCH	Diskette changed by user!
8	I/O ERROR	Door open, wrong drive, bad disk
9	DISK FULL	Disk full!
10	FILE LOCKED	Write to locked file
11	SYNTAX ERROR	Improper command syntax
12	NO BUFFERS AVAILABLE	Too many text files open
13	FILE TYPE MISMATCH	Command improper for file type
14	PROGRAM TOO LARGE	HIMEM or LOMEM wrong, not enough RAM
15	NOT DIRECT COMMAND	Text-file command in direct mode

Figure 4. Disk error summary.

A truly complete and bulletproof error routine might use the system error number in the X register as an index for an ON system err# GOTO (CASE) type construct. Every error type could have a corresponding routine to handle that specific error. Two things are needed to implement this type of construct. The first is a list of addresses (vectors) for each error condition. Such a vector list would be similar to our system table (figure 5), but would have more vectors.

DOS

```
ERRVEC DA NOTAVL  ; language not available
       DA RNGERR  ; range error
       DA RNGERR  ; range error
       DA WRPROT  ; Write-protected diskette
       DA ENDFIL  ; End of data
       DA NOFILE  ; File not found
       DA BADVOL  ; Volume mismatch
       DA IOERR   ; I/O Error
       DA DSKFUL  ; Disk full
       DA FILOCK  ; File locked
       DA SYNTAX  ; Syntax error
       DA NOBUFF  ; No buffers available
       DA BADTYP  ; File type mismatch
       DA TOOBIG  ; Program too large
       DA NOTDIR  ; Not direct command
```

Figure 5. Error vector table.

The coding of each routine is up to you. To simplify the task, it is generally easy to use the same routine to handle many types of errors.

This brings us to the second requirement. To execute a routine from the list, you need an ON ERR# GOTO routine. In figure 6, three bytes were used to test for the zero condition. How can it be done more directly in two bytes? As coded, the routine in figure 6 is acceptable, but it isn't optimal. It requires an indirect jump vector to be reserved in RAM. You can do without that vector and make the routine smaller by performing a JMP (stack pointer) via the RTS instruction. How would you code it?

```
ERROR   DEX             ; Error #s start at 1
        BMI HUH?        ; Error # < one?
        CPX #15         ; Error # >= 16?
        BCS HUH?        ; Yes ... we can't handle that!
        TXA             ; (Error number-1) to accum
        ASL             ; Times two for index (2-byte address)
        TAX             ; Index to index register
        LDA ERRVEC,X    ; Get low-order address
        STA INDADR      ; Store for jmp (indirect)
        LDA ERRVEC+1,X  ; Get high-order address
        STA INDADR+1    ; Store high byte for jump
        JMP (INDADR)    ; Go for it!
INDADR  DS  2           ; Reserve two bytes for indirect address
HUH?    LDA $3F4        ; If you get to this point
        EOR #$FF        ; something is very
        STA $3F4        ; wrong. DOS has
        JMP ($FFFC)     ; passed you a non-
                        ; existing error number!
                        ; Maybe you have
                        ; crashed it or are using
                        ; something other
                        ; than DOS 3.2. I sug-
                        ; gest you abort and
                        ; reboot DOS 3.2
```

Figure 6. ON ERR# GOTO error routine.

Cold-Start Vector. The cold start is used by DOS to initialize the system language for the first time. This results from one of several conditions. The INT or FP command will cold start Basic, as will the RUN command specifying the Basic that is not currently operating. The system vector area is overwritten by DOS prior to transferring control via the cold-entry vector. As a result, your machine language program should never receive control via the cold-start vector. Restarting DOS via $3D3 will cause a cold start; this is the recommended way to exit your system. It ensures that the resident Basic will be initialized and that the operating system will be ready to accept the next task.

Warm Reentry Routine. DOS comes here in response to a $3D0 restart or reset when using an autostart monitor ROM. Since reset will affect all peripheral hardware and set text mode, some initialization will probably be required. Reset can happen anytime; when it has happened, you cannot assume your data is valid. If your application demands data integrity, your data structure should include an error detection or correction mechanism. Keeping checksums is one easy, but not foolproof, way to detect errors caused by reset-interrupted system operation. Chances are, your program won't be controlling a nuclear reactor; so jumping to a system menu would be an appropriate function for warm-start entry.

Correctly interfacing your machine level program to the operating system is an important part of "finishing" the job. Ironically, you've done a good job if the user feels as though he never left Basic. The methods outlined here will help DOS 3.2 users achieve that goal.

TRADETALK

from page 5

☐ *Computers and Mathematics,* written by **James L. Poirot** and **David N. Groves,** has been adopted for statewide use in Texas school systems. The text, published by Sterling Swift Publishing Company, is accompanied by a hand-on workbook consisting of four cassettes, or two diskettes, that teach Basic on the Apple II.

Softalk photo

Dana Perkins

☐ **D. C. Hayes** has appointed **Dana Perkins** to the newly created position of marketing manager. Perkins, who will be responsible for advertising and marketing strategies, was formerly advertising manager for Digital Communications Associates.

☐ **Jean-Louis Cleenewerck** and **Guy Boisserie**, general directors of Sivea S.A., Paris, France, the largest computer store in that country, made a U.S. purchasing tour in August.

The executives were buying for their microcomputer store, located at 31 Boulevarde des Batignolles, 75008 Paris, and were negotiating distributorships for some U.S. product for European territories.

Cleenewerck said Apple was outpacing all microcomputers in sales.

☐ Newspapers and television do a more than adequate job of reporting on disasters, crises, and other hair-rising items. But they seldom find time to cope with good news. Tradetalk is intended to fill part of that void as it pertains to the microcomputer industry and Apple by reporting on the progress of the people and the companies who comprise the "Apple industry."

Naturally—meant literally, as a result of natural phenomena—our good intentions for Tradetalk have been sidetracked at the first switch.

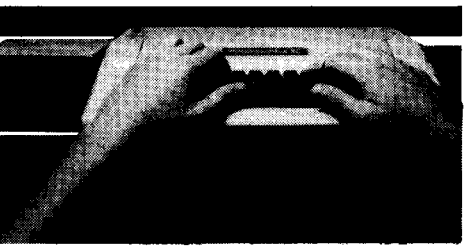
Powersoft, one of the early pioneers in software publishing, was literally struck by lightning in July, destroying the memories in all its computers, knocking out its telephone system, and destroying its entire inventory of product.

Lightning struck just after nine on the day before the Fourth. **Jim Powers,** president, reports their offices were "filled with a bright light, which was followed by a sound much like an explosion." Lightning had struck their heating ducts, jumped to one of their interconnected computers, traveled through all of them, destroying memory chips as it went, and leaped through a D. C. Hayes micromodem into the telephone system, knocking out communications.

The electrical charge was so great that it completely destroyed the inventory of cassettes and disks ready to ship as well as diskettes containing all of Powersoft's business activity records.

Powers reports that, with the assistance of his software authors, he and his staff have succeeded in recreating masters of all their software programs and are now well on the road to recapturing their business.

NEWSPEAK



There are those people who fear the ripening of the computer age; according to such prophets of doom, world takeover by machines is the ultimate outcome of the microcomputer boom. But the outcropping of general news stories concerning computers belies the naysayers. Overall, they show computers helping, benefiting, freeing their users in greatly diverse applications.

Newspeak is an attempt to digest these news items for those of you who might have missed them in the general newsmedia and to keep you up on the progress of Apple in the news.

☐ **Bumper Crop.** According to *Business Week,* Apple sold 65,000 computers last year and will double that number this year. And, despite Tandy's (TRS-80) sales of 135,000 in 1979, Apple is expected to take over first place in the micro market by 1982—because Tandy's projected sales growth is only 15 percent to 20 percent. All this in an article on Tandy. And none of the figures include the Apple ///.

The /// opens up some fascinating speculation. Aiming Apple /// at the very small business computer market brings Apple Computer Inc. into head-to-head competition with the big bopper, IBM, which claims 46 percent of this business.

☐ **Pedal Pusher.** On Monday, June 16, when thirty-one-year-old John Marino, of Newport Beach, California, set out to break his own record for crossing the USA by bicycle, an Apple went along. Throughout the twelve-day, three-hour, and thirty-one-minute trip from Los Angeles to New York's City Hall, covering 2,900 miles, Marino on his ten-speed was followed by a van in which sat the Apple, whose purpose was to chart Marino's course and his menu. Marino ate only food labtested to have no toxic effect on his body; he shared the van with the Apple for an average of four hours' sleep a day. And, yes, he broke his old record—by a full day.

☐ **Digital Yogi Bear.** There's no news in cartoons about computers, but now there's a computer that aims to be a cartoonist—one that's purportedly able to do the work of twenty-eight artists and technicians in producing animated cartoons for television. Actually, it allows seven people to do the work of thirty-five. The computer doesn't create the cartoons, but it does do many of the repetitive tasks required in animation.

According to Bud Hester, spokesman for the Motion Picture Screen Cartoonists union, as quoted in the Los Angeles *Herald Examiner,* the computer can paint backgrounds, do limited animation, copy pages, paint drawings, and pull out animation stock footage. Hanna-Barbera is the company leading the computer revolution in animation, and a revolution it may be if cartoonists facing less work have anything to say about it.

Disney Studios, on the other hand, declare that they'll never use computers. Their emphasis is on quality, and all their work is hand-done. It's no surprise that Disney finds unacceptable what's peaches and cream to Hanna-Barbera; but we wonder about the "ever" in Disney's declaration. The vast strides in the quality of general computer animation leads to the assumption that top quality capability is not at all far off.

☐ **Apple Goes to Camp.** The summer just passing saw the advent of a new kind of camp, the California Computer Camp in the Los Padres National Forest north of Santa Barbara, California. The brainchild of twenty-seven-year-old Denison Bollay, the camp offers the regular activities you expect at a good summer camp—swimming, horseback riding, cabin living, singing, and outings—to kids from ten to fifteen. With one difference. Twice a day, the activity for an hour or two is learning to program microcomputers. The camp promises one computer for every two campers, and there's a variety of brands. Bollay is easily talked into letting a computer go home with an ardent camper when the parents are willing to buy it, so by season's end there were fewer Apples than some of the other brands, and more kids on them. Free time finds about half the campers back at the computers, but this time with games.

The children at the camp are unusual in several ways, not only because they come from families well-off enough to pay $795 each to send them to camp for two weeks. Asked if any of them had ever worked, all but two or three responded that they had, and in meaningful, paying jobs. One was spending the summer working out a story line for a movie for MGM—*at* MGM; another was the receptionist for her physician father's computerized office.

☐ **Casting Couch Gives Way to Computer Terminal.** Universal Studios has about six thousand extras in its active computer files, according to the *Los Angeles Times* of July 8. When a production at Universal needs actors, the producers call on Al Jerumanis, director of Universal's computer operations. Jerumanis's department inputs the specifications for the roles and, within seconds, all actors who fit the description are displayed on a video terminal in the casting office. These people are called to check their availability, and within a couple of hours the roles are cast.

"What formerly took us until eight P.M. we now get finished by one in the afternoon," Jerumanis says. "The computer also prepares a work voucher which, when turned back into the computer, say, the next day at one, generates a paycheck before the afternoon is through."

The Universal casting computer specifies up to one hundred fifty personal characteristics, from measurements and coloring to background and special talents. In a matter of minutes, the computer can find a belly dancer who owns her own costume, a task that might take a conventional casting office an entire afternoon, according to Jerumanis; or it can call up all those extras with attractive hands to stand in for a star whose hands aren't "right" for a closeup.

A number of people in the business believe that computer casting will give greater opportunity to all actors, because the computer doesn't forget or overlook anyone, and because it doesn't play favorites.

Even better for little known players will be the interaction of the computer with videodisc players, a partnership Jerumanis foresees in the not too distant future. Then casting directors and producers will be able to see clips of each actor the computer has chosen from the specifications given at the same time as the computer is displaying the actor's dossier.