Dr. Dobb's Journal

of Computer Calisthenics & Orthodontia

FREE SOFTWARE

COMPLETE SYSTEMS & APPLICATIONS SOFTWARE
User documentation, internal specifications, annotated source code. In the two years of publication, *DDJ* has carried a large variety of interpreters, editors, debuggers, monitors, graphics games software, floating point routines and software design articles.

INDEPENDENT CONSUMER EVALUATIONS

PRODUCT REVIEWS & CONSUMER COMMENTS
Dr. Dobb's Journal publishes independent evaluations—good or bad—of products being marketed to hobbyists. It is a subscriber-supported journal. *Dr. Dobb's* carries no paid advertising; it is responsible *only* to its readers. It regularly publishes joyful praise and raging complaints about vendors' products and services.

REVIEWS

"A publication that is a must for everyone in the hobbyist world of computers. Don't miss it."
'Newsletter'
The Digital Group

"THE software source for microcomputers. Highly recommended."
'The Data Bus'
Philadelphia Area Computer Society

"It looks as if it's going to be THE forum of public domain hobbyist software development. Rating – ☆ ☆ ☆ ☆ ☆"
'TRACE'
Toronto Region Association of Computer Enthusiasts

"The best source for Tiny BASIC and other good things. Should be on your shelf."
'The Computer Hobbyist'
North Texas (Dallas) Newsletter

Dr. Dobb's Journal is published 10 times a year by People's Computer Company, a non-profit educational corporation. For a one-year subscription, send $12 to *Dr. Dobb's Journal*, Dept 57, 1263 El Camino Real, Box E, Menlo Park, CA 94025 or send in the postage-free order card at the center of this magazine.

people's computers

$1.50

VOL 6 NO 5 MARCH-APRIL 1978

SUBMITTING ITEMS FOR PUBLICATION

LABEL everything please, your name, address and the *date*;

TYPE text if at all possible, double-spaced, on 8½ x 11 inch white paper.

DRAWINGS should be as clear and neat as possible in black ink on white paper.

LISTINGS are hard to reproduce clearly, so please note:

- Use a new ribbon on plain white paper when making a listing; we prefer roll paper or fan-fold paper.
- Send copies of one or more RUNs of your program, to verify that it runs and to provide a sense of how things work – and to motivate more of us to read the code. RUNs should illustrate the main purpose and operation of your program as clearly as possible. Bells, whistles and special features should just be described in the documentation unless they're particularly relevant.
- Make sure your code is well documented – use a separate sheet of paper. Refer to portions of code by line number or label or address please, not by page number. When writing documentation, keep in mind that readers will include beginners and people who may be relatively inexperienced with the language you're using. Helpful documentation/annotation can make your code useful to more people. Documentation should discuss just which cases are covered and which aren't.
- If you send us a program to publish, we reserve the right to annotate it (don't worry, we won't publish it if we don't like it).
- Last but not least, please try to limit the width of your listings: 50-60 characters is ideal. Narrow widths mean less reduction, better readability and better use of space.

LETTERS are always welcome; we assume it's OK to publish them unless you ask us not to. Upon request we will withhold your name from a published letter, but we will not publish correspondence sent to us anonymously. We reserve the right to edit letters for purposes of clarity and brevity.

Cover illustration Ellery McKnight
(415) 325-9883

SUBSCRIPTIONS

U. S. Subscriptions
- ☐ $8/yr. (6 issues)
- ☐ $15/2 yrs. (12 issues)
- ☐ Retaining subscription @ $25 ($17 tax deductible)
- ☐ Sustaining subscription @ $100+ ($92+ tax deductible)

Foreign Surface Mail
- ☐ add $4/yr. for Canada
- ☐ add $5/yr. elsewhere

Foreign AIRMAIL
- ☐ add $8/yr. for Canada
- ☐ add $11/yr. for Europe
- ☐ add $14/yr. elsewhere

Payment must be in U.S. dollars drawn on a U.S. bank.

These back issues are available at $1.50 each:
Vol 5, No 6
Vol 6, Nos 1, 2, 3, 4

Foreign Distributors of *People's Computers*

Vincent Coen
LP Enterprises
313 Kingston Road
Ilford IG1 1PJ
Essex, UK

Rudi Hoess
Electronic Concepts PTY Ltd
Ground Floor Cambridge House
52-58 Clarence St
Sydney NSW 2000

ASCII Publishing
305 HI TORIO
5-6-7 Minami Aoyama
Minato-Ku, Tokyo 107
JAPAN

Home Computer Club
1070-57 Yamaguchi
Tokorozawa, Saitama,
JAPAN

Kougakusha Publ. Co., Ltd
Haneda Biru 403, 5-1
2-Chome, Yoyogi
Shibuya-Ku, Tokyo 151
JAPAN

Computer Age Co., Ltd
Kasumigaseki Building
3-2-5 Kasumigaseki
Chiyoda-Ku, Tokyo 100
JAPAN

People's Computers is published bimonthly by People's Computer Company, 1263 El Camino Real, Box E, Menlo Park, CA 94025.
People's Computer Company is a tax-exempt, independent, non-profit corporation, and donations are tax-deductible.
Second class postage paid at Menlo Park, California, and additional entry points.

people's computers

VOL 6 NO 5
MAR-APR 1978

CONTENTS

SPECIAL STUFF

ARTICLES

REGULAR STUFF

EDITOR'S NOTES

'. . . REWARD for information leading to the arrest and conviction of anyone reproducing our software in ANY way without our written permission. . .'

The above quote is NOT a paraphrase from a wanted poster for rustlers–it's part of an ad currently running in computer magazines. Software protection is an issue of growing concern. Unless the software producer can expect a reasonable return from his/her efforts we will not see the massive quantity of quality programs needed to realize the potential of home computers. Would-be freelance software producers and small software distribution companies are especially vulnerable, since they lack capital to pursue suspected software thieves through the legal system.

The issue hits close to home: people are trying to sell the copyrighted DRAW program published in this magazine. A distributor of PET software has told me that DRAW has been submitted for sale to his company by several individuals.

No obvious solution to the dilemma exists. Hopefully a general consciousness-raising will lead the vast majority of software users to realize that stealing software and contributing to the spread of stolen software has the undesirable long-term effect of discouraging software production. Meanwhile, rustlers are rustling, vigilantes are becoming increasingly vigilant, and those of us considering freelance software production are taking a long hard look at the rocky road ahead.

Phyllis Cole

LETTERS

Jim Day has not addressed the issue of 'The Computer as Art Critic' (Vol 6, No 3) at all, but only described a use of the Computer as Counting Machine. But besides this rather obvious fault, his remarks display two somewhat more significant errors.

The first of these deals with the nature of Art. It is of little consequence whether people like three green blobs better than two red circles. The art of the picture is greater than the sum of its parts. If this were not so, it would be a relatively trivial matter to write a computer program to analyse and correlate the parts of known art, and then extrapolate to produce vast amounts of new art to the same specifications. Why is it, for example, that three green blobs and two red circles appearing on one canvas contribute to its being 'Art', while the same forms on another (say, that of the artist's lazy understudy) only show it up to be an uninspired imitation?

The second of Jim Day's errors is, I think, much more serious. That is that he ignored the fact that art is a reflection of the artist's personal philosophy. Likewise, the critic expresses *his* personal philosophy in responding to the art. When these philosophies agree, the critic sees the art as 'Great' or at least, 'I like it.' When the artist and critic have differing *Weltanschauungen*, the critic will not like the work, and given the choice, will not aquire it for his future enjoyment. This is of course something of an oversimplification, but the point is that if you show a random set of art objects to a random set of people with no consideration for their presuppositions, you will get a random set of responses. But if you mechanically produce the elements of art (whatever that means), they will have as much significance as a mechanically produced collection of words: they may be interesting from a technical point of view, but since they are a reflection of a mechanistic world-view (either the programmer or his machine, depending on how you look at it) they will be appreciated only by critics who see the world mechanically. Those of us who do not consider people to be warm-blooded machines cannot see in this the same quality of Art as we see in Rembrandt or da Vinci.

The computer will only become an adequate art critic when the people making the evaluation become themselves computers.

Tom Pittman
San Jose, CA

Replies to Dennis Allison's notes on my letter in the Jan-Feb 78 issue of *People's Computers* on Tiny Languages:

Point 3: The reason this is important is that the user (a kid, probably) will be composing his program *at the keyboard*. If he *intends* to fit 12 more statements but there is only room for 6, we should ring the TTY bell (or beeper, or whatever) when available space is 75% consumed. Humans have a *poor perception* of storage requirements when coding at a keyboard. He can then alter his thinking and perhaps review what he already has, knowing there is 25% of the space left for changes.

Point 4: I do not follow your reference on unconditional branching in TILK. The trace branches referred to are on the *generated* pseudo-code level, *not* the source level, if you take the approach of generating pseudo-code. TILK source syntax has no GOTOs.

Point 10: Detection of 'Infinite Loops': A fundamental assumption we can make is that a program that *does not interact* with the user is probably malfunctioning. Since *kids* (inexperienced programmers) are likely to be writing programs in TILK, if their program does not request *input* or *output* often, say, a few thousand pseudo-instructions, we should make the *graceful assumption* the young programmer has *erred*. If he really does have a very long running algorithm (*unlikely* for a *beginner*), we can easily add a command to disable the check or the number of pseudo-instructions executed *between input* and *output* operations.

Todd Voros
Milwaukee, WI

HELP!!

We have a computer: Digital PDP8/a (identical to 8/e bus)

We have a printer: Mohawk Data Sciences MDS 7160. It's a remote print station with a 7 track tape drive and a drum printer rated at 1250 lines per minute.

We also have: thirty-seven student members, and one slightly tired faculty advisor, who do not have the slightest notion of how to interface the two devices. We know that it looks like quite a job. (6 months? 6 years?)

We need: H E L P

If you have any ideas for us in this dilemma, please contact me at your convenience. This printer is a real showstopper in looks, and I really hope that there is some way we can use it in our computer center, to give us the output capability we desperately need.

Incidentally, the computer center in this school has been supported through student fund raising, and will be available to every student in the school who has need for such a device. I believe it to be the only student-purchased computer center of its type for miles around!

Please help us if you can. Many people will be grateful for any assistance you can give.

W.J. Bajcz
Faculty Advisor
Lakeland High School
1630 Bogie Lake Road
Milford, MI 48042

"I'll have an oil on the rocks."

During an especially indolent December day I started to wonder what makes people play games. I also began to wonder why Bob Albrecht hasn't been pushing the Don Quixote Starship lately. The second question was easier to answer than the first, because the first is questioning the thought processes of over four billion people. While the second is only trying to fathom the incongruencies of an old dragon's intellect.

Second things first. There are two possible answers to the second question: either Bob hasn't been doing any synergistic beer drinking lately and has given up on the idea, or he has been putting all his effort into the work he and Dennis Allison are doing with Tiny Languages. It had better be the second reason, for if it's the first I won't renew my subscription.

First things second. What makes people play one game instead of another or even play games at all? I'm not an expert by any means, but I figure that I've spent about 200 hours in a computer room so far this school year. I have watched all types of people play games. From a programmer that knows five or six languages to a non-computer person, the motivation behind playing seems to be the need to win. I think I see a correlation between the amount of computer experience a person has and the kinds of games he/she plays. Keith, who knows five or six programming languages likes to play COMBAT; Bill likes LIFE (he has been programming for quite a while); I like STAR TRADER. Everyone (well, almost everyone) likes STAR TREK. However, the non-computer people tend to go for a different kind of game. I think the favorite game for non-computer people (at least around here) is OREGON. Evidently someone explained the ENTER statement to them for it is fun to watch them type 'Bang' as fast as they can. I guess that perhaps a personal challenge is needed by all game players and the possibility of pseudo-death gives the player a sense, however small, of reality.

Now for some ideas I got after answering these questions for myself. Most of the talk concerning the Don Quixote Starship has been theoretical. Now mind you I have nothing against theory and am myself an idealist but I have answered all my DQS questions and am ready to proceed with the physical planning and implementation of the thing. Here are some ideas for an interactive, progressive, realistic computer space simulation:

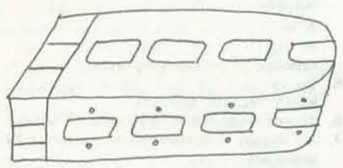
1. It can be accessed by many terminals at the same time. In fact, it will work much better with many.
2. All types of people must be represented in the program. Good guys, bad guys, etc.
3. All needs of a space society must be foreseen and met with timely and adequate answers.
4. Political, social, and biological differences between species must be foreseen and again, an adequate and timely answer must be supplied and changes implemented.
5. Entire game in four dimensions. Length, width, breadth, and time.
6. Speeds measured in kilometers per second.
7. All money is automatically transacted electronically.
8. There are three types of space ships:
   A. Trading ships
      1. Captains pick their own speeds. Cargo limit is inversely proportional to speed.
      2. Unit of currency is the URanium SPecie unit (URSP). One URSP equals the cost of one mole of Uranium 238.
      3. Purpose is to make as much money as possible by trading merchandise.
   B. Aggressor ships
      1. The equipment is crude and bulky but the aggressor has great mobility.
      2. The aggressor's purpose is to attack enemy trading ships, enemy defending ships, and hostile planets.
      3. After an aggressor has successfully crippled an enemy trading ship or enemy defending ship, the aggressor absorbs all energy, fuel, etc. from the

crippled ship. Successfully conquering a planet gives command of that planet to the aggressor. The program automatically CHAIN's over to a 'KING'-like program after a planet has been conquered.

C. Defender ships
   1. The defender ship has more sophisticated equipment than an aggressor but it is usually defending either a trading ship or a planet and can't wander far off.
   2. Purpose of the defender ship is to defend planets and trading ships.
   3. When defenders protect trading ships and planets they are paid in uranium (fuel for their engines).

9. The trading rules and merchandise are basically like STAR TRADER.
10. Star system classes are as follows: I-Urban, II-Suburban, III-Fringe suburban (rich), IV-Rural, V-Frontier, VI-Industrial, VII-Specialized: VIIa-Penal, VIIb-Entertainment, VIIc-Medical, VIId-Educational, VIIe-Retirement, VIIf-Science research and development.
11. High taxes may motivate some players to turn pirate.
12. All ships are controlled by humans. No computer-controlled ships with perfect moves or purposefully botched up moves.
13. Aggressors and pirates have the ability to change the time and point in space the ship is sitting in.
14. At the point at which a player enters the game he/she decides to be either a trader, aggressor, or defender. This should give most people a chance to express themselves politically within the game.

That's about all concerning DQS except the drawing I sent along. It is an aggressor ship. The little holes above and below the port holes are fixed lasers.



I hope you don't mind if I solicit help from your omnicient readership. Could someone please fill me in on the PICTURE attribute in PL/1? Also, when are you going to print a listing for a chess game in BASIC? (I'm kidding again—or am I? Sometimes I can't tell).

Another thing: I wish you would do some calculator stuff. Especially the programmables. And while learning math with calculators may seem elementary there is a specific art to it.

You ought to (there I go again, playing editor—sorry, Phyllis) have some articles teaching the very basic parts of other languages like PASCAL or SNOBAL. I realize that you are trying to prepare people for the home computer boom but you must realize that kids are getting access to more and different languages in school and have to learn them somehow.

And you know as well as I do how expensive manuals are. When, when, when, when, when are you going to go to monthly publication????????? I almost need you monthly. After about a month and a half I find myself (it's almost too hard to say) looking at other computer magazines. I don't mean little stuff like *Creative Computing*, I mean the heavy stuff like (shudder, shudder) IEEE *Computer* and (shudder, shudder) *Byte*. I try, I really do, but the bug has bit too hard.

Kudos on your robot articles. Keep 'em coming. Let's hear some more from PISA—please. I would like anyone who is interested in *amateur* rocketry to contact me by mail.

Also, thanx for keeping your format as a cockeyed concoction of intellectualism, craziness, zaniness, and crud. Without the last we wouldn't be able to distinguish the first three. Nuf said.

Phil Dolan
7415 Portland Ave
Richfield, MN 55423

□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□

I once read about a person putting Buddhist prayers on a disk. I wonder if you could publish any such prayers? Then, on the theory that they can't hurt and might help, your readers could all put prayers to spinning on the various disks they have access to. I will be glad to put them on my disks if you print them in *People's Computers*.

As the people who built Notre Dame, I remain,

Anonymous

We have a policy not to publish any correspondence from Anonymous, but in this case an exception seemed in order. Anonymous' letter, mailed in Bloomington, IL, was the second rumor we'd heard as to the use of such prayers on a disk–perhaps as a modern-day prayer wheel? And yes, we found the rumors to be true–see page 44.

□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□

The TEASER game, described in the September '74 issue of *People's Computer Company* and also in *What to Do After You Hit Return,* is a very clever and popular game. Has anyone ever noticed that the diagram showing the '102 possible positions (excluding rotations and reflections)' is incomplete? There are, in fact, more than 102 possible positions. *People's Computers'* readers may enjoy deriving a complete analysis. (Hint: look for complementary positions.)

Jim Day
Granada Hills, CA

□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□

I like the new magazine format except for one thing. Leafing through old issues I noticed as I got to the newer issues that you have less of the good old ink drawings–*not good*. They are (were) one of the things that set your mag apart from others. So please bring back the dragon stuff–I miss the dragon emeritus.

I have 102 different games to exchange with anyone. Keep up the good work.

Douglas 'Dit-Dit' Philips
Box 329
Venetia, PA 15367

"I compute, therefore I am."

□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□

THIS MACHINE

This machine, it played one,
It pushed START and PROGRAM RUN,
It's an I-B-M 360-85,
This computer came alive!

This machine, it played two,
Told programmers what to do,
It's an I-B-M 360-85,
This computer came alive!

This machine, it played three,
Printed out errors endlessly,
It's an I-B-M 360-85,
This computer came alive!

This machine, it played four,
Spit its punch cards on the floor
It's an I-B-M 360-85,
This computer came alive!

This machine, it played five,
Wrote pure noise onto tape drive,
It's an I-B-M 360-85,
This computer came alive!

This machine, it played six,
Told the C. E. what to fix,
It's an I-B-M 360-85,
This computer came alive!

This machine, it played seven,
Decided it was sent by Heaven,
It's an I-B-M 360-85,
This computer came alive!

This machine, it played eight,
Shipped itself to Rome air-freight,
It's an I-B-M 360-85,
This computer came alive!

This machine, it played nine,
Told the Pope it was divine,
It's an I-B-M 360-85,
This computer came alive!

This machine, it played ten,
To sing once more push START again,
It's an I-B-M 360-85,
This computer came alive!

Lyrics (more or less) liberated from the Science Fiction Folksong Songbook. All responsibility denied.

Contributed by the Milwaukee Science Fiction Club via Lee Schneider.

□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□

READER SURVEY

We've had a gratifying response to the reader survey published in our last issue. We've tabulated enough returns to begin to form a picture of our readers. About 50% of you are computer professionals and about 25% educators while 70% identify with the category 'hobbyist.' Apparently some hobbyists don't yet own working computers (55% of our readers do), but 75% of you expect to purchase a home computer or additional equipment within the next 6 months. About 40% of you have extensive computer experience and another 40% are fairly experienced. The balance have little or no experience.

Not unexpectedly, 96% of our readers are male. About 50% of you are 30-50 years of age, 30% are 19-30, and 15% over 50, with the rest 18 and younger. A whopping 80% have at least a bachelor's degree—and many of you who don't just haven't yet completed college.

To date, education is the category in which most readers (70%) expressed interest. About 45% of you want to provide materials for family and friends; interest is equally divided in teaching people of varying ages. Games are of importance to many (60%) as were small business applications, scientific applications, household records, and graphics (about 40% each). Approximately 20-25% are interested in each of the other survey categories except for work with the handicapped, which drew the attention of about 5%.

Issues concerning what you want more or less of are hard to get a handle on—you're a very diverse group. About 25% want more related to languages and programming, another 25% are interested in more applications such as games, graphics, education, and CAI. About 15% would like to see more hardware related articles. For everyone who loves Fortran Man, someone else hates it—with 15% objecting to date.

Ah well, we'll just keep doing the best we can!

□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□

The letter referred to below appears in column 2 of page 58.

After my letter to you, a copy of *An Introduction to Your New Pet. . .* arrived. It isn't what I had hoped for, but it is a start in the right direction. In addition to an informative dialogue on simple BASIC the 38-page booklet included some interface info, a brief memory map and some discussion of first-order PET trouble shooting.

The PET repair service is SUPERB! My PET was returned in good running order exactly one week after it was shipped to them.

A new copy of the PET warranty arrived, which was filled in and returned. I assumed it is to be effective this date for 90 days. If so, BRAVO for Commodore; they stand behind their products.

My big ears have caught some conversation nth hand from friends that Commodore does not intend to come out with a manual or booklet on the PET machine or assembly language. If PET has such plans, I suggest you publish articles about the PET machine language.

Philip Gash
Redding, CA

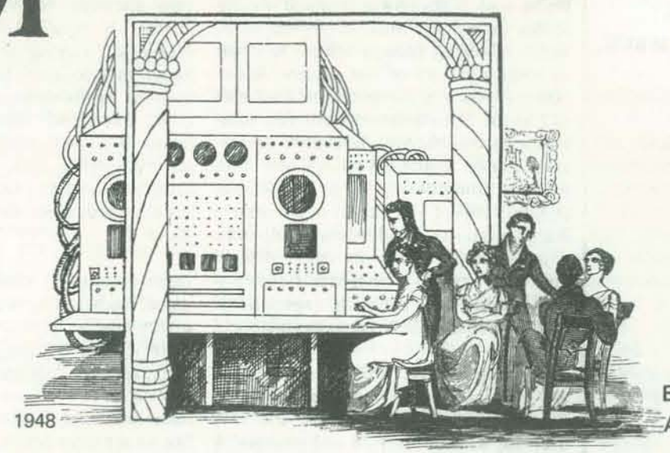
Commodore is publishing its cassette Monitor 13.1 along with instructions. Meanwhile there's a Kim-oriented 6502 Programming Manual available from Kim dealers and 6502 specifications are available from MOS Technology. Our 7-part Data Handler series, which concluded last issue, also dealt with assembly programming of the 6502.

□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□

Sorry to hear that the Data Handler User's Manual will end with Part 7. I hope Mr Don Inman will write a complete book on the operation of the Data Handler. What do I think of *People's Computers*? IT'S GREAT.

John Waskowitz
Jackson Heights, NY

MICROCOMPUTERS in 1978



BY
ADAM OSBORNE

Since 1975, Adam Osborne's books have been introducing people to microprocessors and their applications. His well-written, comprehensive books have been enthusiastically greeted by microcomputer fans. Of special interest to newcomers to the field is his series An Introduction to Microcomputers, *available through the PCC Bookstore. Volume 0, 'The Beginners Book,' (see review in this issue) assumes you know nothing about computers, math, or science (300 pages, $7.50). Volume 0 prepares you to read Volume 1, 'Basic Concepts', which explains concepts common to all microcomputers, but specific to none (350 pages, $7.50). Volume 2, 'Some Real Products', was revised in 1977 by Osborne, Susanna Jacobsen, and Jerry Kane. This Volume describes every common microprocessor and their support devices (1200 pages, $15.00).*

People's Computers *is pleased to present Adam Osborne's look at microcomputers in 1978.*

1978 is a year that will bring massive upheaval to the semiconductor industry and the microprocessor market, but very little of this will filter through to the average microcomputer user. In this paradox we see one of the most interesting phenomena of the semiconductor industry—the fact that chip fabrication technology is moving far faster than any user can hope to keep up with. And it is very important that microcomputer users understand this phenomenon; you must come to terms with the fact that there will always be radically new microprocessor products and that they may be far more powerful than the microprocessor you are using now, but that is no cause for panic.

In order to understand the realities of this situation, let us look at what is going to happen in 1978. During 1978 semiconductor manufacturers will pour forth an incredible variety of new products and enhanced products. Beginning with microprocessors themselves, 1978 is likely to be the year of the one-chip microprocessor and the 16-bit microprocessor. One-chip microcomputers are not very interesting to the average 'hobbyist', therefore let us look at 16-bit microprocessors.

First of all, there is Texas Instruments, who for a long time have had the TMS 9900 but have chosen to do nothing with it; they feared that TMS 9900 sales could damage their upwards compatible TMS 990 minicomputer systems business. (The TMS 990 minicomputer has exactly the same instruction set as the TMS 9900 microprocessor.) Texas Instruments have partly changed their minds, and will support the TMS 9900 more aggressively in 1978. Were they to throw their full efforts behind the TMS 9900, it would create havoc within the microprocessor industry; via the TMS 990, the TMS 9900 has potentially the most formidable software base of any microprocessor in existence. Unfortunately for microcomputer users (but fortunately for other microprocessor manufacturers), Texas Instruments is not likely to make much TMS 990 software available to TMS 9900 users.

Next there is the MicroNova from Data General and the 9440 from Fairchild; both of these products are now becoming available in commercial quantities. Fairchild will be offering a 3-card 9440 system that is compatible with the S100 bus. The three cards will provide a Nova compatible instruction set CPU plus 16K bytes of read/write memory.

The LSI-11, though not strictly a microprocessor, is an equivalent product now commercially available through Heathkit. Like the MicroNova and 9440, the LSI-11 has an extensive existing software base which can be run on the new microprocessor products.

But there are also a number of brand new 16-bit microprocessors on the way. Most noticeably there are the Intel 8086 and the Zilog Z8000, both of which will be available in mid 1978.

It is unlikely that there will be any brand new 8-bit microprocessors in 1978 or thereafter, because it would simply be too hard to convince users that a new entry into this already saturated market it worth considering. For that matter, it is not clear that the 16-bit microprocessors will establish significant markets for themselves, but at least the fact that they are 16-bit devices gives them a basis for hope.

What about support logic? Probably the most significant new support devices to appear in 1978 will be peripherals' controllers—floppy disks, CRT and keyboard controllers. Do not be misled into thinking that these are really one-chip controllers; they are not. But they do provide on one chip much of the logic required by control interfaces to appropriate peripherals, thereby significantly reducing cost and chip counts for interface logic.

In addition to the peripheral device controllers, there will be a number of new support parts offered to enhance existing microprocessors. These support parts will do two things: they will provide functions previously unavailable, and they will provide previously available functions combined on single multifunction chips. Previously unavailable functions include direct memory access and priority interrupt control for microprocessors that did not have it, plus analog-to-digital and digital-to-analog converters and telephone communications devices. Multifunction devices will provide read/write memory, parallel input/output, primitive interrupt control logic, and timers on a single chip, making it possible for small microcomputer systems to be configured out of just two or three chips.

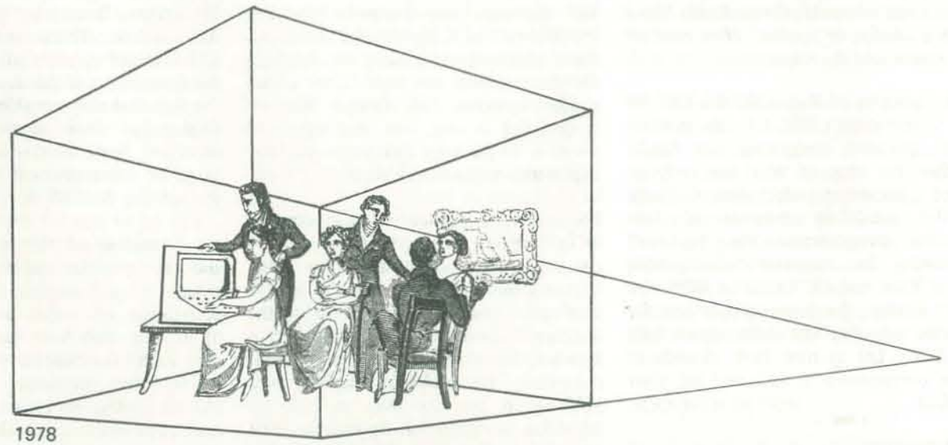
What does all of this mean to the microcomputer user? Strange to say, not very much. The microcomputer industry—characterized by the IMSAI/ALTAIR-type microcomputer, became a reality when microprocessor prices dropped low enough for such products to be commercially viable. The 8080A, 6800 and 6500 microprocessors were quickly adopted by this new industry.

The earliest microcomputers were of little value to anyone, since they had no software, no peripherals and no support devices. Over the past two years a significant body of software has been developed—for the 8080A, 6800 and 6500 instruction sets. A large number of peripheral devices is now available—providing you have an S100 bus. A bewildering profusion of memory cards and other support cards is available—again presuming you have an S100 bus.

Just because a large number of 16-bit microprocessors appear in 1978, is everybody going to abandon products already on the market? By no means. However great these new 16-bit microprocessors may be, they will initially have very little software and very few peripherals, and they will demand, should you decide to adopt them, that you give up everything you have done thus far and start again.

But there is nothing particularly new in the situation I describe. The way the microprocessor industry works, and for that matter the way the whole semiconductor industry works, is that they introduce radical new products and dramatically drop prices, at which point new markets emerge based on the new economics. These may be looked upon as 'entry points'. A new industry emerges when it sees its entry point. Once an entry has been made, an enormous amount of time and money must be invested in areas that force you to stay with the microprocessors and semiconductor devices that caused your entry point. Thus, industries that spring up based on their ability to use microprocessors stay with their entry point microprocessors for a long time. Any move to a new product is held back by the inertia of redeveloping new software and support peripherals.

For your part, you should ask yourselves the following question: 'Is the hardware I now have adequate for the job I wish to do?' If the answer is yes, then the new developments of 1978 are not particularly important to you, since you will reap relatively little benefit from them. If you find that your current needs are not being met by available microprocessors, then you are waiting for a new entry point. The new products I predict for 1978 may bring you to your entry point. □



TRS·80
A Consumer's Computer?

BY TOM WILLIAMS

This review concerns itself with the 4K version of the Radio Shack TRS-80 Z-80 based microcomputer system. The TRS-80 with 4K Level I BASIC sells for $599. Although Radio Shack is providing for expansion of the system, we limit ourselves to the 4K unit supplied to us for review–that is, the system you can buy for $599.

The following options for the TRS-80 are now available from Radio Shack: A 12K Extended BASIC written by Microsoft and incorporating the full Z-80 instruction set, on ROM for $99; 12K additional RAM (for a total of 16K user memory) for $289.95. The 12K RAM expansion consists of replacing 8 4K chips in sockets on the main board with 8 16K chips.

Since our review of this system has not been favorable, we have held off publishing it until we could provide Radio Shack with a chance to respond. Here now are the review and the response.

With the entry of Radio Shack's TRS-80 and Commodore's PET into the marketplace, personal computing has finally reached the stage of what can truly be called consumer electronics. Radio Shack's candidate consists of four separate components: the keyboard containing the computer circuitry with 4K of RAM and 4K BASIC on ROM, the video display, the power supply and the cassette recorder. The entire system sells for $599. Let us now look at each of these components in turn and see what we find.

HARDWARE

Radio Shack advertises its full-size 'professional-type' keyboard as a big plus. The keyboard is indeed nicely low-profile with well-spaced keys much like a selectric typewriter. Our unit has a slight clunky feeling to the keys although other people we've talked to report that they are quite satisfied with the smooth feel of the keys . . . especially the space bar. The keyboard does have one annoying feature, however: it lacks rollover.

Rollover is the ability of the keyboard to detect a key being depressed while the previous one is still being held down. A well-designed keyboard will then print the second key's character as soon as the first key is released. The result of not having this feature on Radio Shack's keyboard is that you tend to lose track of characters if you get up any typing speed. You then have to stop, type back arrows (←) back to the mistake and then continue.

The keyboard and computer circuitry are enclosed in a plastic case which has three identical receptacles on the back for video, power and tape. There is also a plastic access door (which falls off frequently) in the rear that opens to reveal a 40 pin edge connector–for later expansion–and a reset button.

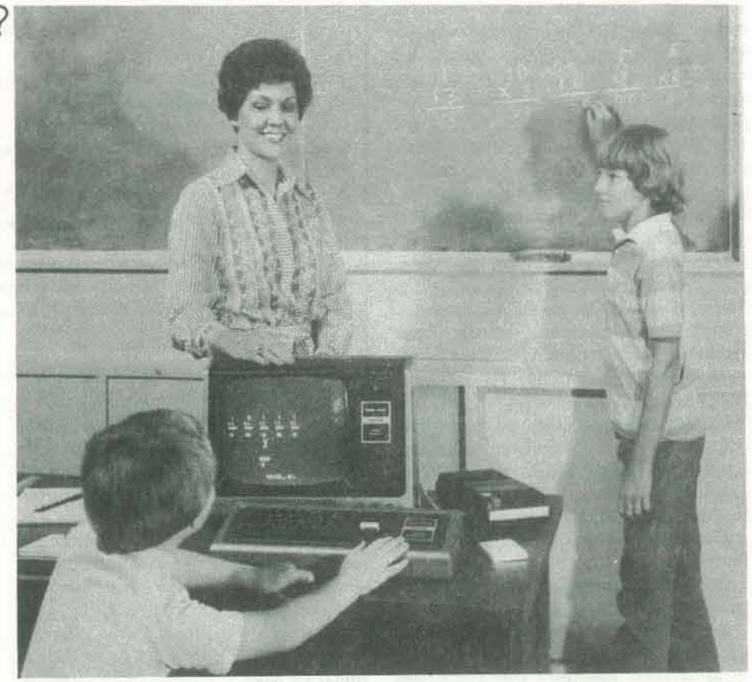
The twelve-inch monitor, also encased in light plastic, displays 16 lines of 64 characters. We looked inside the video unit and the verdict was that the display is of rather poor quality and is more like a stripped-down television than a true monitor. Monitors designed for use with computers are built to display more information per line than an ordinary television is capable of. By making some internal changes to a TV it is possible to 'push the limit' and that seems to be what Radio Shack has done. The display on the whole is a bit fuzzier than professional ones we've seen and on our unit the 'M' and 'W' were rather indistinct. We also experienced a slight wiggle that repeatedly walked up the screen and didn't seem to want to go away.

A feature related to both the computer and the display is the way characters are transferred to the screen. Most home computers do not accept new input while they are performing some computation and the TRS-80 is no exception. During that time, the computer should also not display input it is ignoring. The Radio Shack system *does* display typed characters, whether or not they are accepted as input by the computer. This can be quite annoying when you have a display like the backgammon board which must then be redrawn on the next move.

Of the four separate units that make up the system, three must be plugged into AC outlets. There seems to be a difference of opinion among users as to the desirability of this feature. Some like the fact that they are able to position the display to their convenience; most, however, feel inconvenienced by the kluge of wires involved with setting up and moving the TRS-80.

An evaluation of the cassette recorder and its operation takes us into some aspects of software, a more complete description of which will follow. All of us here who have used the TRS-80 have found the cassette recording system to be very unreliable. We have had trouble loading programs from tape and have experienced particular exasperation when trying to save programs we had keyed into the machine. Here is a typical sequence of events when the machine fails to save a program:

▬▬▬▬▬ at school?



1) Program has been keyed into the machine, RECORD and PLAY keys on the recorder depressed, the command CSAVE entered on the computer.
2) The computer starts the recorder but after what seems a more than adequate time fails to return the READY on the display which indicates a successful 'save.'
3) Remove REMOTE plug from recorder and depress STOP key.
4) Rewind tape.
5) The computer is still trying to write the tape, so open rear access door and depress RESET button and/or turn off computer and then turn it back on.
6) Is your program still in memory? If not, retype it then replace REMOTE plug and start over again.

We have discovered that the cassette communicates with the computer at 250 baud (slower than most other cassette formats) and by using an oscilloscope have found that recording demands a bandwidth of about 250 to 10,000 Hertz. This is very wide and is the source of many of the recording problems.

SOFTWARE

Turning now to the software, we can log a couple of plusses. The foremost is that when an error is encountered in a program, Level I BASIC prints 'WHAT?' then reprints the line with a question mark inserted where the error was encountered. This is a valuable tool in debugging. The second nice feature of Level I BASIC is that when given an invalid input, it doesn't just 'go ape' but reprints a prompt and allows the user to try again. An additional convenience allows the user to abbreviate most of the commands. So far we've found no bugs in Radio Shack's BASIC.

The capabilities of Level I BASIC are rather limited and may be a disappointment to the naive user who expects to take the unit home and key in 'Lunar Lander' or some other program requiring square roots, exponents and/or trig functions. Only two strings exist, A$ and B$, and the only use we've been able to find for them is inserting them in PRINT statements. You cannot compare strings or manipulate them in any other way.

The TRS-80 does have a limited graphics capability that is pretty versatile for a 4K machine. The graphics field is 48 X 128 points of oblong shape that can be SET (made to show up light) or RESET (light portions of the display turned dark again) by designating the X,Y coordinates. The system does not have graphics characters.

Since Level I BASIC does not provide for named files, the user must keep track of his programs by means of the digital counter on the cassette recorder. This is not as great a problem as it might seem as long as you save only one program per tape. This is a good practice anyway, and Radio Shack is marketing tapes short enough that you can do it without leaving 80% of your cassette unused.

When you buy the machine, you get a tape with two games: Blackjack and Backgammon. We found the Blackjack game to be very ill-conceived. You can bet a negative number, lose the hand and as a result win the money. The Personal Finance Package which we purchased was very poorly documented (no examples) having less than 1½ pages of text and a very confusing flowchart (no explanation) which is almost certainly unintelligible to the beginner. At one point the program asks you to load the 'cancelled checks' data tape without ever telling you how to get data onto that tape.

The Payroll Package we had a chance to try out was, however, much better. We found the instructions to be clear and the program well thought out. These advantages were, nonetheless, negated by the crudeness of the file system. In fact, the main problem in using the software packages is that you can't save and read tapes reliably. This problem coupled with often poor documentation will render even the best software package worthless.

DOCUMENTATION

Tandy Radio Shack is justifiably proud of their fine beginner's manual, written by Dr David A Lien of San Diego, which is designed to be used in conjunction with

the TRS-80 and Level I BASIC. An underlying light touch and sense of humor make this well-written easy-to-read volume an especially pleasurable reading experience.

The twenty-six chapters in Part A are well thought out, and provide ample exercises to allow you to evaluate your understanding of the material. (Answers to these exercises are in Part B of the book.) It's assumed that the reader is an inexperienced computer user, but a reasonably intelligent adult. The material is paced for the beginner, but the clear structure of the book (including handy summaries inside the back cover) make it easy for the more experienced programmer to use the book for reference. Lien covers a great amount of material, including the issue of accuracy—eg. when the TRS-80 Level I BASIC multiplies 2/3 by 3/2 it gets 1.0000009 instead of 1. As the author notes in his own inimitable style (page 71):

> TRS-80 users who have LEVEL II Basic will not notice this routine 'rounding error'. If we solved all the world's problems with the bottom-of-the-line machine, you might not want to upgrade to the higher power models, and one doesn't stay in business long that way, does one?

Part C of the manual consists of listings of 'Prepared User's Programs' which cover topics such as test grading, a program for a 12-hour clock, speed reading, computing the Dow-Jones Industrial Forecaster (as described in a June 1977 *Forbes* article). A number of programs which exercise the graphics capability of the TRS-80 are also included.

The manual concludes with three Appendices. Appendix A contains annotated listings for math functions not directly available in Level I BASIC, such as square root, exponentiation, logarithms, and trigonometric functions. Appendix B covers cassette data files; Appendix C discusses various trouble-shooting tests to perform if you suspect computer problems.

CONCLUSION

A word should be said about Radio Shack's advertising of the TRS-80. On page 13 of the Christmas Sale Catalog number 293 the reader is advised to borrow $600 to give Pop a TRS-80 because 'Pop will make it all back using TRS-80 as a business tool, believe me.' From what we've seen, 'Pop' might be better off with a $7.95 abacus. Ad brochures talk of a 'high resolution screen' which is no such thing; salesmen talk of 'full graphics capability' to customers who are impressed by what they see but who don't know what else is possible and have no standards by which to judge computer quality. 'String capability' gives no hint of the *limited* string capability in the 4K machine. 'Cursor control' in the 4K machine means you can move the cursor forward and backward and that is *not* what cursor control means in the world of computers.

Yes, the era of personal computing has finally reached the stage of consumer electronics. In my view, however, the TRS-80 represents 'cheap electronics.' The entire feel of the system with its display and keyboard encased in light plastic is more like that of a toy than that of a seriously designed computer and certainly not like that of a business machine. The TRS-80 represents, in my view, an attempt to capture a vast consumer market that is ignorant of the details with a quick and cheap machine and is a disservice to the personal computing industry as a whole. Radio Shack would be best advised to tone down their advertising hype, improve their product or mercifully remove it from the market.

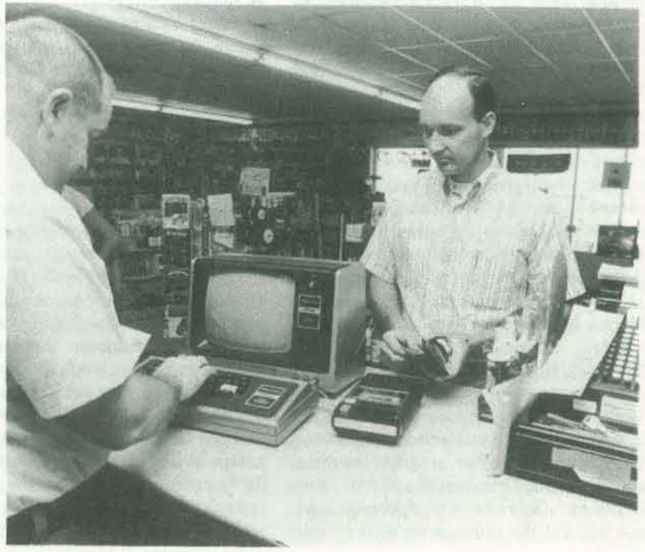
TRS-80 PERIPHERALS

Radio Shack stores are now taking orders for March 1st delivery of various TRS-80 peripherals.

'Professional' printer:	$ 1299. 95
Thermal printer:	599.95
Floppy drive:	499.95
Expansion device:	299.95

The expansion device contains sockets for up to 4 floppies and 2 cassette drives. It is needed as a controller for all of the above peripherals and for expanding memory beyond 16K; it can handle more than one device at a time.

at work?



Radio Shack Responds

By Hy Siegel
National Publicity and Promotion Manager
Radio Shack

Thank you for providing us the opportunity to respond to your review of the Radio Shack TRS-80 Microcomputer System.

We were surprised that you objected to the housing material which is a good grade of ABS plastic specfically selected because it is light in weight yet highly durable.

Regarding the cassette recorder, our engineers advise me that required bandwidth is only 1000 to 4000 Hz, rather than the 250 to 10 kHz that you indicate. It has also been their experience that loading problems are almost invariably the result of operator error, rather than a problem with the machine.

It is true that you can bet a negative number in our blackjack program—it even says so in the instructions—but, really Tom, that's hardly according to Hoyle and you wouldn't do it in Las Vegas, would you? We feel that it's much more worthwhile to note that our blackjack program does keep track of one or two decks of cards rather than simply being a random number generator.

Cursor control in the TRS-80 is accomplished by using the PRINT AT command. This function is unique to the TRS-80 and we feel that it is a positive feature.

Overall, we believe the TRS-80 to be a well thought out and carefully designed microcomputer system that offers excellent dollar value, and, apparently, so do the thousands of people who are already using the 4K TRS-80 system in home and business applications.

Far from removing the Radio Shack TRS-80 from the market, we plan to continue to market it, expand it, provide peripherals and extensive software support.

A Reader Writes

In response to Mr. Williams' call for comments on Radio Shack's TRS-80, I, for one, am delighted. I endured the frustration of building a computer from the ground up for use in my business. It was quite a delight for me to simply plug in the TRS-80, push a button and see BASIC in ROM up and running right away. True, the Level I BASIC is limited in the string department and the graphics are as slow as the dickens, but that may change. After all, simple software is intended for simple use, though many programmers would disagree. I invite you to look at what you can do with that simple software before you pass judgement. Radio Shack assembled some very useful programs for home and business, some of which might not be further improved with a fancier BASIC.

But let's look at the machine itself. The designers used a good keyboard, the Z-80, and a very good cassette interface method born out of the old Computer Hobbyist Newsletter. The pulse modulation method is more reliable than some techniques used in other systems today.

There was a time less than two years ago when all the computer magazines could talk about was the ostensible 'War of the Processors'. Indeed one could perceive it as such what with the Altair clock troubles and the Imsai 'memory clobbering phantom'. I endured all of that talk of comparison. But now *People's Computers* wants to compare the PET to the TRS-80. I see it as a comparison of a Ford to a Chevy. Why?

I know 3 others that have stopped working with their SOLs and Imsais long enough to buy a TRS-80 and see what it will do. We all seem to like it. And none of us work for Radio Shack!

If you do choose to take a negative stand, I suggest you temper your arguments with the certain knowledge that 'However Radio Shack goes, so goes the world.' They've got the distribution! Please don't take all this as a challenge. After all, they could have chosen to use the 4004 in a partial kit delivered in a bubble pack with a keypad in octal!

Stephen Gibson
Gibson Engineering
Los Angeles, CA

at home?

REVIEWS

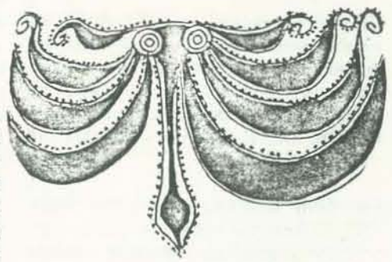
COMPUTERS AND
THEIR SOCIETAL IMPACT
by Martin O. Holoien
John Wiley & Sons, 1977
264 pp, $10.95

The author apparently planned this text to introduce students and the interested public to computers and their domain. He tries to indicate the ever increasing role that computers play in our lives and to give readers a small taste of programming via BASIC. He met his goals and did a creditable job in this well-written and easily understood work.

He covers, in order, a history of computational devices from man's fingers through Charles Babbage and Lady Lovelace, twentieth century information processing devices, how one communicates with a computer (Chapter 3 is a mini-course in BASIC), the present-day use of computers in the fields of education, business, industry, politics, government, law enforcement and health, and, finally, the role of computers in our future. There are two appendices, one on flowcharting, the other on the use of the Teletype ASR-33 terminal. Each of the nine chapters in the book is concluded with a set of exercises and a list of selected references covering the chapter.

Other than the author's confusion of privacy with anonymity (a difficult concept to separate) and the lack of a discussion of mini-computers and their development and importance to today's society, this is an excellent book. I recommend this work highly as both a text and a simple exposition of the computer 'explosion' of the past two decades. Thus I see it as a book of value to the laymen, student, teacher and computer hobbyist.

Reviewed by Willard J Holden.



AN INTRODUCTION TO MICRO-COMPUTERS
Volume 0: The Beginner's Book
by Adam Osborne
Osborne and Associates, 1977
300 pp, $7.50

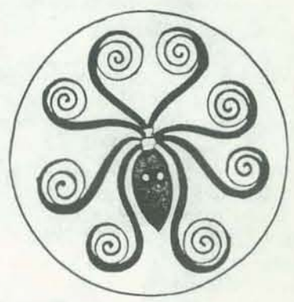
Back when I was selling Altairs, people used to wander into the store and stand uncertainly staring at the mysterious machines until I came to the rescue with a 'May I help you?' The reaction was usually one of relief followed by a request for information such as, 'I don't know anything about these things but my brother-in-law says I can use one in my furniture store.'

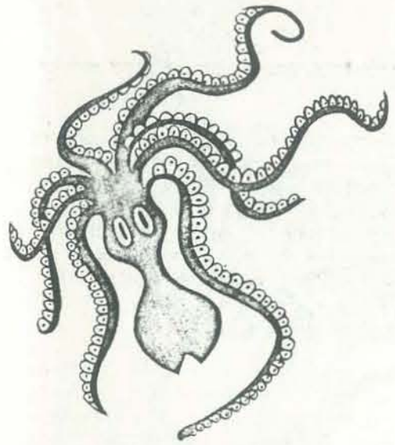
I would then launch into an explanation and unabashed sales pitch. Most people would end by asking what they could read that would get them started understanding computers. It was then that I wished I could have referred them to a book like Adam Osborne's *Volume 0*.

The book attacks the subject of microcomputers and what they are at the very point where the novice first encounters them: the assembled and running machines themselves. An overview of different system components and how they are related to each other within the system provides the reader with the first level of understanding what is going on. A generous amount of space is devoted to explaining the various things that can happen when a key is depressed, and this forms the basis for showing how parts of a system can be physically connected together yet *logically* linked or separated in different ways. The discussion then moves on to memory and mass storage and the ways files are formatted on floppy disks and cassette tapes.

I believe the chief advantage of *Volume 0* is the truly ingratiating way it relates to the novice's level of sophistication. Bruce Mishkin's cartoon character, Joe Bitburger, is a naive but enthusiastic beginner and he is used to point out some of the disillusionments the beginner can fall prey to. Joe is also used to explain some of the fundamentals of how computers can do what they do. The impracticality of using switches and lights to do anything useful leads into a discussion of the teletype terminal.

Once the basic functions of certain system components have been made clear, *Volume 0* devotes a whole chapter to a sort of buyers' guide, or 'what to look for when you go shopping' for hardware components. There is a particularly good discussion of video display options: cursor control, text insertion and deletion, vertical and horizontal scrolling, etc. The section on keyboard options has a detailed description of 'rollover' and why it is an important consideration in selecting a terminal. Other options discussed include paper tape, punches and readers, printers, cassette units and floppy disk drives.





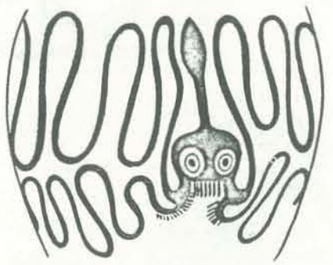
The first three chapters of *Volume 0* can be considered an orientation course which bestows the confidence to plunge further into this strange new world. There follows a guided excursion into the perilous realm of binary arithmetic. This is, of course, run-of-the-gauntlet for all introductory computer books, but Osborne is lavish with examples and does not hesitate to leave large amounts of white space on the page in the interests of visual and conceptual clarity.

Things flow easily from arithmetic operations—how you can subtract binary numbers by *adding* them—to a description of octal and hexadecimal representation of binary numbers. The next thing the reader knows, he is understanding how binary math and status flags are used to perform logical operations.

At this point, the reader is ready to journey beyond the portals of the CPU into the land of addresses and registers. First, several haunting questions about the different types of computer languages are answered. The characteristics of machine language, assemblers, compilers and interpreters are detailed and the discussion moves rapidly on to microcomputer functional logic. Before long the reader is following data and instructions to and from memory and between registers in the CPU. At one point he may even do a double take when he realizes he's been reading a timing diagram and understanding the various operations. At that point one is apt to look back and be surprised at the level of sophistication one has achieved.

There is, of course, much more to be learned and to the credit of *Volume 0* it does not attempt to cover it all. Rather, it presents general concepts in a way that gives the reader confidence and a sense of achievement. At certain points the author points out that what is to follow may not be of interest to everyone and those who wish may skip to a different part of the book. All in all *Volume 0* is an excellent introduction and one I highly recommend.

Reviewed by Tom Williams.



PERIODICAL GUIDE
FOR COMPUTERISTS
E Berg Publications, 1360 SW 199th Court
Aloha, OR 97005
60 pp, $5.00

This is the third collection (I think) in an on-going effort to provide pointers to articles from 25 hobby and professional electronic and computer publications. It's a very handy booklet if you do reference work or if you just never can quite remember which periodical contained that article you just *must* find. Articles, editorials, book reviews and relevant readers' letters are indexed by subject under 100 categories ranging from Altair 680 and Amateur Radio through Clubs, Education, Time-sharing, and Video Displays.

The price tag of $5.00 may seem a bit steep for 60 pages, but the time saved by having such a reference handy makes it worth it.

Reviewed by Christine Anne Brunet.

THE SCHREIR SOFTWARE INDEX
TO PUBLISHED MICROCOMPUTER
SOFTWARE
Volume 1, Number 1
The SSI, 4327 East Grove Street,
Phoenix, AZ 85040.
47 pp, $5.00

This booklet is a useful reference for microcomputer enthusiasts who are interested in locating published software in 130 categories from Address and Aircraft through Business, Golf, Graphics, Tickertape and Type Setting. While the majority of published software won't directly solve most problems, it's nice to at least know where to look for implementation ideas. The majority of listed programs appear to be games in BASIC—I'm making that assumption based on the fact that non-BASIC programs are annotated (eg 8800, 8080, 8008, 6800, 6502).

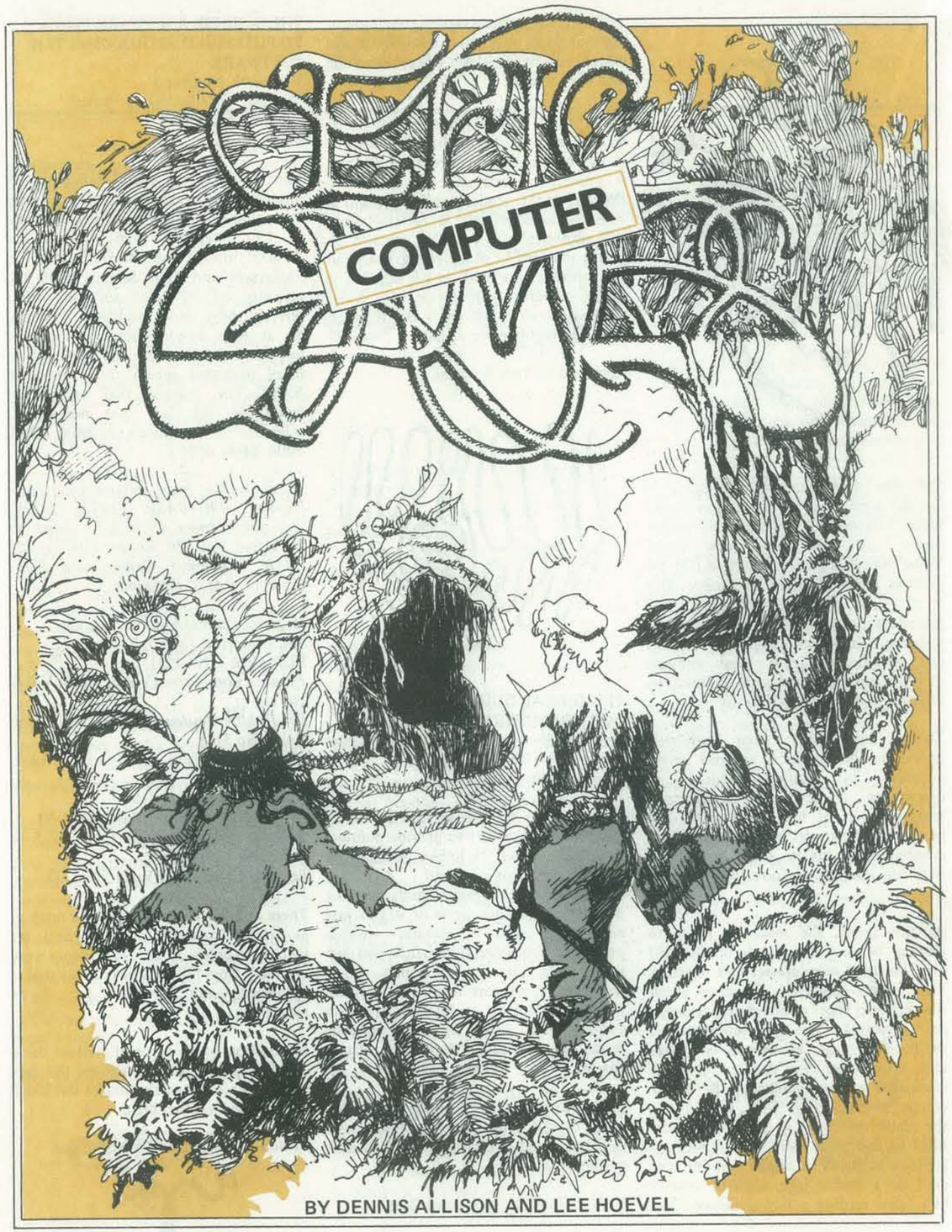
These books were referenced: *Game Playing with BASIC* (1977), *BASIC Software Library* (Vol 1-5), *What to Do After You Hit Return* (1977), *101 BASIC Computer Games*, and *Some Common BASIC Programs*.

Magazines indexed in this issue are
SCCS Interface (Vol 5, No 1-10, 12)
ROM (Vol 1, No 1-3)
People's Computers (Vol 5, No 6; Vol 6, No 1)
Dr Dobb's Journal (Vol 1, No 1-10; Vol 2, No 3-6)
Interface Age (Vol 1, No 9-12; Vol 2, No 1-10)
Kilobaud (Vol 1, No 1-9)
Byte (Vol 1, No 1-16; Vol 2, No 1-9)
Creative Computing (Vol 2, No 1-2, 5-6; Vol 3, No 1-5)
Personal Computing (Vol 1, No 1-3).

There appears to be no particular method as to which issues were selected for inclusion—unless it was what issues were on hand. Hopefully the next Index that is published won't have such gaps in its references.

In addition to 47 pages of content there are 5 pages of ads. Somehow the fact that paid ads are carried makes the $5.00 price tag seem rather high.

Reviewed by Christine Anne Brunet.



BY DENNIS ALLISON AND LEE HOEVEL

the topics covered in this article may well entice and lure you into Adventure mania. It's not improbable that readers will spend long hours finding out how to access games like Adventure and Zort on local time-sharing systems or even trying to implement versions of such games on micros.

Dennis Allison, an active long-time supporter of ours, is a computer consultant in the San Francisco area. Lee Hoevel is a graduate student at Stanford University who is just about to receive his Ph.D. in Computer Science. Neither author, nor the editor, nor the publisher accepts responsibility for addictions stemming from this article.

INTRODUCTION

Few themes are more common in literature than the epic adventure. The questing adventurer–who overcomes tremendous odds to achieve his purpose and saves the princess, discovers the holy grail, finds the golden fleece, or destroys the rings of ancient evil–is an almost universal story line, common to all cultures and peoples. Even the pulp novel, be it an adventure saga, a murder mystery, spy thriller, or science fiction extravaganza, owes its structure and character at least in part to the epic.

Games based loosely on this idea have been around for a long time. In an earlier issue (Volume 5, Number 2) we reported on the game Dungeons and Dragons which is, in some sense, an Epic Game. But it is not (yet) a computer game.

Recently, a computer game has appeared which is something like Dungeons and Dragons. It is called Adventure, and is available only for large machines with FORTRAN compilers, large amounts of memory, and on-line disk storage. The original form was written by Willie Crowther (now at the Xerox Palo Alto Research Center) and greatly modified by Don Woods at Stanford University's Artificial Intelligence Laboratory. The version we have played was on the DEC System 20 at Stanford's LOTS (Low Overhead Time Sharing).

Computer games are intrinsically boring. It is very hard to get passionately involved with guessing a number the computer has picked or finding the one stable solution to a highly oversimplified economic model which keeps asking for that same data over and over again. Occasionally, a game of Chess is interesting, but most games just don't hold one's interest. It is hard to get involved with the output of a random number generator. But Adventure is different. It is not a game, *it is an adventure!*

WHAT ADVENTURE IS LIKE

In Adventure the computer plays two roles. First it is an intelligent companion that provides an interface to the game-world for you. It is your eyes, hands, and ears. You *tell* the computer what you want to do–explore, kill, take, or whatever–and it *responds*, informing you of the consequences of your actions. There is none of the mechanical interrogation characteristic of so many computer games:

DO YOU WANT TO PLAY ANOTHER GAME? (0 FOR NO, 1 FOR YES) ?

Second, the computer is both a protagonist and an active antagonist. It manipulates several hostile entities–dwarfs and pirates–that impede your search for knowledge and treasure. It also controls certain forces of nature, some of which are entirely predictable, and some of which are probabilistic. These two roles are kept distinct, however; the intelligent companion never lies (although he sometimes doesn't tell all he knows right away–you have to beg for hints), nor does he pass information to your enemies as to what you know or what tools you have.

The universe of Adventure is a forest and a cave. There is treasure in the cave; the player must find the treasure and escape the many dangers. The world in which the game is played is created with wit and humor as well as cleverness.

Our experience with Adventure was very positive. It got us to thinking about just how such games could be constructed. The ideas behind the game were even more general than, one suspects, the authors supposed. With a little thought one could construct a very general framework for many different and unique games.

After playing Adventure, we had the opportunity to look at the code which implements the game. It is a small FORTRAN program (about 74 pages of listing, 3600 lines), with game structure pretty much equally divided between tables and procedure. The implementation is brute force. Even a cursory inspection shows that FORTRAN is not particularly suited to games implementation–the reason for using it is equally obvious, though: portability (of sorts). The usual problems associated with string and character-oriented processing with a numerical analysis language are manifest.

This program is, unfortunately, specific to the game of Adventure. One cannot factor out the world and characters of Adventure and replace them with others to make a new game. While some of the program is table-driven, most actions are treated as special cases. A lot of testing is done–is this true, then do this, if that, then do that, and so forth–in the main line of the game. This makes the game hard to modify and repair; changes are not easily localized.

What was most surprising was that Adventure is so complex and satisfying to play yet its internal structure is reasonably simple. We cannot resist speculating on how more complex program structures could be generated, and how much more enjoyable the resulting games would be. . .

WHY EPIC GAMES ARE FUN

The game designer needs to know something about why games are fun–the mechanics and psychology of gaming. Fundamentally, we believe that games of lasting interest must be so complex that it is not possible to devise a deterministic winning strategy. This requires an underlying set of 'natural laws' as well as an element of *deus ex machina*. While purely mathematical games have the most consistent structure, games based on mathematical laws are generally of less lasting interest than those that simulate some features of real life.

Players must be actively involved. They must be called upon to make decisions which affect the course of the game and participate with their imagination; one's

fantasies are important and should be nurtured. There should be conflicts and some means of conflict resolution (running away should be OK, if one is fleet of foot!). It is both the exercise of power and the battle itself that is important.

There must also be some measure of wit and humor; humorless games are dull. The best games have a curious mixture of discovery and invention, the familiar and the unfamiliar. A good game has both variety and complexity. Experience in playing should not make the game dull either by repetition or by advantage.

And games must relate, in some way, to ritual. Perhaps secular ritual, but ritual nonetheless. Everyone understands a large number of traditional symbols, stories, relationships. Trite they may be, but they are the substance of all stories. And a good game exploits them.

The real attraction, for us, is not in the winning, but in the adventure of discovery. One can appreciate a good game in much the same way as one appreciates a novel, a film, or a play. And buried within the game are some new and perhaps useful insights into the structure of our world and lives.

The most interesting games, then, exploit:

- Player involvement
- Wish Fulfillment
- Conflict Resolution
- Discovery and Invention

Variety is the spice in each of these areas. Player involvement is achieved, and interest maintained, by giving the computer personality–witty rejoinders, a sense of humor, and an occasional change of phraseology are essential elements of an Epic Game. Wish fulfillment can be approached in at least two different ways. The nature of a game itself may be a filter–if you are playing a simulation game based on the Mayan conquest of Central America, it may be assumed that you are already interested in the history and culture of the Mayans (or their victims). Alternatively, an Epic Game may provide a rich enough variety of actions, places, and objects for the player to construct personalized experiences. Adventure takes this latter approach; there are a number of ways to explore the cave and lots of different things to do (although if you want to score all the points you must visit specific places, perform specific actions with specific objects, etc.). Both strategies work because people tend to do things they like best when playing games.

Variety in conflicts and conflict resolutions is also important. There should be many weapons available to the adventurer –as well as many avenues of escape–so that at least some judicious selection is involved. Otherwise, the game deteriorates into the apocryphal automation that merely prints:

YOU WIN
YOU LOSE
YOU LOSE
YOU WIN
⋮

Including the joy of discovery in a game is a real test of the game designer. Most games start out by asking if the human player wants an explanation of some sort. This is understandable, since in the typical game you can't tell a klingon from a klaxton without a program! You really need to know that 1 means 'move', 2 means 'fire', 3 means 'look', and so forth. In contrast, discovering what words 'work' in Adventure is almost as much fun as discovering treasure!

Vocabulary is only part of it, though. The constraints on behavior are better left unstated until an illegal (or stupid) action is attempted. Only after trying a number of ineffective combinations should a player be able to deduce the underlying natural laws involved. If this approach is taken, it is important to keep the game's laws similar to the familiar natural laws of the real world–or consistent with the particular mythos that forms the basis of a fantasy game. It is inappropriate, for example, for a hero to be able to pass through a wall (instead of going over it or around it) without finding the secret passage or knowing the magic word.

One could go even one step further, by providing a conscience for the player through the game's interpreter. Thus, a knight would find it difficult to kill a damsel whereas easy to kill a dragon. Similarly, a pirate could steal the gold, but an honest sailor would find it harder.

Such laws–game rules–must be internally consistent. An opening big enough for a dragon to pass through should also be big enough for a dwarf to pass through. Also, just as scientists are constantly invalidating or modifying the 'laws of physics' as our model of the world becomes more accurate, so should the adventurer have to modify his theories of cause and effect as new information is accumulated.

The best laws for an Epic Game are those with multiple levels of interpretation, each of which is 'nearly' right and is thus usable as a 'rule of thumb'–but these are also the hardest to define. Alternatively, an Epic Game could synthesize new laws when all the existing game rules have been discovered (actually changing rules in midstream is not really fair, even in a fantasy game). This might be done by keeping track of things the player has tried in previous situations. Weizenbaum's Eliza program uses such an embedded state scheme to simulate intelligent behavior by changing the subject of conversation to something touched on previously.

Just as nature itself is constantly revealing new laws, so an Epic Game should constantly be able to expose the adventurer to new cause-effect relations.

SOME OF HOW TO DO IT

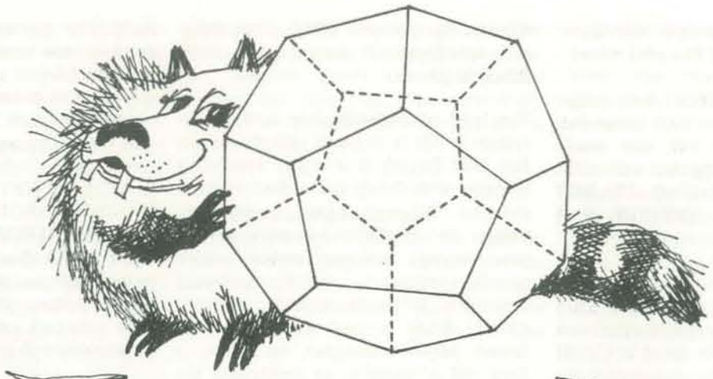
In thinking about how one might implement these games, one can identify a number of distinct problem areas: representing natural laws, and representing players. The intent here is not to provide a cookbook of how to construct games, but rather to catalog some of the possibilities.

In considering the problems of implementation we have to touch on many different areas including the representation of a game universe on the machine, natural language understanding, scripting the game itself, some issues of asychronous processes, to mention but a few. In the last section some of the possible models for games are considered.

Representing the World

The world for all Epic Games is a collection of places. Associated with each place are a number of things. Places have names, they have histories (the battle of Foobah was fought here long ago), there are movable objects present (and other people are movable objects), to name just a few.

Places don't exist alone, they are connected to other places. The connection may



be direct–by road or train–or indirect by airplane or even hyper-space warp.

The underlying model of the structure of the game universe is that of a graph. Each node on the graph corresponds to a place, each connecting arc corresponds to a connection from one place to another. One might, for example, use the secret passage from the castle, under the moat, to the humble peasant's cottage. But maybe one needs to know the magic word to get there.

The game Wumpus, originally developed by Greg Yob and published in our pages way back in 1973, uses a graph for its universe. Each room of the cave is a vertex of a dodecahedron. But it lacks all the interesting features–there are no names, no history, nothing but a list of vertices. And there are only twelve places. A real epic universe must have many more than that!

There is a substantial analogy between the game graph and the board used to play games like Clue and Monopoly. Here the board defines the game-world topology or geography. But in these games the possibilities are far more limited than in an Epic Game. In the traditional board game, each move corresponds to one step in time; this need not be the case in an Epic Game.

In a complex game, both the nodes (places) and the arcs (connections) must be labeled. Just what the labeling is will depend upon the details of the particular universe, but might include some of the following:

Place Nodes:
- a description of the place
- a name of the place
- the list of connections to other places
- the history of the place
- a list of objects at this place
- a list of people (players) at this place

Connection Labels:
- the target place
- the accessing mechanisms (e.g. must have a magic word or own an airplane)

Most everything here has the fundamental data structure of a list of lists. Labels don't need to be values (that is, numbers); they can be the names of procedures. If marked correctly, they can be invoked to accomplish some purpose which furthers the game.

It's not necessary to have a world with only a single level of detail. One could have a representation in which each node in the graph at one level corresponds to an entire graph. This hierarchical structuring allows the game designer to concentrate on what is structurally important at each level. For example, in a space exploration game, the levels of representation might consist of the galaxy, the star system, the planet, the landing site(s), the ice mountain, the cave inside the mountain, the hall of ice ants, and so forth, each representing a refinement of the other. It's all a matter of scale.

Teaching the Game to Speak English

There are few things quite so *dumb* as having some game repetitively ask a question like:

WHAT DO YOU WANT TO DO:
MOVE (0); SHOOT (1); HIDE (2);
QUIT (3)?

Such mechanical response contributes to the irritation and frustration associated with most games. We want to be able to tell the computer just what we want it to do and not have it ask inane questions. Unfortunately, this is very very difficult to implement. Programs that 'understand' English are on the edges of contemporary research in Artificial Intelligence. Fortunately, most of what we need for simple games is not really that sophisticated.

At the very simplest, we can simply require most game commands to be imperative statements of the form:

<verb> [<object>]

where the angle brackets delimit the part of speech and the square brackets indicate that the <object> is optional. Commands in this class might include such things as RUN, EAT FOOD, PLAY AGAIN (AGAIN is not really an object, but it could be treated as one by a game-playing program). Even this small concession to the human being can enliven a game; we greatly enjoyed trying out different words and word combinations while learning Adventure, just to see what they would do. The actual

vocabulary in this game is only about 300 words, but it *seems* like a lot more!

It is up to the semantics of the parser to sort out and disallow such commands as EAT SHOES. To wit, one would expect the parser to respond with some clever rejoinder. For example, I'M NOT HUNGRY, THANKS ANYHOW or I PREFER SOUL FOOD.

A more complicated system might be patterned after the Eliza program. Eliza is an AI program due to Joe Weizenbaum at MIT and described in detail in *CACM* (*Communications of the Association for Computing Machinery*) 9:1, January 1966. A revised version is to be found in *CACM* 10:8, August 1967, pages 474-480. Bert Raphael's *Thinking Computer*, and Weizenbaum's own *Computer Power and Human Reason* contain interesting discussions of the problem.

Unlike programs which try to understand language based upon some model of how language functions, Eliza-like systems are based upon a simple keyword pattern match. The choice of pattern is based upon a hierarchy of keywords.

One implementation of a natural language processor of this sort might work in the following way. As the input sentence is entered, each word is looked up in a dictionary and replaced by a pointer to the dictionary entry. Synonyms may map to the same dictionary attributes. At the same time, a priority queue of important words (keywords) is maintained. Whether a word is important may depend upon when it is said in the game, but mostly it will be independent of that.

Associated with each word is a possibly empty list of patterns, actions and rewrite rules. Beginning with the most important word and continuing until the priority queue is empty (in which case a default response is applied) dictionary entries are consulted for the head of the pattern-action-rewrite list. Each pattern is tried. If there is a match, the action is performed, the rewrite rule applied, and the next game cycle started. If there is no match, then the next rule for this keyword is tried. If there are no more rules, then the rule set for the next keyword is tried.

It is important to observe that scripts are contextual objects and any truly effective system will utilize a hierarchy of scripts keyed off circumstances rather than a single one.

The level of understanding in the Eliza system is nil; it depends entirely on the fact that English is a highly redundant language with firmly entrenched sentence patterns. Whereas Eliza attempts to mimic an intelligent *conversation*, a game-playing program would attach specific actions to specific sentential constructs, so as to simulate an *interaction. When a pattern matches the human player's dialogue*, its action is fired off in addition to developing the program's output–i.e., the witty rejoinder. In some ways this is more appealing since it would give the greatest semblance of an intelligent computer. But as all such systems go, one must sometimes accept (and produce) absolute gibberish without complaint.

Representing the Player(s)

Each participant in the game has his own descriptor which describes himself and his station in life. Like most other game elements, it is nothing but a list of lists. One might generalize and consider a participant to have:

a name
capabilities (magic, leaps tall buildings, etc.)
objects (sword, food, money, etc.)
goals (which may interact with those of other participants)
a location (where he is now)
an activity (what he is doing at the moment)

Just how all these are interpreted depends upon the nature of the particular Epic Game.

Fundamental to understanding how an Epic Game might function is the observation that a participant need not be a human player. One can perfectly well implement a program having a dragon participant who simply wanders about the world at random looking for beautiful young virgins. Presumably, beauty and virginity are attributes that would be carried in the 'descriptor' for other players, and would determine the course of action a hungry and highly irritable dragon might take.

One would want to provide some mechanization of a strategy–motivation, really–for the dragon participant. For example, one could cause it to seek out beautiful virgins and/or knights (preferably wounded knights) in order to eat/battle with them. One such mechanization is English conversation with a human game player (HOW ABOUT SOME LUNCH, HONEY? or YOU'VE BEEN HANGING AROUND TOO LONG–IT'S TIME I TAUGHT YOU A LESSON!). There could (hardware willing) also be more than one player in a 'universe' at the same time; they could interact not only with each other but with computer-implemented phantom players.

While some participants are certain to be implemented via a mechanical process, there should always be some element of *deus ex machina* to see that time runs smoothly and to introduce random events of the kind usually attributed to acts of God.

Participants (human or not) should be able to communicate with each other. Not only must there be dealings between the human player and his aide(s), but other messages should be possible in the manner of stage directions:

(VOICE IN THE DISTANCE):
Help! Help! I'm drowning!

Scripting

Just how to represent knowledge is one of the fundamental problems of Artificial Intelligence research, and remains pretty much unsolved in the general case. The issue of representing knowledge is also basic to Epic Games–we'll discuss the subject in terms of how to prepare a game's script.

There are several different scripts which contribute to the nature of an Epic Game. These scripts may be incorporated into the fundamental structure of the program itself or may be data bases which are executed or interpreted.

One script must provide the set of natural laws of the universe in which the game occurs. If the game has bouncing balls, then one might expect the laws of physics to be in the game. And only wooden stakes kill Dracula.

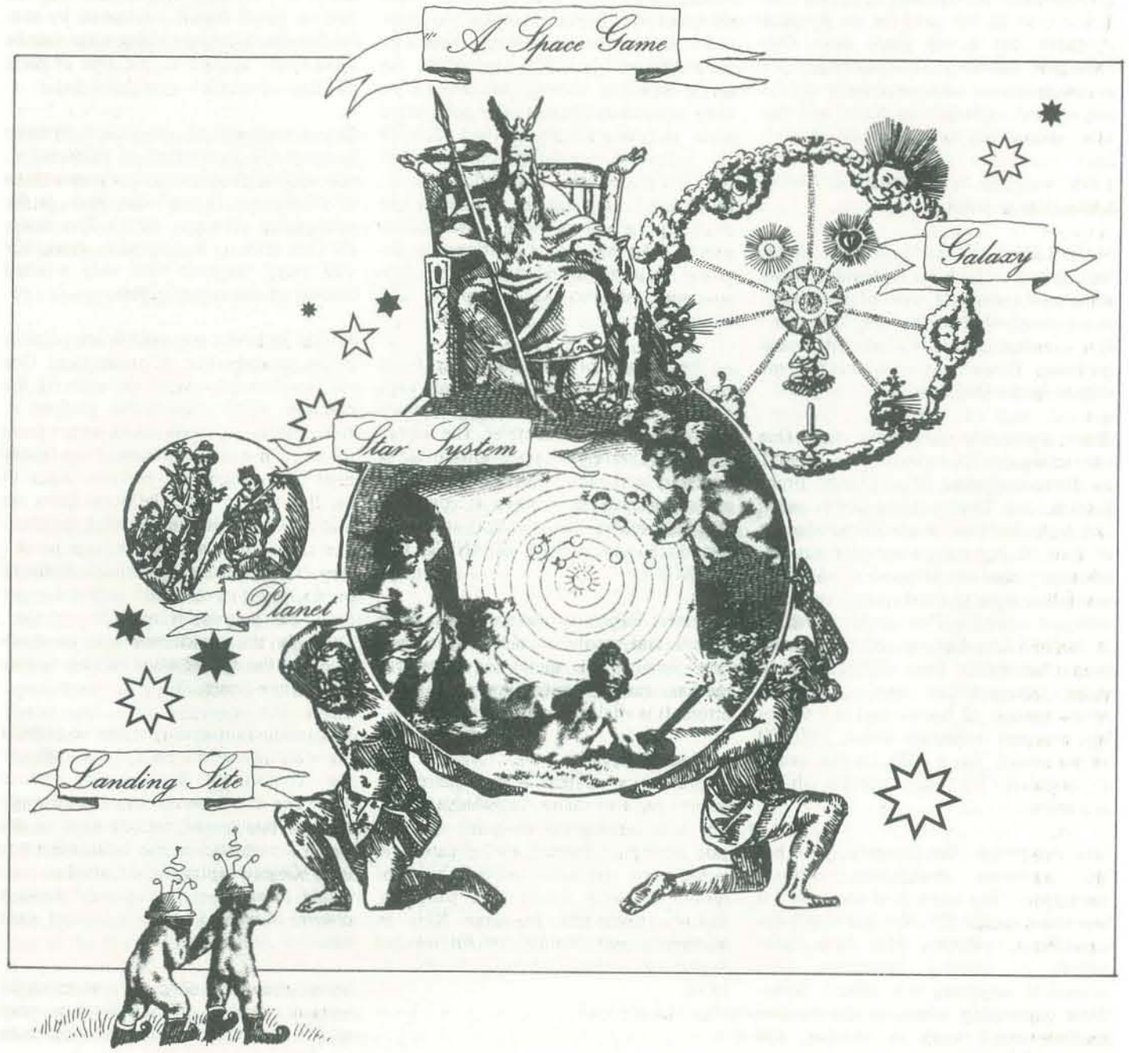
Scripts are also the mechanism for developing characters. The idea of using an Eliza-like language structure is, in part, attractive because of the ease with which one can construct a variety of different characters. Each character corresponds to a script. It is important that the script can be constructed by someone more interested in the structure of the game than the way computers are programmed. Eliza scripts have this property.

Individual objects and players are themselves scripted. There is a lot to be drawn from the ideas of Transactional Analysis here (see Eric Berne's *Games People Play*). The motivations of many gameplayers are rather trivial. The dragon wants to survive and hoard treasure. The knight is out to kill dragons. Foxes eat chickens, bad wolves eat little pigs. The exobiologist wants to communicate with the aliens. And so forth.

Confrontations and Contests

One cannot expect the player to really do battle with a dragon in a game, yet the contest should be determined by something more deterministic than the throw of a random number generator. One approach might be to use subgames. One could use a computer number-guessing game like Stars, a more traditional game like Tic-Tac-Toe, or even a riddle contest.

The black knight encounters the white knight in the forest.
Let us battle says the black knight.
Can you answer this riddle:
What did the pig say when caught by the tail?
with the authorized response:
That's the end of me.



One could also include some measure of strength, much in the manner of Dungeons and Dragons. Characters have intrinsic strength based upon their makeup, and gain strength as time passes and deeds are successfully performed. Time mends wounds and adds to strength. Doing appropriate deeds, (good or bad, depending upon the role) also adds to strength. This index of strength might be used to determine the difficulty of the chosen confrontation contest.

MORE ON GAME-BUILDING

The authors of Adventure suggest that the problem in building a game like this is not in the program or program structure, but in the game itself. One must plan carefully to avoid bugs. One must anticipate the responses of the players and capitalize on them, and one must insure that the game itself is internally consistent. Adventure took only a few weeks to build initially. It's taken a long time to refine it.

FORTRAN is not the language of choice for games. It lacks adequate data structures, adequate control structures, string manipulation features, and manifest constants, to mention but a few problems. However, it is portable to some extent. Hence the choice.

Other games like Adventure exist. One very elaborate one was done at MIT by Tim Anderson, Marc Blank, Bruce Daniels, and Dave Lebling and is called Dungeons, but runs under the pseudonym of Zork. It has a more complex natural-language user interface, a different geometry, more internal complexity, and different surprises. This version is written in MUDDLE, a language which is much more hospitable than FORTRAN to string manipulation and procedural representation of knowledge. It's a very big program requiring about 120,000 36-bit words on a PDP-10—not really a candidate for easy transportability to a micro.

Like Adventure, the Dungeon game has had problems maintaining internal consistency. The universe of the game has been built on the fly. New game features (containers, vehicles, etc) have been added to existing structures with occasional surprising side effects. Sometimes something which works in one context won't work in another, and there is no known rational reason why this should be so, given the universe and natural laws of the game.

Will Crowther is creating a new version of Adventure which is driven off an 'English-like' description of the game itself. He doesn't feel he will be able to totally excise all game-dependent special-case code, but he will be able to remove a substantial portion of it. Unfortunately, this work is being done on a machine which is not generally available.

Tools for Game-Making

To keep game-making problems within bounds, you need a good set of tools. One cannot expect a gamemaker to memorize all the various encodings (how is he to remember that 876 means 'in the castle with the bear' or whatever?). To keep everything straight, the gamemaker must fabricate programs which develop the necessary encodings from the input descriptions. These descriptions themselves are a language for describing and declaring the objects which constitute the game. There are strong relationships between these tools and such system programming standbys as compilers and assemblers.

In the absence of the resources to build tools, one can make do rather nicely with a macro assembler or a macro preprocessor followed by an assembler. The former is usually preferable since arithmetic is necessary and most macro preprocessors are enormously slow when it comes to addition, subtraction, multiplication and division to say nothing of AND's OR's and NOT's.

The most important feature is that game objects and attributes can be described symbolically. This allows for easier generation, easier maintenance, and fewer errors. It is vital to good gamesmanship.

Production Systems

A production system is an organization scheme used in many 'knowledge based' Artificial Intelligence programs to provide their fundamental mechanization. A rather nice overview of this kind of system is to be found in a paper by Randy David and Jonathan King in *Machine Representations of Knowledge*, Reidel Publishing Company, Dordrecht, 1976.

The basic idea is to encode knowledge as a set of productions or patterns which, when matched against the data base containing the state of the system, invoke a set of rules. These rules perform external actions and modify the data base. The data base is repeatedly examined to find those data elements to which the productions may be applied. They are applied, and the cycle is then repeated.

Production systems have been used to considerable advantage in programs which must reach conclusions based upon inexact reasoning. A medical consulting program, MYCIN, has been written at Stanford University to provide consultation on blood disease treatments by antibiotics. Production systems may also be successfully applied to the area of game playing.

The advantages of production systems lie primarily in locality of information, and uniform and compact representation of knowledge. (Knowledge, here, is the relationship between data.) The entire MYCIN system, for example, uses only 400 rules. Systems with only a small number of rules are plausible.

Several different approaches are possible in the interpretation of productions. One can, for example, search the rules for the first one which is applicable, perform it, and continue. Or one could restart from the top. Or one could mark all applicable rules and effectively perform them in parallel. Or. . . well, the possibilities are many. Observe, if you will, that the Eliza-like approach to natural language processing is, in fact, a specialized form of production system. Which is best for our particular purpose is open to question. Certainly the choice needs to be made prior to the specification of the various rules of the games.

The production system serves to provide the structure of the universe and specify the relationships between the various data. The data base contains the information relating to the current state of the world represented by the production system. Objects (players) of similar type would share all or part of a production system but would have individual data bases.

To see how a production system might work, consider the following very simple system, which follows closely an example in a Carnegie-Mellon Computer Science Department Report by McDermott, Newell, and Moore, *The Efficiency of Certain Production System Implementations*. This production system counts down from five to zero and then blasts-off whenever the word 'ready' is input. Now, in a game implementation, ready would actually be a variable and blast-off would probably be a procedure, but the idea is the same. The production system for this is:

```
p1: 0 → say (blast-off); stop;
p2: <dig> → say (<dig>);
            <dig> = <dig> – 1;
p3: ready → 5;
```

where all productions have the form:

```
label: conditions → action; action; . . .
```

and a lot of the mechanization is implicit rather than explicit. Looking at the applicable part of the data base (as other things might be going on due to other productions) one might see the sequence as performed in Figure 1.

The corresponding output would be

```
. . .
. . .
5
4
3
2
1
0
blast-off
```

Production systems are notoriously slow because they involve a great deal of searching and data manipulation to create and maintain the information required to make deductions. The smaller and more specialized a production system, the faster and more responsive the system will be. Thus, a word of caution to implementors.

Storing Messages

In an Epic Game of any complexity, one of the major overheads will be the storage necessary to hold all the possible variations of messages. There are several ways one can encode the messages to keep the size of the space required within bounds.

The number of different words used in conversation is relatively small, perhaps a few thousand words. The vocabulary for game conversation and messages is even smaller. Let's presume that we can make do with a vocabulary of 1024 words and punctuation marks. That would, assuming an average word size of five characters, require about 5K bytes of memory for table storage.

One approach would be to simply store all the words in memory. We could use the high-order bit to flag the last character of a word. We would need something like 13-bits to represent the beginning of any arbitrary word.

But that's somewhat wasteful. Most messages use only a few very common words. We could use a frequency encoding where short codes are used for frequent words and longer codes for words which don't appear as often. Accepting the byte-oriented nature of current machines, we could use the following algorithm.

Words are represented by either one- or two-byte quantities. To find the word associated with an encoded value, examine the first byte. If it is greater than 127, that is, it has its high-order bit on, then it represents one of the high frequency words. We subtract 128 from it and find the beginning address of the associated word in the string table in another table, CWTAB. If it is less than 128 then it is the high-order byte of a two-byte quantity which is, itself, the address of the word relative to the start of the string table. All strings in the string

	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	ready	. . .	. . .	. . .	. . .	. . .	. . .
[apply p3]	ready	5	. . .	. . .	. . .	. . .	. . .
[apply p2]	ready	5	4	. . .	. . .	. . .	. . .
[apply p2]	ready	5	4	3	. . .	. . .	. . .
[apply p2]	ready	5	4	3	2		
[apply p2]	ready	5	4	3	2	1	
[apply p2]	ready	5	4	3	2	1	0
[apply p1]	ready	5	4	3	2	1	0

Example of Production System Sequence
Figure 1

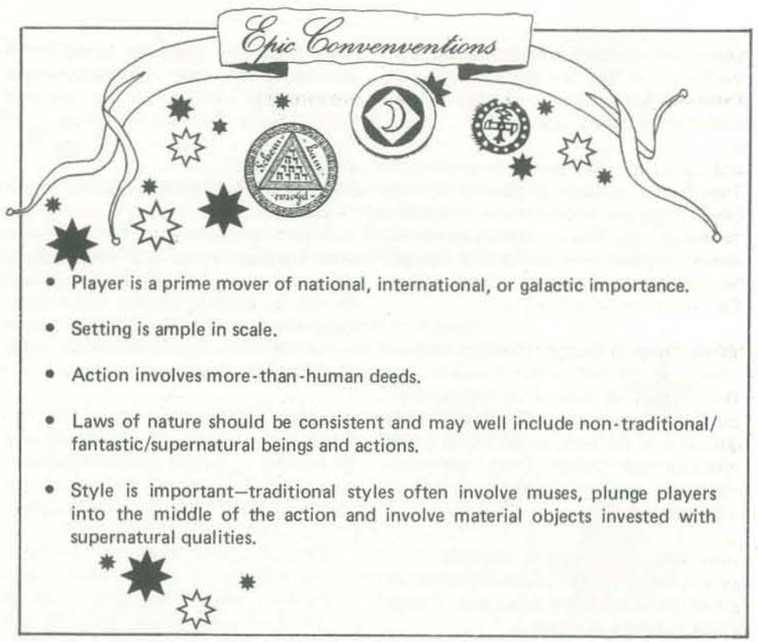


table are terminated by having the high-order bit of their last character on. Note that there is nothing to prevent a phrase from being treated as a word.

The auxiliary table for the frequent words implies an overhead of 256 bytes, since two bytes are necessary to represent a string address. This overhead is acceptable since we break even (except for the code to mechanize the encoding) when there are two or more references.

Such things as capitalization can be introduced by the routine which does the output by keeping track of the beginning and ending of sentences. Line spacing can be handled in the same manner with a line-feed/carriage return being introduced whenever the line would be too long for the terminal.

Consider the following example, a bit contrived. All the various words are maintained in a single string area. It might look as shown in Figure 2.

The top line in the Figure 2 indicates addresses in the array; the bottom line indicates those characters with their high order bit on, to indicate the last character of a word. For common words, we need a secondary array to tell us where to begin. That array, call it CWDS, might look like this:

```
CWDS [0] =   0   a
CWDS [1] =   1   the
CWDS [2] =   4   is
```

and so forth...

The message, 'games are beautiful', would have the encoding:

9	2 bytes	games
134 (6 + 128)	1 byte	are
38	2 bytes	beautiful

for a total message storage requirement of four bytes. Even if words appear only once, this storage technique has only a two-byte overhead. While one could embed such words directly in the text using an 'escape' byte, the saving would be only one byte. The mechanism required to implement the scheme could possibly dominate the savings.

```
12345678901234567890123456789012345678901234567890123456789...
atheisaregamesfrogspigsdogssummerapplebeautiful.....
!..!.!..!....!....!...!...!.....!....!........!....
```

Example of (Contrived) Word Storage Scheme
Figure 2

LIMITATIONS OF PRESENT TECHNOLOGY

It is not without reason that Adventure and Dungeons have used simple static geometries and reasonably passive actions, though far more interactive and realistic than other computer games. These games are about at the limit of what can realistically be built at the present time.

The Electric Novel

Sometime in the future, however, it may be possible to have a computer game be something like a novel, but a participatory one. Imagine the following scenario:

> When the game begins you are being interrogated by one of the nastier members of the homicide squad of the XYZ city police. He's all but accusing you of murder. He lets loose of a few cogent facts, and then you are saved from jail by his boss who comes in and tells him to release you.
>
> You are now free to leave. Your job (to complete the game) is to clear yourself of suspicion and find out who really committed the crime. In doing this you can roam about the city, talk with various people who knew people concerned or were witnesses, discover various pieces of physical evidence, get into conflicts with various other characters, and in general play the role of the detective.

Within the game world, the detective can move about, discover clues, interrogate witnesses, and reach conclusions. And, of course, there could be the usual complement of attacks, beautiful young blondes (male or female depending upon the sex and preferences of the player), and perceptive ideas. Sounds wonderful from the player's point of view. But consider the author, who creates an enormously complex world. If there are witnesses, each one must be scripted to respond to fairly arbitrary questioning. In addition, an underlying plot and subplots must be created in much the same way an author of a novel does, but in a far more complete form.

Practical Considerations

There is one certainty—a game of any complexity needs to have a lot of memory and a lot of mass storage. It is not practical to even consider a Tiny Epic Game as anything more than an example. The purpose of an Epic Game is to be complex enough that even its creator cannot muddle through with ease. It is not merely that the range of possible actions should be large; indeed, if there are *too* many complexities (objects, players—'pieces' on the board), the game will cease to be a recreation. What we mean here is that the *consequences* of interesting actions should be neither trivial nor unlearnable.

The approach here has been to push what might be called 'Existentialism in Game Design' to the limits. Objects are created which are instances of more general prototypes. The mechanization of the game depends upon actions common to all similar objects. Only one set of actions (the game's 'natural laws') need then be provided.

Whenever possible the system should encode data and use interpretive functional execution. Text to be output should be stored as pointers to a word table rather than the direct character representation. Actions for each participant are encoded in a script which contains knowledge about how that participant behaves in particular situations. It is in the creation of an appropriate script that the skill of the games designer is most needed—and apparent!

MAKING IT EVEN BETTER

One can think of the obvious ways to make the whole system look even better. If one could draw pictures of the present place, if one could give commands by voice, or listen to the other participants respond verbally (or scream as appropriate), then the game would be less intellectual, more exciting and more addictive. □

Game Ideas

Star Explorer. A three-level universe—star systems, planetary systems, planets. Planets may or may not be inhabited by creatures that may or may not be malevolent. Each planet provides a different problem for the player to solve. Points are awarded for surviving, discovering, and solving problems. Sort of an active, less combat-oriented Startrek.

Cybership. You are the human brain that controls an intergalactic space ship. You and your human partner (played by the computer in this role reversal game) work as partners to survive space exploration missions.

Close Encounters of the Fourth Kind. Learn how to communicate with an alien—hopefully you can figure out how to do so in time to save yourself/planet/ship from misadventure.

Historical Simulations. You are a prime mover in a historical situation. You can interact with events and, perhaps, change the course of history.

Watergate. You can be Nixon, Dean, or even Senator Ervin; the question to be answered in this game is 'what are the rules?'

Spy in the Night. You are an intelligence agent who must prevent the assassination of the President. You are in New York (London, Tokyo, Madrid) with little to go on. You must search out the truth and get there before it is too late.

Detective. You are a private detective. A murder has happened. You have been hired to solve the crime.

Master Criminal. You are to rob the Bank of Paris and escape. The world is Paris and its famous sewers. You have a certain amount of capital, but no plans.

Oz. Dorothy calls the shots here, but you can try your hand at being a scarecrow, tin man, or even a cowardly lion. The wicked witches are already (type)-cast.

Voyages of Sinbad. Be the hero and do his mighty tasks—if you can.

Lord of The Rings. You too can be Frodo Baggins, Sauron, or any number of other interesting characters.

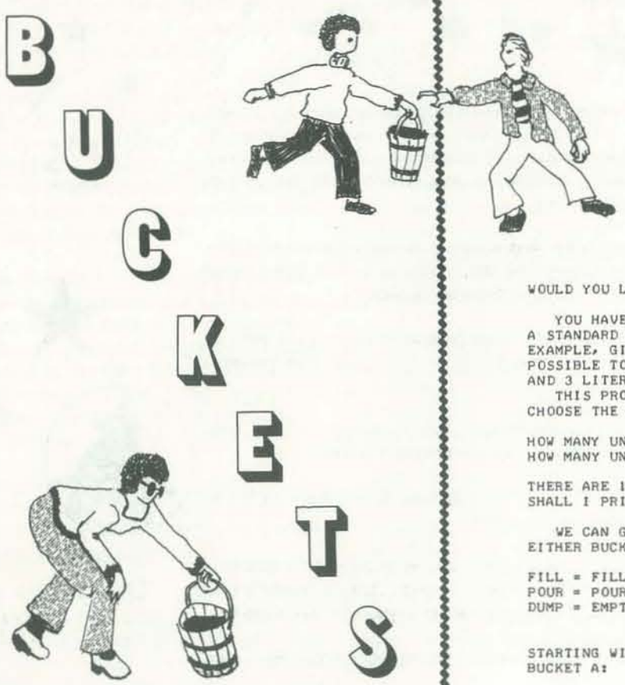
Tales of King Arthur. The usual joust and maiden scene with opportunities to slay dragons, both literary and firebreathing.

The Aeneid. Be the son of Anchises and Aphrodite, the defender of Troy, and captain of the Argos. Virgil has set down a number of interesting feats for you to perform, but you can be sure the game designer has included a few new wrinkles.

Travel into Hell. A travelogue visit to Dante's Inferno with suitable updating. Perhaps not so much blind verse.

The Search for Dracula. See if you can track down the monster before daybreak; an expanded version could be multi-level as in the Startrek game, with a global chase across Europe and several more detailed playing arenas in London, Transylvania, etc.

BUCKETS



BY MAC OGLESBY

Can you measure 2 liters of water if you only have a 4 liter and a 7 liter bucket? Problems of this type are familiar to most of us. Here's a program from Mac that produces all possible outcomes for such two-bucket problems.

Notice that line 990 in the program uses G$ to design the format in which the possible outcomes are to be printed. '< # # #' indicates a 4-character string; '– # #' indicates a number up to 3 digits long. To adapt this program to BASICs without formatting capabilities, additional code is needed to replace lines 990-1000 and neatly print results in a table. □

```
WOULD YOU LIKE INSTRUCTIONS? YES

   YOU HAVE TWO EMPTY BUCKETS AND A SOURCE OF WATER (A RIVER).
A STANDARD PROBLEM IS TO MEASURE OUT SPECIFIED AMOUNTS.  FOR
EXAMPLE, GIVEN A 4 LITER BUCKET AND A 7 LITER BUCKET, IS IT
POSSIBLE TO END UP WITH JUST 2 LITERS OF WATER IN ONE BUCKET
AND 3 LITERS OF WATER IN THE OTHER?  IF SO, HOW?
   THIS PROGRAM SOLVES PROBLEMS OF THIS GENERAL TYPE.  YOU
CHOOSE THE SIZE OF EACH BUCKET.

HOW MANY UNITS DOES BUCKET 'A' HOLD? 3
HOW MANY UNITS DOES BUCKET 'B' HOLD? 5

THERE ARE 16 POSSIBLE OUTCOMES FOR THESE BUCKETS.
SHALL I PRINT THEM? YES

   WE CAN GET ALL OF THE POSSIBLE OUTCOMES BY STARTING WITH
EITHER BUCKET.  THERE ARE THREE OPERATIONS INVOLVED:

FILL = FILL THE STARTING BUCKET FROM THE RIVER
POUR = POUR INTO THE OTHER BUCKET
DUMP = EMPTY THE OTHER BUCKET INTO THE RIVER.

STARTING WITH          STARTING WITH
BUCKET A:              BUCKET B:

OPER-   RESULT         OPER-   RESULT
ATION    A   B         ATION    A   B
-----  --- ---         -----  --- ---
                         0   0           0   0
 FILL    3   0          FILL    0   5
 POUR    0   3          POUR    3   2
 FILL    3   3          DUMP    0   2
 POUR    1   5          POUR    2   0
 DUMP    1   0          FILL    2   5
 POUR    0   1          POUR    3   4
 FILL    3   1          DUMP    0   4
 POUR    0   4          POUR    3   1
 FILL    3   4          DUMP    0   1
 POUR    2   5          POUR    1   0
 DUMP    2   0          FILL    1   5
 POUR    0   2          POUR    3   3
 FILL    3   2          DUMP    0   3
 POUR    0   5          POUR    3   0
 FILL    3   5          FILL    3   5

WOULD YOU LIKE TO WORK WITH DIFFERENT BUCKETS? YES

HOW MANY UNITS DOES BUCKET 'A' HOLD? 1200
PLEASE TYPE A WHOLE NUMBER FROM 1 TO 999.
HOW MANY UNITS DOES BUCKET 'A' HOLD? 57
HOW MANY UNITS DOES BUCKET 'B' HOLD? 997

THERE ARE 2108 POSSIBLE OUTCOMES FOR THESE BUCKETS.
SHALL I PRINT THEM? YES

STARTING WITH          STARTING WITH
BUCKET A:              BUCKET B:

OPER-   RESULT         OPER-   RESULT
ATION    A   B         ATION    A   B
-----  --- ---         -----  --- ---
                         0   0           0   0
 FILL   57   0          FILL    0 997
 POUR    0  57          POUR   57 940
 FILL   57  57          DUMP    0 940
 POUR    0 114          POUR   57 883
 FILL   57 114          DUMP    0 883
 POUR    0 171          POUR   57 826
 FILL   57 171          DUMP    0 826
 POUR    0 228
```


```
100 ' NAME:  ELEMLIB***BUCKETS
110 '
120 ' BY:  MAC OGLESBY ON 31 JUL 75.
130 '
140 ' DESCRIPTION:  GIVEN TWO EMPTY BUCKETS AND A SOURCE OF WATER,
150 '   A STANDARD PROBLEM IS TO MEASURE OUT SPECIFIED AMOUNTS OF
160 '   WATER.  THIS PROGRAM SOLVES PROBLEMS OF THIS GENERAL TYPE.
170 '
180 ' INSTRUCTIONS:  TYPE "RUN" FOR COMPLETE INSTRUCTIONS.
190 '
200 ' REFERENCE:  "PUZZLES & PARADOXES" BY T. H. O'BEIRNE (OXFORD
210 '   UNIVERSITY PRESS, 1965)
220 '
230 '
240 '
250 PRINT "WOULD YOU LIKE INSTRUCTIONS";
260 LINPUT A$
270 IF SEG$(A$+1,1)<>"Y" THEN 300
280   GOSUB 1150
290   GOTO 310
300 LET Q=1
310 PRINT
320 PRINT "HOW MANY UNITS DOES BUCKET 'A' HOLD";
330 INPUT A(1)
340 IF (999-A(1))*(A(1)-1)<0 THEN 370
350   IF A(1)<>INT(A(1)) THEN 370
360   GOTO 390
370 PRINT "PLEASE TYPE A WHOLE NUMBER FROM 1 TO 999."
380 GOTO 320
390 PRINT "HOW MANY UNITS DOES BUCKET 'B' HOLD";
400 INPUT B(1)
410 IF (999-B(1))*(B(1)-1)<0 THEN 440
420   IF B(1)<>INT(B(1)) THEN 440
430   GOTO 480
440 PRINT "PLEASE TYPE A WHOLE NUMBER FROM 1 TO 999."
450 GOTO 390
460 '
470 '
480 GOSUB 1070                          'GET GREATEST COMMON FACTOR
490 LET T=(2*(A(1)+B(1)))/F             'T IS HOW MANY DIFFERENT OUTCOMES
500 PRINT
510 PRINT "THERE ARE";T;"POSSIBLE OUTCOMES FOR THESE BUCKETS."
520 PRINT "SHALL I PRINT THEM";
530 LINPUT A$
540 IF SEG$(A$+1,1)<>"Y" THEN 580
550   IF Q=1 THEN 570
560   GOSUB 1260                        'GO PRINT EXPLANATION OF LISTS
570   GOSUB 690                         'GO COMPUTE AND PRINT OUTCOMES
580 PRINT
590 PRINT "WOULD YOU LIKE TO WORK WITH DIFFERENT BUCKETS";
600 LINPUT A$
610 IF SEG$(A$+1,1)<>"Y" THEN 630       'GO GET DIFFERENT SIZES
620   GOTO 310
630 PRINT
640 PRINT "GOODBYE FOR NOW..."
650 STOP
660 '
670 '
680 ' COMPUTE AND PRINT ALL POSSIBLE OUTCOMES
690   LET R(2)=A(1)                     ' INITIALIZATION FOR THE 2ND LIST,
700   LET A(2)=B(1)                     '   WHICH STARTS WITH BUCKET B
710   LET R(1)=R(2)=C(1)=C(2)=0         '  RESET LOCAL VARIABLES
720   PRINT
730   PRINT
740   PRINT "STARTING WITH";TAB(20);"STARTING WITH"
750   PRINT "BUCKET A:";TAB(20);"BUCKET B:"
760   PRINT
770   PRINT "OPER-   RESULT";TAB(20);"OPER-   RESULT"
780   PRINT "ATION    A   B";TAB(20);"ATION    A   B"
790   PRINT "-----  --- ---";TAB(20);"-----  --- ---"
800   PRINT "          0   0";TAB(20);"          0   0"
810   FOR J=1 TO T-1                    '(FIRST OUTCOME ALREADY PRINTED)
820     FOR K=1 TO 2                    'COMPUTE OUTCOMES FOR THE 2 LISTS
830       IF C(K)=0 THEN 960            'SEE IF STARTING BUCKET IS EMPTY
840       IF R(K)=B(K) THEN 930         'SEE IF OTHER BUCKET IS FULL
850         LET R(K)=R(K)+C(K)          'POUR STARTING BUCKET INTO OTHER
860         IF R(K)>B(K) THEN 890       'CHECK FOR OVERFILLING OF OTHER
870           LET C(K)=0                'STARTING BUCKET NOW EMPTY
880           GOTO 910
890           LET C(K)=R(K)-B(K)        'STARTING BUCKET KEEPS EXCESS
900           LET R(K)=B(K)             'OTHER BUCKET NOW FULL
910           LET D$(K)="POUR"          'SPECIFY POUR AS OPERATION
920           GOTO 980
930         LET D$(K)="DUMP"            'SPECIFY DUMP AS OPERATION
940         LET R(K)=0                  'OTHER BUCKET NOW EMPTY
950         GOTO 980
960       LET D$(K)="FILL"              'SPECIFY FILL AS OPERATION
970       LET C(K)=A(K)                 'FILL STARTING BUCKET
980     NEXT K
990     LET G$=" <### -## -##  <### -## -##"
1000      PRINT USING G$,D$(1),C(1),R(1),D$(2),R(2),C(2)
1010    NEXT J
1020    PRINT
1030    RETURN
1040  '
1050  '
1060  ' FIND THE GREATEST COMMON FACTOR
1070  FOR F=B(1) TO 1 STEP -1
1080    IF A(1)/F<>INT(A(1)/F) THEN 1110
1090      IF B(1)/F<>INT(B(1)/F) THEN 1110
1100        GOTO 1120                   'F IS GCF
1110    NEXT F
1120    RETURN
1130  '
1140  ' INSTRUCTIONS
1150    PRINT
1160    PRINT "   YOU HAVE TWO EMPTY BUCKETS AND A SOURCE OF WATER (A RIVER)."
1170    PRINT "A STANDARD PROBLEM IS TO MEASURE OUT SPECIFIED AMOUNTS.  FOR"
1180    PRINT "EXAMPLE, GIVEN A 4 LITER BUCKET AND A 7 LITER BUCKET, IS IT"
1190    PRINT "POSSIBLE TO END UP WITH JUST 2 LITERS OF WATER IN ONE BUCKET"
1200    PRINT "AND 3 LITERS OF WATER IN THE OTHER?  IF SO, HOW?"
1210    PRINT "   THIS PROGRAM SOLVES PROBLEMS OF THIS GENERAL TYPE.  YOU"
1220    PRINT "CHOOSE THE SIZE OF EACH BUCKET."
1230    RETURN
1240  '
1250  ' EXPLAIN THE LISTS
1260    PRINT
1270    PRINT "   WE CAN GET ALL OF THE POSSIBLE OUTCOMES BY STARTING WITH"
1280    PRINT "EITHER BUCKET.  THERE ARE THREE OPERATIONS INVOLVED:"
1290    PRINT
1300    PRINT "FILL = FILL THE STARTING BUCKET FROM THE RIVER"
1310    PRINT "POUR = POUR INTO THE OTHER BUCKET"
1320    PRINT "DUMP = EMPTY THE OTHER BUCKET INTO THE RIVER."
1330    LET Q=1
1340    RETURN
1350  '
1360  END
```

BYTE INDUSTRIES

NELS WINKLESS TELLS WHAT'S HAPPENING

Byte Shop™ has become a synonym for 'computer store' in some areas. In fact, the numerous stores are licensees of Byte Industries, Inc, one of the early entrepreneurial endeavors to foster the computer hobbyist movement. 'Byte Shop' is trademarked by Byte Industries.

During the first part of 1977, stories of Byte's cash flow problems were circulating. Until September, disaster seemed imminent–then rumors of capital backing for Byte were heard.

Here's the story of what's been happening, from one of the new faces at Byte Industries, Nels Winkless, former editor of Personal Computing. *Chuck Bradley interviewed Nels for* People's Computers *in December.*

QUESTION: Nels, can you tell me what your present relationship with the Byte organization is?

To start with, the organization I'm associated with is Byte Industries, Inc–formerly Byte Inc. Obviously, it needs to be distinguished from *Byte* magazine and its publishers, Byte Publications. My relationship is that of consultant to the people who just bought Byte Industries. I've spent a year and a fraction rummaging around in the personal computer business to see what's going on and see who the people are, and the new owners of Byte Industries thought I might be useful in getting the company reorganized. I'm simply an outsider coming in to look for problems and do something helpful where I can. But I report to John Peers of Logical Machine Corporation, the new owner of Byte Industries.

QUESTION: Could you briefly describe the Byte organization?

I'll give you a touch of history. A couple of years ago, Paul Terrell was an electronics rep; among other things, he was selling Altairs, made by MITS. He and Boyd Wilson watched Dick Heiser open a computer store and said, 'Gee, if Dick can do that, I bet we could run a store too.' They set up a computer store in Mountain View, California, and promptly realized that since computer stores were springing up like weeds, a real distribution operation was needed. They got a good legal grip on the name Byte Shop and began to set up Byte Shops around the country. Their real business is getting merchandise in one door from the manufacturers and pushing it out the other door to the retailer–but this is not a franchise operation. What they did was license the use of the name to a store owner and offer to sell their merchandise. But of course they did national advertising for the name and are still doing it. As they went along, they began, defensively, to build some of their own equipment, and so on. So that's really the structure. They're a distribution operation.

They did very well although they never had any capital. They set up new stores, did some product development, did some literature development, built a staff, did the promotion and so on, all out of current sales. They got gorgeous national publicity–presidents of major companies began dropping in to chat. Felt like the big time. And on they went without capital. But they just flat could not manage to do that indefinitely without any money. So although they had pretty decent sales, it began to catch up with them. And sales began to taper off and stores began to buy around them.

It was a cash flow problem. It is hard to think of anything in business that is not a cash flow problem. They couldn't pay the people who were selling them merchandise and they couldn't deliver stuff. So by midsummer in '77, things were getting pretty tough. And Paul was doing his quaint native dance searching for capital, and trying to keep a pretty bold front.

Paul went to call on John Peers and said 'How'd you like to buy control of something?' And John said, 'Sure, why didn't you come in sooner?' And away they went. By the end of September it was actually done. Lomac (Logical Machines Corporation) acquired the company 100% outright. They have plugged in a modest amount of capital, a few hundred thousand dollars, but not the rumored millions.

QUESTION: John Peers, President of Lomac and Chairman of the Board of Byte Industries, has said he plans to make computers so easy to use they can be sold 'like motorcars.' Does this mean he will provide his 'natural' programming software concepts for personal computers the way it's done for Lomac's Adam computer?

The answer is yes. I don't know that it will be that particular software, but it certainly will be the concept–that pervades everything. John really believes it, it isn't just lip service, and that's partly why working with him is so much fun. He has a very broad, sweeping concept of information handling and the interaction of individuals. He talks about, for example, personal access to information that has not already been collected and put aside where it can be reached. He thinks in terms of instant access to a machine that says, 'Oh, that's what you want, I'll go find it.' The machine turns information up when you want it and gives it to you. That's very different from most information systems. That's an active searching system.

**. . . there's really no industry yet–
just a band of terrified people.**

Peers is dedicated to the proposition that a computer ought to be useful to people. He would like naive users to get at the machines and make them do something helpful without a professional intermediary.

The new president of Byte Industries is Jerry Brandt, a consultant whose specialty is coming to companies that need help in doing whatever it takes to get them moving. Jerry came in and stirred things up and in October they shipped probably twice what they had done the previous month. And in November I think they shipped about twice that. We are now in December and things are really rolling.

Jerry describes himself as an energizer. He's not much on formal structure, but he is big on getting everyone excited and interested and moving, and the change in that establishment is incredible. Morale was really pretty bad. Now all of a sudden, there are people coming in at night so they can get some work done on their own, and–they're really moving.

Paul Terrell is still part of the operation; he's looking for the place in which he can help best. I believe Byte Industries has sold the two stores that it owned; they want to be in the distribution business instead of the retailing business.

QUESTION: How does Byte Industries supply its retail Byte Shops? Is it by some kind of formal contract, or what?

There is a formal agreement of some kind, whose details I have not studied yet. It is on demand at the dealer's option. I don't know if dealers are compelled to buy anthing from Byte–it's just that it is easier. You know, one stop shopping, instead of dealing with a hundred different suppliers. But Byte Industries is planning to put together a franchise package. They'll make it a little bit different from other such packages–really do the right kind of job. They are hoping to have a thousand stores instead of the 60 or 70 which they have now–and do it in style.

QUESTION: In what way might Byte help the absolute novice buying a computer system?

To talk about that, I have to say something about my views. I have to assume that the reason I'm involved at Byte/Lomac is that people there are in sympathy with my views, else they wouldn't have me hang around.

A couple of points. First, personal computing is computing a *person* can do, as contrasted with an *institution.* I don't care if it's a terminal hung on an IBM 370/165. If a private individual can make that thing do what he wants–that's personal computing. A second point: There are a whole lot more people who don't know about computers than do. And if you address yourself only to those people who know something about computers, you're deliberately limiting your market. So you've got to be able to get at the novices. It is crazy not to.

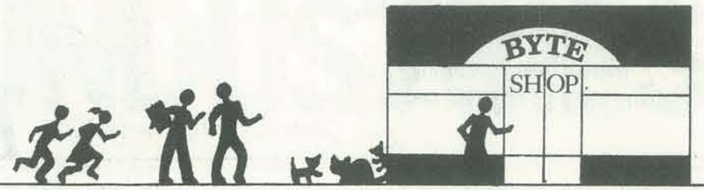
The issue we're dealing with here is not technology. People in general absolutely don't give a fig for technology per se. But they *are* interested in *personal freedom*. There was a time when we had many choo-choo trains that carried people, and if you wanted to go where the railroad tracks went when some remote bureaucrat said you could and if you wanted to obey the rules, you could get a ride on that train. Now, the instant automobiles were available, people bought cars and disposed of that remote bureaucrat telling them where they could go when.

That's what personal computers are about: personal freedom. I think people have been intimidated by computers for 30 years. Now I don't think people are using these computers to do anything *useful* in terms of utilitarian, immediate application. I think 95% of the time people are still horsing around with them, and finding out what they have to put up with from the remote bureaucrat. When they find out, they're going to begin to make their own judgments about what freedoms they want. I think this is a significant cultural revolution, and what we're dealing with is the kind of issue people get excited about, chop off heads over, you know–riot, rebellion and carrying on.

How does the Byte Shop help that kind of thing? It makes it possible for the guy to come in and do that sort of fooling around. The Byte Shops, like all computer stores, have varied immensely. You know, the people who are running these shops are independent; and certainly they don't accept orders from Byte Industries Inc. The guys out in the field now are pioneers, they're on their own, they don't take any foolishness from anybody as they cut a path through the wilderness. So the Shops are very different from each other; some of them won't touch a hobbyist with a ten foot pole. I've been in places where the guy says, 'I don't ever want to see a hobbyist in here again' and they take steps to drive them out. They can't stand to sit around and gab with them. And there are other dealers that say, 'Oh, gosh, I love that. . . oh boy, those guys keep me alive. They come in here and they buy a board and they buy a chip and I don't know, if it weren't for them I couldn't make it.' What Byte Shops need to do is to evolve so as to make it possible for anyone with any interest at every level to come in at that level and do whatever he wants to with information handling equipment. That's what the Byte Shops can do.

QUESTION: Do you expect Byte Industries to ultimately become a full-service organization? By a full-service organiza-

Mass marketers are going to sell video games and create the situation in which people want personal computing.



tion I mean a set up to provide all hardware, software and documentation for a rather wide variety of users and applications. Full service would also include, I think, that when a customer has problems he has a place to go where he can get some help, whether the problem be hardware or software or perhaps even that he needs some additional training.

Flatly, yes. Absolutely. Byte Industries must deal with those issues. Computers and their peripherals have to be as reliable, as predictable and as well supported as Sears' appliances and as easy to understand. I may be in hock to Sears Roebuck for the rest of my life and perhaps even in an afterworld! I think they hope I will pay up before I go on–they may have to sue the estate but one of the reasons I always deal with Sears is that Sears is always there, and whatever it is, they've got it. If Sears doesn't sell it, I don't own it, and so on.

QUESTION: In view of some of the industry's problems, such as long delivery schedules and faulty equipment, can you tell me what kind of guarantee might be offered to customers to protect them against such deficiencies?

No, I can't. Partly because, as you know, personal computing is just forming. Let me share with you Jerry Brandt's view of the field based on his first computer show, the Personal Computing show in Chicago. There were probably 4,000 people in the place, and he came out and said, 'That place is an energy field, it's in there just pulsating. . . with *anxiety*. The manufacturers there are all terrified, and the dealers are all terrified, and the people who are running it are terrified, and the customers are all terrified. They're all afraid that they're going to miss something, that they're going to do something wrong, that it is all a mirage, that they're buying the wrong thing and they'll be sorry tomorrow, that they're not buying the right thing and they'll be sorry tomorrow, or something. . .'

The field is totally immature. It is not an industry. It has never yet paid any attention to the standard structure in our society that let such industries work in the long run. There is no standard distribution system. A kid who has been selling memory boards over a card table and making $8,000 in a single weekend gains an illusion that, 'Why, this is easy', so he sets up a company and finds pretty soon that the next step he takes he can't do, because what he did in the first place was invest some time and money and his time and money are already gone.

Now what? Well, most of the manufacturers in the field have been selling directly to the customer, because they can make more money than by selling to a distributor. People have been selling at 10% margin–that kind of thing can't last. Manufacturers' sales costs are eating them up, and they're out dealing with the customers in onesies and twosies–it's killing them. What Jerry did was take a look at the process in action and say, 'Wait, wait, this whole thing has got to be reformed. We're going to try to ease this.' And that's largely the function of Byte Industries.

To some extent Byte's functions address that question, 'What about service, what about a guarantee?' Obviously it's a thing that *must* be dealt with, and it's going to be. But remember, those Byte Shop guys out there are mean, tough individualists. They cannot be compelled by headquarters to do *any*thing. However, headquarters can make it easy for them to do the right things and help them to make a little bit more money and take a lot of this curse off them. And gradually lead the whole thing along. But there's really no industry yet–just a band of terrified people.

QUESTION: How do you expect the Association of Computer Retailers to improve the retailing function?

My suspicion is that the Association of Computer Retailers, which is the one that Portia Isaacson is associated with, will survive because Portia is the best leader and shaper around. I don't know what effect they're going to have, just because I don't know enough about that kind of thing. I suppose they can be useful if they relieve some of the terror. It's the same problem. They're all afraid of each other, they're afraid of the customers and so on. If they can get together and figure out what they're doing, if it gives them the chance to consult with each other, it's probably a helpful thing.

I suppose that what the leaders of any group like that typically aspire to is power–I am not applying this to Portia. I think that there are people at the top of most of these associations who are on a power hunt, and I don't know how much power they're going to have. They're going to find that it is as hard to discipline their membership, as Byte finds it to discipline the Byte Shops. There's *no* discipline.

QUESTION: Do you expect the Byte organization to make any inroads towards resolving the problem of standards for components?

Well, obviously, you can't impose a standard, but I expect Byte to be very much concerned with that. I think they're going to have people working on it and giving papers and participating on panels, and doing whatever they can to

Computers and their peripherals have to be reliable, as predictable and as well supported as Sears' appliances and as easy to understand.

sort it out. Of course it's in the interest of the stores, because you hate selling somebody something that doesn't work; it comes back to haunt you, and if you can't tell why it's not working, it makes it harder still. There are no standards. But there have to be standards. Or else we'll be sorry.

QUESTION: How might the impact of large retailers, such as Sears and Roebuck, affect Byte's position in the market?

I think it'll help. The large retailers cannot handle a broad line, they can't learn enough about the field to serve all purposes. I'm for having the customers naive, but you can't have the stores naive, because *somebody's* got to know what's going on. I think it would be fine if Sears gets in with stuff like the PET or whatever the equivalents are, but I doubt that they are going to be able to be all things to all people in this field–that's really the function of the Byte Shops.

I think that computers are going to be in the same class as refrigerators and television sets. As a matter of fact, I think video games are going to be the key for getting into this market. The games are going to turn into computers; the large retailers who sell video games are making a mass market, *not* the computer industry.

Television had to be developed for *entertainment* so that a lot of people *wanted* it. Mass marketers like Sears and so forth, are going to sell video games and create the situation in which people *want* personal computing. We're doing a lot of pioneering, but even a few hundred computer stores is a drop in the bucket.

QUESTION: Do you expect any large microprocessor manufacturers to get into the personal computing market, perhaps the way they did with digital watches?

Yes. A lot of them took a terrific bath in digital watches and hated the whole thing. Consider Commodore and National Semiconductor, as examples. Texas Instruments is doing well, and I think that TI is the one company I expected to get into this field. I don't think that when the big guys get in they're just going to mash the poor little guys. *Everybody* in the business is amateurish, and I mean when National Semiconductor gets into it, they are amateurs, because what they know about the semiconductor business and about the established computer business doesn't apply to this at all. And I don't think the microprocessor manufacturers learned enough from digital watches to tell them a lot about selling computers to the populace at large.

I haven't the faintest idea when I expect several big companies to do something, although obviously Generous (sic) Electric company is not going to walk away from something that looks like a real consumer market and Zenith and RCA are not going to ignore a real consumer market. Texas Instruments is not really a household word: GE is, Zenith is, Motorola is. That's where the excitement is going to be, in the long run. And Lomac is going to be there.

Incidently, never underestimate IBM. I'm not sure they want to get into this end of the business, but among their 280,000 employees worldwide they've got a bunch of smart ones, and they do some intelligent stuff, and they have the resources to do pretty much what they choose. I went recently to an IBM press seminar and I was greatly stirred by it. Some of their people were awfully smart and interesting and intelligent and a lot of fun and I was impressed. It strips away your standard image of IBM. I don't think they're going to ignore this thing. I don't think they're going to stop being a force. I think that's true of some other companies too, such as General Motors. General Motors owns Frigidaire and they sell refrigerators and washing machines, why not personal computers?

QUESTION: How do you expect the manufacturers to change their products in the next year?

I don't know about the next year, but I expect there will be excitement at the national Computer Conference (NCC), the major national professional computer show. It's being held in Anaheim, California about June 6. It's the big show. The same kind of rumors that we were hearing about what would happen last year have had another year to mature–maybe something will happen. I kind of expect bubble memory to become a reality. Everybody keeps looking over at Texas Instruments and saying 'What's going to happen there?' I am not aware of any dramatic technical breakthroughs, I just see more and better–chips are bigger, speed is picking up. I think prices on neat computers like the Apple are going to be tumbling–that's the major thing.

Incidentally, at NCC and elsewhere, people who are paying taxes and paying phone bills at last get a chance to look at our computer stuff. It wasn't that the computer industry unbent and said, 'Come on in, we really love you'–they didn't. People battered down the doors and fought their way in, and Portia opened up the personal computing thing at NCC in Dallas last June, and Jim Warren is doing the same with his second and third Computer Faires in March and November. I don't know that the next year will bring anything dramatic, but I think the field will keep moving forward very rapidly. □

FORTRAN MAN

BY LEE SCHNEIDER & TODD VOROS

Adventure 3, Episode 1

Yes, dear readers, once again it is time to put down those programming pencils (the ones with no erasers left on them), unplug those soldering irons, and turn off that paper-tape reader, and indulge in yet another unavoidable episode of the one, the only, FORTRAN MAN!

Faster than a speeding DO loop, more powerful than a two-pass assembler; can leap tall matrices in a single DIMENSION! Wherever there are wrongs to be righted, errors to be decoded, problems to be solved. . . Fortran Man will be there, engaged in the endless struggle for Truth, Justice, and the Algorithmic Way!

As today's story opens, Our Hero has just recently returned from a lengthy stay in the 'Old Country' of Transistoria, where he met and conquered his latest arch-rival, the evil Count Algol! And now, after the long trip down the I/O channel, his transport comes finally to rest at the port of his beloved home, 360 City. . .

After the customary check through the Gates, F-Man changes his sign bit and branches ashore, back into the heart of the busy 360 City. . .

Aahhh. . . feels good to see that familiar third-generation architecture skyline again!

I/O Ports

And, at last, Fortran Man arrives back once again at his resident location. . .

Ahhh. . . home at last!

But then, as the dynamic fighter of programmatic evil branches into his resident location, he discovers a little surprise waiting for him!

Wha. . .???

Who are *you*, and what are you doing here????

LDA?

MOV?

Fortran Man can see immediately that the thing is harmless. . . still, he seems to have some difficulty in decoding its actions. . .

Hmmm. . . seems to be speaking in Microcode. . . and in a dialect that I'm not at all familiar with!

LHLD?

INR?

DCX?

What I'm going to need is an INTERPRETER. . . . . and I think I know just the program for the job!

Moving with his usual fast execution time, F-Man makes a quick CALL to the local DMA control office. . . and, within a few milliseconds, his young friend and fellow crime-fighter Billy Basic is linked to him at his location. . .

Welcome back, F-Man!

What's up?

Don't quite know yet, Billy. . . take a fast scan of this, and tell me what *you* think!

Billy Basic does a quick examine of the new data which has entered F-Man's location. . .

Yes, this is Microcode it's speaking. . . but it's not really saying anything!

XCHG?

There seems to be some sort of a ROM built into it. . . if only I had a way to get a readout on it!

That's simple, Billy. . . I'll just plug the little fellow into my private data terminal!

F-Man relocates his 'guest' over to the interface section of his code, where the simple link function is performed. . .

Aha. . . it's working!

Here comes the message. . .

There we go. . .

click!

DRAWINGS BY A.MIYA

The text begins to read out from the ROM. . . and as it flashes across the screen, Billy interprets. . .

No sweat, F-Man. . . this is Basic code. . . and if I can't read this, no-one can!

It's a private-line transmission addressed to you, F-Man. . . from Clan McIntel in Micro Land!

The Land of the Little People? What does it say?

And so, Billy reads on. . .

A FEW CYCLES AGO, A DEMENTED NOISEMAKER CALLING HIMSELF 'THE GLITCHMASTER' CAME TO OUR LAND, AND IMMEDIATELY ANNOUNCED HIS INTENTION TO RULE IT!

Gee. . . listen to this!

Hmph. . . I've heard *that* line of code before. . .

WE THOUGHT NOTHING OF THIS THREAT. . . UNTIL WE DISCOVERED THAT HE HAD MANAGED TO PENETRATE OUR FILE SECURITY AND MADE OFF WITH OUR CLAN MASCOT, THE LOCKOUT MONSTER!

Uh-oh. . .

WITH OUR BELOVED MONSTER UNDER HIS CONTROL, THE GLITCHMASTER HAS ALL LINES OF COMMUNICATION BLOCKED. . . WE WERE THEREFORE FORCED TO USE THIS RATHER UNCONVENTIONAL MEANS OF TRANSMITTING OUR MESSAGE!

DON'T KNOW HOW LONG WE CAN HOLD BEFORE OUR DATA ORGANIZATION IS DISSIPATED. . . PLEASE SEND HELP IMMEDIATELY. . . . . END

I think I'm beginning to detect the pattern, Billy!!

HELP!

THE UNDERGROUND RESISTANCE MOVEMENT, LED BY GENERAL WIREWOUND, IS PUTTING UP A VALIANT EFFORT TO PULL DOWN THESE LINES, BYPASS THE CURRENT REGIME AND RESTORE ORDER, BUT WITH LITTLE SUCCESS. . .

Well, F-Man, that's it! Sounds serious!

Hmmmmm. . .

Branching away from the display area, Billy turns and looks expectantly at Our Hero. . .

Sure sounds like they need help in a hurry. . .does this mean. . . ?

Yes, Billy, it does. . . and when Fortran Man receives a CALL for aid, there is only one thing to do. . .

I must relocate myself into Micro Land immediately!

Fortran Man decouples his terminal and leaves the little Micro Messenger to go about on its way. . . and then, with his usual speed, begins to branch furiously about the area, assembling his travelling things. . .

But. . . F-Man! How will you get in? I could go easily, of course. . . it's my homeland. . . but *your* code isn't made to fit into a Micro!

Finally, after a quick assembly, F-Man passes out of his resident location, with a very puzzled Billy Basic close behind him. . .

Don't worry, Billy. . . I've got a plan—and you're a part of it! Come on. . . no time to lose!

And what diabolical adventures await Our Hero this time? Can the Land of the Little People be saved from the clutches of the Glitchmaster? Will the Lockout Monster ever be returned to its rightful owners at Clan McIntel? Tune in again next issue. . . . . same time, same I/O channel!

MICROCOMPUTER COMMUNICATION for the HANDICAPPED

BY TIM SCULLY

This article is more technical than many published in People's Computers, *but we believe that the general discussion will be interesting, informative, and thought provoking to all, even those who choose to skip the program listings and discussion.*

Tim Scully has been designing biofeedback equipment and doing biofeedback research for many years. Tim is a Research Fellow of the Humanistic Psychology Institute; he is now working towards his doctorate in psychology. His dissertation project involves researching and developing biofeedback systems and techniques for use in drug rehabilitation.

Tim is also teaching a computer class to fellow inmates at a Federal penitentiary. Although prison resources are scarce and he is not allowed to solicit donations, he is hopeful of somehow eventually acquiring a computer system for the prison.

The potential of microcomputers as tools for the handicapped is enormous and exciting: we encourage dissemination of such information. For this reason we are making copies of this article available. To receive a reprint, send a stamped, self-addressed envelope (24¢ for business size, 35¢ for 8½ by 11 inch) to People's Computers.

How would you communicate if you couldn't talk, didn't have the use of your hands, and could only somewhat control the movements of one knee? This is the problem which Robin, a young lady in her 20's has lived with all her life. She has cerebral palsy.

I met Robin in 1976, and this is the story of how a microcomputer communication system came to be built for Robin. The general concepts applied in the development of Robin's communication system may prove helpful in the development of microcomputer systems for other handicapped people.

When I first met Robin, her communication was accomplished by use of a word wheel. She could understand speech and she could read, but she needed help in 'talking'. Her word wheel was made from an electric clock motor and a bicycle spoke, with the bicycle spoke attached where the second hand of a clock would normally be mounted. A sheet of cardboard was mounted behind the spoke, with the letters of the alphabet on it, arranged in a circular pattern. The spoke pointed to the letters, one at a time, as it rotated. Robin could move her knee to one side and hit a kneeswitch mounted on her wheelchair, thus stopping the motor so that the spoke would freeze, pointing at the letter she had chosen.

The spoke rotated at one revolution per minute, so spelling proceeded at about one letter per minute! The person Robin was conversing with often had to write the letters down, to keep from forgetting them, as a message slowly built up. To speed up the communication process, a few words were written next to each letter of the alphabet, so that when the spoke stopped it would point at a group of words as well as a letter. The person with whom she was conversing would have to guess which of these Robin intended. It took considerable patience to hold a conversation with Robin, and not very many people took the time.

When I first saw Robin's communication system, I thought of replacing her word wheel with a microcomputer and video display, using a vocabulary of words stored in the computer's memory in place of the sheet of cardboard. A little over a year later, that system now exists and is being installed on Robin's wheelchair.

HOW IT WORKS

The present system is an expansion of the word wheel concept which uses a TV display with 16 lines of text. The top line is reserved for the display of a 'menu' of items (words, letters of the alphabet, punctuation symbols or control codes) from which Robin can choose. The second line is kept blank and the bottom 14 lines provide space for the display of a message of about 200 words.

As items are displayed on the menu, Robin can choose one by hitting the kneeswitch mounted on her wheelchair. In some modes of operation several items will appear on the menu at once, in which case the item at the left is the current item, the one which can be selected by hitting the kneeswitch.

On start-up, the system blanks the TV screen and then offers the SPELLING? mode by putting that word on the menu. This item remains on the menu for a time 'T1' (an adjustable time delay). If the kneeswitch is hit during that time, the SPELLING? mode is entered, otherwise the next menu item is displayed: PUNCTUATION?. If that item isn't chosen either, after another delay equal to T1, then the system will begin displaying the names of groups of words: A-BONE, BOOK-CROWN, CRY-FINGER, FINISH-HIDE, HIGH-LOT, LOUD-OUGHT, OUR-ROSE, ROSE ANN-STAY, SQUARE-TWENTY and TWO-YOURSELF, one group at a time. Each group of words contains about 120 words in alphabetical order. The name of each group is made up from the first and last words in the group.

If Robin doesn't pick any group of words, the computer then offers an ESCAPE? from the groups of words. If this isn't chosen, the names of the groups are offered again. If the ESCAPE? is chosen, the system returns to near the beginning of the program and offers SPELLING? again. This ESCAPE? to the beginning is offered from every mode of system operation.

If Robin does pick a group of words, HIGH-LOT for example, then the names of subgroups in that group begin being displayed, one at a time: HIGH-HONOR, HOPE-HUNT, HURRY-IMPORTANT, IN-INTERESTING, INTO-I'VE, JENNIFER-JUMP, JUST-KISS, KITCHEN-LAKE, LAND-LEAST, LEAVE-LIE, LIFE-LITTLE, LIVE-LOT and then ESCAPE?. If Robin picks a subgroup, such as LEAVE-LIE, then the words in that subgroup are displayed across the top line of the TV, with two spaces between each word:
LEAVE LED LEFT. . . LIBRARY LIE

If Robin hits the switch at this moment, LEAVE will be transferred down to the first available space in the message area of the TV screen and the menu will begin all over again by offering SPELLING?. If the first word, LEAVE, isn't chosen, then after the usual time delay T1, the list of words on the menu will shift one to the left, so that LED is on the extreme left and it becomes the current item. This process continues until a word is chosen or until the end of the subgroup, LIE. If LIE isn't chosen, ESCAPE? is offered, and if it isn't chosen, the complete list of 11 words in the subgroup is displayed across the menu and the cycle begins again.

By this system of groups of words, subgroups, and finally words, it is possible for Robin to look through a list of 1200 words in a short time, find the one she wants and add it to a message she is assembling on the TV screen. The computer automatically adds a space after each word chosen, so it isn't necessary for Robin to worry about spacing between words–she can just choose one word after another. All letters and words are upper case, so she doesn't have to shift.

When a sentence is complete, and when she wants punctuation symbols, Robin can select the PUNCTUATION? mode. The first item offered on entering this mode is CONTROL? and if that isn't chosen, then after the usual time delay, the punctuation symbols will be spread across the menu in much the same way that the words in a subgroup were displayed:
. ' ? ; : ! 0 1 2 . . . 9 # $ % & ( ) * + –

These items leave the screen at the left, one at a time, if they are not chosen. If one is chosen, the computer backspaces once (to undo the automatic spacing) and adds the chosen symbol to the message on the screen. Then the system starts over by offering SPELLING? again.

The CONTROL? mode offers Robin a few useful commands, one at a time, if it is chosen: BACKSPACE?, ERASE LAST WORD?, SPACE?, ERASE SCREEN?, and NEXT LINE?. These control codes operate immediately if selected. Then the system starts over by offering SPELLING? again.

The SPELLING? mode exists to allow Robin to spell words not found in the 1200 word vocabulary stored in the computer's memory. To speed up the process of spelling, letters of the alphabet are not offered in alphabetical order. Instead they are offered in the order of their probability of use in English. Except at the beginning of a word, the likelihood of a letter appearing in a word depends on the last letter chosen.† If we are in the middle of a word, and the last letter chosen was 'A', then the most likely next letter is 'E', the second most likely is 'B', etc.

Robin's system has 27 different alphabets stored in it. The first alphabet has the letters organized so that those most likely to appear at the beginning of a word will be displayed first. This is the alphabet which appears when the SPELLING? mode is first entered. The letters are spread out along the menu line as usual, with the first offering on the left. If no letter has been chosen by the time all of them have moved off the screen to the left, the usual ESCAPE? offering is made and the alphabet redisplays.

If a letter is chosen, it is added to the message area of the screen, and ESCAPE? is offered on the menu. If Robin decides to stay in the spelling mode, the computer then displays one of the 26 remaining alphabets–which one is determined by the letter she just chose. When she picks a letter from this new

† Mr A Ross Eckler suggested the bigram spelling scheme used in Robin's system. He supplied me with letter use frequency tables which he credited to F Pratt, *Secret and Urgent: The Story of Codes and Ciphers*, Blue Ribbon Books, 1942 pp 258-259.

alphabet, it is added to the message, immediately after the first letter (the system automatically backspaces to undo its automatic spacing). This process continues until she has completed spelling a word. Then she picks ESCAPE?, which returns her to the beginning of the program, which offers the SPELLING? mode, and a space is left after the word she has just completed.

This spelling scheme allows comparatively rapid spelling of words because Robin only has to wait for a few letters to display before the one she wants is likely to become the current item. The automatic spacing also speeds up communication.

Now that we've looked at what Robin's system does, let's examine the hardware and software which do the work.

SYSTEM DESIGN

Robin's system was designed around the special limitations of her situation and my own situation. I met Robin through a United States Probation Officer, who was supervising me while I was temporarily free on appeal bond. I was waiting for the Court of Appeals to decide if it would uphold my conviction for conspiracy to manufacture LSD (back in 1968 and 1969). As it turned out, the Court did uphold my conviction, and I'm now serving a 10 year Federal prison term at McNeil Island Penitentiary in Washington.

My personal problems limited the system design to the use of a commercially available computer kit because of the difficulty of sending materials into prison. Robin's family had only a limited budget, and Robin's capabilities formed the remaining design limits.

In 1976, the budget we had (about $1,300) was just about enough to buy a computer kit with keyboard, cassette tape system, video monitor and 8K of memory, so this is the size system we planned on. The average word in English is about 5.5 characters long and we initially planned on a vocabulary of about 1,000 words, which uses up 5,500 bytes of memory. This left about 2,500 bytes for the program to control the system together with storage for spelling and punctuation symbols.

That's not enough memory for the use of a high level language such as BASIC, so the program had to be written in assembly language. Since my previous assembly language experience was with the 8080A, this was the CPU chosen for Robin's system.

We wanted the system to be expandable. In the future, Robin may want to add more memory, a printer, a speech synthesizer or other additional peripherals. For maximum flexibility in expansion, the S-100 bus structure was chosen because of the wide range of commercially available plug-in circuit cards. The computer also had to be small and light enough to mount under the seat of Robin's wheelchair. In order to modify the menu and message areas of the video display independently, the computer needed a memory-mapped video display. These constraints pointed us toward the Polymorphic Systems' Poly 88 System 4 kit.

The Poly 88 uses a 5 slot S-100 chassis, which makes it small and fairly light in weight. The Poly video card is memory mapped and displays 16 lines of 64 characters each–just right for Robin. The features of the Poly CPU card were also useful: it has 512 bytes of RAM together with a monitor program in ROM. A cassette tape interface card works together with tape loading software in the monitor ROM to handle program storage and loading.

The vocabulary for Robin's system is stored in RAM because we expect her vocabulary needs to change once she can communicate more freely. The problem with storing vocabulary in RAM is that RAM is volatile–the memory and thus the vocabulary are erased every time the computer is unplugged. So a battery back-up card was added to the system. This card keeps the program and vocabulary stored in RAM even though the computer may be unplugged for hours at a time while Robin's wheelchair is moved from place to place. Robin's computer uses the Seals Electronics BBUC card with NiCad batteries.

We had, at one point, considered battery powering the entire system, but ended up rejecting the idea. A large and heavy battery would have been required for reasonable life, and this would bring the total weight of the wheelchair and system up so high that Robin's mother wouldn't be able to lift it in and out of their family van for trips to school and other errands. As it is now designed, Robin's system has to be plugged into a wall outlet to operate, but the battery back-up card keeps memory alive while the system is unplugged so that it is instantly ready to start upon being plugged in.

HARDWARE MODIFICATIONS

A few additions and modifications were made to adapt the commercially available hardware to Robin's application. The Poly 88 chassis has only two controls: an on/off switch and a reset pushbutton. This is because it is designed to use a keyboard for functions which a control panel might perform. The reset pushbutton starts the ROM cassette tape loading program. I added a second pushbutton which activates a vectored interrupt and jumps to the beginning of Robin's program. This makes it possible to start up Robin's system without the keyboard. A schematic for this simple addition is shown in Figure 1.

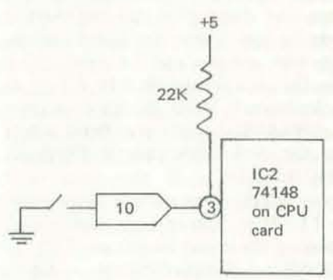


Figure 1

As a computer powers down, it can scramble data stored in memory by sending out false write commands. To eliminate this problem, the memory in Robin's system was partitioned so that an 8K block of RAM, containing the main program and stored vocabulary, could be write protected. This left only the 512 bytes of RAM on the CPU card unprotected (and the memory mapped video display, of course). The small CPU RAM area is used for all scratchpad functions and is one of the features of the Poly CPU card which encouraged its selection.

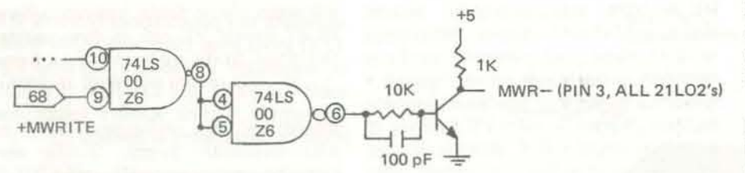


Figure 2

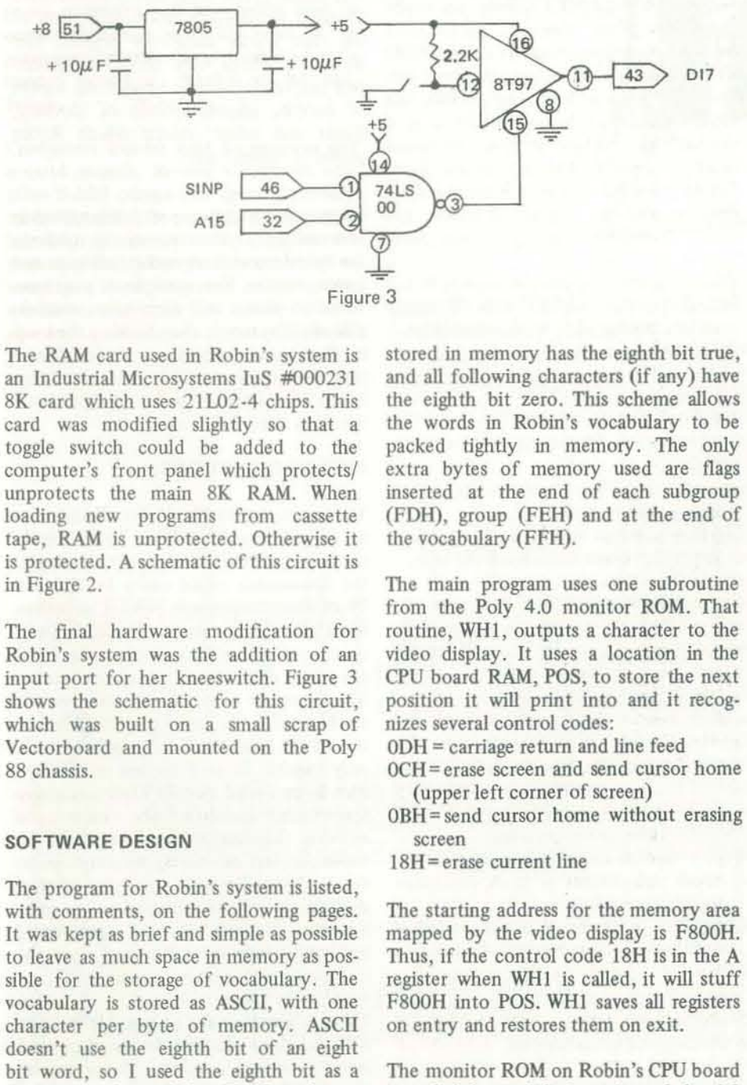


Figure 3

The RAM card used in Robin's system is an Industrial Microsystems IuS #000231 8K card which uses 21L02-4 chips. This card was modified slightly so that a toggle switch could be added to the computer's front panel which protects/unprotects the main 8K RAM. When loading new programs from cassette tape, RAM is unprotected. Otherwise it is protected. A schematic of this circuit is in Figure 2.

The final hardware modification for Robin's system was the addition of an input port for her kneeswitch. Figure 3 shows the schematic for this circuit, which was built on a small scrap of Vectorboard and mounted on the Poly 88 chassis.

SOFTWARE DESIGN

The program for Robin's system is listed, with comments, on the following pages. It was kept as brief and simple as possible to leave as much space in memory as possible for the storage of vocabulary. The vocabulary is stored as ASCII, with one character per byte of memory. ASCII doesn't use the eighth bit of an eight bit word, so I used the eighth bit as a 'beginning of word' flag. The first character of any character, word or phrase stored in memory has the eighth bit true, and all following characters (if any) have the eighth bit zero. This scheme allows the words in Robin's vocabulary to be packed tightly in memory. The only extra bytes of memory used are flags inserted at the end of each subgroup (FDH), group (FEH) and at the end of the vocabulary (FFH).

The main program uses one subroutine from the Poly 4.0 monitor ROM. That routine, WH1, outputs a character to the video display. It uses a location in the CPU board RAM, POS, to store the next position it will print into and it recognizes several control codes:

0DH = carriage return and line feed
0CH = erase screen and send cursor home (upper left corner of screen)
0BH = send cursor home without erasing screen
18H = erase current line

The starting address for the memory area mapped by the video display is F800H. Thus, if the control code 18H is in the A register when WH1 is called, it will stuff F800H into POS. WH1 saves all registers on entry and restores them on exit.

The monitor ROM on Robin's CPU board is a slightly modified version of the 4.0 monitor: at address 0008H a JMP 2000H has been inserted so that vectored interrupt VI6 jumps to the start of the main program. This allows a single pushbutton to start Robin's system.

Robin's software was hand assembled because I didn't have an assembler program to run on her system. The program listings were typed by hand and may contain a few errors.

TEXT AND EDITOR PROGRAMS

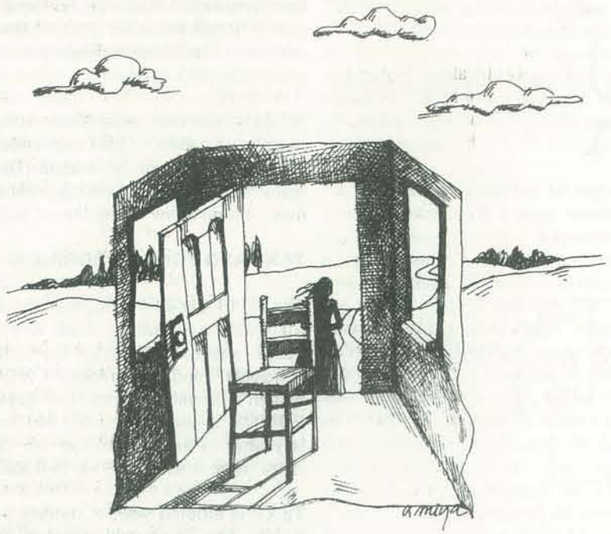
The TEXT and EDITOR programs written for Robin's system are both very short. TEXT was used to enter the messages, alphabets and vocabulary into her system's memory from the keyboard. EDITOR is used to modify her vocabulary and to add to it after the original entry. Here is what they do in detail.

TEXT is entered with a starting address in HL. The TV screen is erased and the system waits for text to be entered from the keyboard. Any unshifted letter is printed on the TV screen as a lower case letter but is stored in memory (beginning at the starting address in HL) as upper case ASCII with the eighth bit zero. The keyboard for Robin's system is a Teletype-like keyboard and does not have lower case letters, so the 'unshifted' letters are actually upper case, but TEXT translates them for display purposes.

Any letter of the alphabet typed while the CTRL key is held down (except Z) is printed on the TV screen as a capital letter and is entered into memory as upper case ASCII with the eighth bit turned on. This allows the first letter of any word or phrase to be identified. The Poly monitor program uses CTRL Z as a command to enter its front panel mode, so this is the one exception to the rule stated above. Shift O jumps to the EDITOR program, at the current address. Rubout erases the last character entered.

TEXT is also capable of inserting the control codes which identify the end of alphabets, subgroups and groups.
CTRL shift L = insert FBH
CTRL shift N = insert FDH
CTRL shift O = insert FEH

EDITOR is a somewhat longer and more complex program which allows the user to examine the text stored in the system's memory. It also allows modifications of



that text by insertions and deletions. If a deletion is made, all of the rest of the text (at addresses greater than the deleted address) is moved down one memory location to close the gap. If an insertion is made, all of the rest of the text is moved up one location to make room for the addition.

EDITOR is entered with a starting address in HL. Upon entry it will display a 'line' of text, beginning at that address. At the left end of the line, the current starting address will appear, in hex, followed by a space. Then the contents of memory are printed, up to and including the first 'control code' found. Any letters stored in memory with the eighth bit high will print on the TV as capitals, while those with the eighth bit low will print as lower case. The control codes will print as special symbols:
FBH = { FDH = } FEH = ~ FFH = ■

The EDITOR recognizes several commands, as listed below:
carriage return = display next line
line feed = display previous line
space = redisplay the current line, shifted one character to the left
NOTE: insertions made by EDITOR will go just in front of the first character on the display. The space is used to move along the current line so that insertions (or deletions) can be made in the middle of a line.
Shift O = jump to TEXT with HL equal to the starting address of current line.
CTRL shift L = insert capital L
CTRL shift M = insert capital M
CTRL shift O = insert FEH
CTRL shift N = insert FDH
rubout = delete first character of current line
any unshifted letter = insert that letter with eighth bit low
CTRL any letter except L, M, or Z = insert that letter with eighth bit high.

The EDITOR and TEXT programs use several more subroutines from the Poly 4.0 monitor ROM. Either program is entered with a starting memory address in HL. The monitor program allows register pairs to be pre-loaded from the keyboard while it is operating in the 'front panel' mode. For a detailed explanation of this procedure, see the Poly system manual Volume 2 pages 58-65. The other subroutines used are:
WHO = fetches a character from the keyboard and returns it in A. No other registers are affected.
DEOUT = print the two byte number in DE as a four character hex number.
MOVE = move -BC bytes from the area starting at (HL) to the area starting at (DE) – only works for moving to lower addresses.

Robin's main program turned out to be shorter than expected. Including the alphabets and punctuations symbols, it is 1250 bytes long. Even with 1,200 words of vocabulary in memory, there is still room for TEXT and EDITOR to remain in memory so that Robin's family can revise her vocabulary as needed.

A FEW WORDS ABOUT WORDS

It may be helpful to briefly mention how the initial vocabulary for Robin's system was chosen. The first 1,100 words were supplied by Robin's tutor, from lists of the first words taught in English. The remaining words were chosen by Robin and her family. These include the names of people, places, articles of clothing, foods and other objects which Robin comes in contact with.

As practical experience with the system is accumulated, revisions may be made in the initial vocabulary and possibly in the main program. For example, it may turn out that Robin will feel more comfortable spelling words than looking them up in the stored vocabulary. If this is the case, we may try adding a set of look-up tables for prefixes, roots and suffixes to speed up communication.

FUTURE DIRECTIONS

The basic system built for Robin can be expanded and modified to fit a wide range of possible situations. For example, the kneeswitch could easily be replaced by an electromyograph (EMG), an instrument which measures the electrical signals associated with muscle tension. An EMG can easily detect levels of muscle tension which are too weak to control a switch mechanically. This is a practical alternative to the kneeswitch for people who are only capable of very limited movement, such as an eyelid twitch. There are many hospitalized patients who experience extreme frustration because they are conscious but cut off from communication. Microcomputer communication systems of some kind may eventually become standard hospital equipment, and could help to make such patients' lives much more rich and meaningful.

There is a wide range of possible options for expanding Robin's system. It would be easy to add a printer, for example. She could assemble a message on the TV screen as usual, and then select a 'print' command which would cause the message area of the TV screen to be copied on paper. This would allow her to write an essay or a letter.

An S-100 compatible card is available from D. C. Hayes Associates (the 80-103A Data Communications Adaptor) which would allow her to select and dial a telephone number and send messages over the telephone to anyone having a computer terminal. A number of computer networks are now being used as communication networks, and it is reasonable to expect a network for handicapped people to develop in the near future.

Computers can be used to generate and control sounds. Several companies now offer S-100 compatible speech synthesis cards. It would be possible for Robin to learn to speak out loud, using one of these cards. Although the necessity of learning a new 'language' of phonemes would initially make this a slow communication process, the potential exists for this to be a very rapid communication mode.

Several companies now offer S-100 compatible circuit boards for music synthesis. It would be possible to write a program which would allow Robin to compose music and instruct the computer to perform it for her. Computer graphics are also possible. With a higher resolution video display, it would be possible for her to draw pictures with fine detail, and with a suitable printer, make 'hard copies' of these on paper.

There are several CMOS microprocessor CPU chips available now. Although CMOS memory and peripheral chips are still somewhat more expensive than TTL and NMOS chips used in Robin's system, it is already practical to build a microcomputer system similar to Robin's which would consume much less power. Such a system would be more expensive, but would be capable of battery operation, increasing portability.

S-100 compatible circuit cards are readily available which allow a computer to control electrically operated devices in its surroundings. It would be easy to expand a system like Robin's to allow her to turn on and off lights, appliances, etc. A system which can communicate can be a flexible control system too.

If you decide to try to build a microcomputer communications system for a handicapped person, I'd like to hear from you. I may be able to help with advice, and Robin might benefit from your ideas. My mailing address is:

Tim Scully
35267-136 CH
P O Box 1000
Steilacoom, WA 98388

NOTE: Thanks are due to the staff of McNeil Island Federal Penitentiary, whose cooperation made this project possible. The staff of Aquarius Electronics in Albion, California were also very helpful in tracking down parts for Robin's system. Robin's family provided the essential financial support, and Robin, her family and tutors all helped by contributing ideas and suggestions.

Tim Scully

McNeil Island December 1977 □

Programs

```
This is Robin's main program
2000 3E0C       START   MVI A,0CH       erase screen
2002 CD240C             CALL WH1
2005 CD4922             CALL PATCH
2008 22820C     ENTERL  SHLD CM         save current message address
200B 00                 NOP
200C 215622     REENTRY LXI H,SPELM-1
200F CDFF20             CALL MESSAGE    offer spelling
2012 CA8F21             JZ SPELL
2015 215F22             LXI H,PUNM-1
2018 CDFF20             CALL MESSAGE    offer punctuation
201B CAD321             JZ PUNCT
201E 217F25             LXI H,BEGIN     start of vocabulary
2021 11FE00             LXI D,00FEH     set flags
2024 CD8120             CALL MENU       offer groups
2027 11FD00             LXI D,00FDH     set up flags
202A 2A840C             LHLD CI         fetch chosen item's address
202D CD8120             CALL MENU       offer subgroups
2030 11FC01             LXI D,01FCH     set up flags
2033 2A840C             LHLD CI         fetch chosen item's address
2036 CD8120             CALL MENU       offer words
2039 2A840C             LHLD CI         fetch chosen item's address
203C 7E         XMIT    MOV A,M         fetch character
203D CD240C             CALL WH1        print it
2040 23                 INX H           next
2041 7E                 MOV A,M         fetch it
2042 E680               ANI 80H         check for start of next item
2044 CA3C20             JZ XMIT         if not next item, keep printing
2047 3E20               MVI A,' '       space after completed item
2049 CD240C             CALL WH1        fetch current message address
204C 2A0E0C     ELOP    LHLD POS        go back to offer spelling again
204F C30820             JMP ENTERL
The following subroutine is used by MENU to save the 'flags' which start out in DE (the
flag in D is 0 unless MENU is asked to display words or individual characters, the flag in E
is the indicator telling MENU what to display, eg. FEH = groups, FDH = subgroups, etc.),
it also saves the data in HL as its starting address and saves the current value of POS in
the 'current message' storage.
2052 EB         ENTER   XCHG
2053 22860C             SHLD FLAGS
2056 EB                 XCHG
2057 22800C     SENTER  SHLD CS
205A 2A0E0C             LHLD POS
205D 22820C             SHLD CM
2060 2A800C             LHLD CS
2063 C9                 RET
The following subroutine erases the top two lines of the video display without disturbing
the message displayed on the bottom 14 lines. It ends with the cursor at 'home'.
2064 3E0B       NEW     MVI A,0BH
2066 CD240C             CALL WH1        send cursor home
2069 3E18               MVI A,18H
206B CD240C             CALL WH1        erase current line
206E 3E0D               MVI A,0DH
2070 CD240C             CALL WH1        go to next line
```

```
2073  3E18          MVI A,18H
2075  CD240C        CALL WH1        erase it too
2078  3E0B          MVI A,0BH
207A  CD240C        CALL WH1        send cursor back home
207D  C9            RET
```

MAJOR SUBROUTINES: SMENU AND MENU

SMENU and MENU, which follow, are the major subroutines for displaying items on the menu (the top line of the video display). MENU is entered with flags in DE and a starting address in HL. The flags tell MENU to display groups, subgroups, words or individual characters. The starting address tells MENU where to find the first item to display. An exit from MENU is accomplished when an item is selected by use of the kneeswitch. Upon exit from MENU, the starting address of the chosen item will be in CI.

```
207E  11FB01  SMENU   LXI D,01FBH   set flags for spelling
2081  CD5220  MENU    CALL ENTER    save address & flags
2084  CD6420  ITEM    CALL NEW      erase menu
2087  22840C          SHLD CI       save current item address
208A  7E      DISPY   MOV A,M       fetch character from memory
208B  CD240C          CALL WH1      and display it
208E  23              INX H         next
208F  7E              MOV A,M
2090  E680            ANI 80H       check for msb=1
2092  CA8A20          JZ DISPY      if not, keep printing
2095  AF              XRA A         are we finished with group or
2096  BA              CMP D         are we printing with words or letters?
2097  C26421          JNZ WORD      if so, go on with words or end
209A  14              INR D         otherwise, set flag
209B  3E2D            MVI A, '-'
209D  CD240C          CALL WH1      print '-'
20A0  2B              DCX H
20A1  23      SEARCH  INX H         and look for end of group or
20A2  7E              MOV A,M       subgroup
20A3  BB              CMP E         by checking for a flag like E
20A4  DAA120          JC SEARCH     keep looking until found
20A7  2B      BACKUP  DCX H         then backup
20A8  7E              MOV A,M       and print it
20A9  E680            ANI 80H
20AB  CAA720          JZ BACKUP
20AE  C38A20          JMP DISPY
```

The next four locations store the timing constants for two time delays: T1 and T2. T1 is the time each item on the menu is displayed and T2 is the minimum time the kneeswitch has to be closed before it is considered intentional (so that accidental twitches will be ignored).

```
20B1  5050            DW 5050H      T1 time constant
20B3  5050            DW 5050H      T2 time constant
```

SUBROUTINE: SWITCH

The subroutine SWITCH looks for a switch closure for time T1 and then returns with zero in D if the switch was never closed. If the switch closes, but not for at least T2, the routine just starts over, extending T1. If the switch closes for at least T2, then after the switch is released, it returns with one in D.

```
20B5  1600    SWITCH  MVI D,0        set up 'never closed flag'
2087  E5              PUSH H
20B8  2AB120          LHLD T1       fetch time constant
20BB  E5              PUSH H
20BC  C1              POP B         put it in BC
20BD  E1              POP H
20BE  DB80    IN      IN 80H        look at switch
20C0  E680            ANI 80H       it's only one bit
20C2  CADA20          JZ CLOSED
20C5  22900C          SHLD 0C90H    waste time
20C8  2A900C          LHLD 0C90H    to make timing loop longer
20CB  22900C          SHLD 0C90H
20CE  2A900C          LHLD 0C90H
20D1  0D              DCR C
20D2  C2BE20          JNZ IN        check switch every time
20D5  05              DCR B
20D6  C2BE20          JNZ IN        keep timing
20D9  C9              RET           time up, no contact
20DA  E5      CLOSED  PUSH H
20DB  2AB320          LHLD T2       fetch time constant
20DE  E5              PUSH H
20DF  C1              POP B         put it in BC
20E0  E1              POP H
20E1  22900C  WAIT    SHLD 0C90H    waste time
20E4  2A900C          LHLD 0C90H
20E7  0D              DCR C
20E8  C2E120          JNZ WAIT      keep timing
20EB  05              DCR B
20EC  C2E120          JNZ WAIT      time up?
20EF  DB80            IN 80H        check switch
20F1  E680            ANI 80H       it's only one bit, the msb
20F3  C28520          JNZ SWITCH    start over if not still closed
20F6  14              INR D         set flag for contact
20F7  DB80    UP      IN 80H        check switch again
20F9  E680            ANI 80H
20FB  C0              RNZ           wait until it is released
20FC  C3F720          JMP UP        meanwhile looping
```

SUBROUTINE: MESSAGE

The subroutine MESSAGE is used to display a number of short messages on the menu. Message is entered with an address in HL equal to one less than the starting address of the message to be displayed. It will display the message found, up to and including a terminating '?'. Upon exit from message, the zero flag in the PSW will be one if the offered item was chosen and zero if it was not chosen.

```
20FF  000000  MESSAGE NOP NOP NOP   I deleted something here
2102  CD6420          CALL NEW      erase menu
2105  23              INX H
2106  7E              MOV A,M
2107  CD240C          CALL WH1      print
210A  FE3F            CPI '?'       check for end of message
210C  C20521          JNZ MESSAGE +6
210F  2A820C          LHLD CM
2112  220E0C          SHLD POS      restore POS
2115  CDB520          CALL SWITCH
2118  3E01            MVI A, 1
211A  BA              CMP D
211B  C9              RET
```

SUBROUTINE: COMP

The subroutine COMP is used by MENU to check the switch.

```
211C  CDB520  COMP    CALL SWITCH
211F  3E01            MVI A,1
2121  BA              CMP D
2122  C22C21          JNZ NEXT      if no contact, offer next choice
2125  2A820C          LHLD CM
2128  220E0C          SHLD POS      restore main text POS
212B  C9
```

MORE ROUTINES USED BY MENU

The following chain of routines are used by MENU to find and display the next item, check for the last item in a list, offer ESCAPE? and recycle to the beginning of the list if nothing is chosen. The details of these operations vary depending on what items are being offered: groups, subgroups, words or characters.

```
212C  EB      NEXT    XCHG          save current address
212D  2A860C          LHLD FLAGS    while restoring flags
2130  EB              XCHG
2131  7B              MOV A,E
2132  FEFD            CPI FDH       are we displaying groups or subs?
2134  D24121          JNC CHECK     if so, check for end
2137  2A840C          LHLD CI
213A  23      FIN     INX H         skip current word or letter
213B  7E              MOV A,M
213C  E680            ANI 80H       and keep skipping until the
213E  CA3A21          JZ FIN        start of the next, then check
2141  1C      CHECK   INR E         the last item will be followed
2142  7E              MOV A,M       by a flag = to E + 1
2143  BB              CMP E
2144  D25721          JNC LAST
2147  1D              DCR E         restore flag in E
2148  FEFB            CPI FBH       if no control code found,
214A  DA8420          JC ITEM       keep displaying
214D  7B              MOV A, D
214E  FEFD            CPI FD
2150  DA5721          JC LAST
2153  23              INX H         skip control code
2154  C38420          JMP ITEM
2157  CD8221  LAST    CALL ESCAPE   if last item was displayed, offer
215A  2A860C          LHLD FLAGS    escape and then loop back
215D  EB              XCHG
215E  2A800C          LHLD CS
2161  C38420          JMP ITEM      and start displaying over again
```

SUBROUTINE: WORD

WORD, the next subroutine, is used by MENU. If groups or subgroups are being offered, it is entered only after the complete offering has been printed and it jumps to COMP to check the switch. But if individual words or characters are being offered, WORD keeps printing words or characters across the menu space, with two spaces between each, until the end of the subgroup or until the end of the line.

```
2164  7B      WORD    MOV A, E      check flag
2165  FEFD            CPI FDH
2167  D21C21          JNC COMP      and split if groups or subs
216A  3A0E0C          LDA POS       check position in menu
216D  FE3C            CPI 3CH       if we are near the end of
216F  D21C21          JNC COMP      the line, stop printing &
2172  7E              MOV A, M      split or if we are at the end
2173  BB              CMP E         of the subgroup, split
2174  D21C21          JNC COMP
2177  3E20            MVI A, ' '    otherwise,
2179  CD240C          CALL WH1      print two spaces
217C  CD240C          CALL WH1
217F  C38A20          JMP DISPY     and add more to menu
```

SUBROUTINE: ESCAPE

The subroutine ESCAPE offers a return to the SPELLING mode and is used often.

```
2182  C5      ESCAPE  PUSH B
2183  214F22          LXI H, ESC-1  set up for message
2186  CDFF20          CALL MESSAGE
2189  C1              POP B
218A  C0              RNZ           return if no escape
218B  E1              POP H         clean up stack
218C  C30C20          JMP REENTRY   and reenter SPELLING?
```

SUBROUTINES USED BY SPELLING MODE

The SPELLING mode uses this chain of subroutines. The first alphabet offered is different from the other 26, and the routine doesn't backspace before printing the first letter, so there is one routine for the first letter and another for all the others. ESCAPE? is offered after each letter is printed and before a new alphabet is offered. A look-up table is used to pick the right alphabet to offer after the first letter has been printed.

```
218F  211523  SPELL   LXI H, ASTART address of initial alphabet
2192  CDB721          CALL FIRST    print first letter
2195  CD5720          CALL SENTER   to restore POS
2198  CD8221  TALE    CALL ESCAPE   offer escape
219B  21C722          LXI H, STAB   start of look-up table
219E  78              MOV A, B      fetch last letter printed
219F  BE      LOOK    CMP M         and look for it in table
21A0  CAA921          JZ FOUND
21A3  23              INX H         each table entry
21A4  23              INX H         is three bytes
21A5  23              INX H
21A6  C39F21          JMP LOOK      keep looking, you'll find it
21A9  23      FOUND   INX H         when you find it,
21AA  5E              MOV E, M      get address from table
21AB  23              INX H
21AC  56              MOV D, M
21AD  EB              XCHG          and put it in HL
21AE  CDC821          CALL SECOND   offer new alphabet
2181  CDBA21          CALL OOP      print the chosen letter
21B4  C39821          JMP TALE      and loop back to do it again
21B7  CD7E20  FIRST   CALL SMENU    offer alphabet
21BA  2A840C  OOP     LHLD CI       fetch chosen item's address
21BD  7E              MOV A, M
21BE  CD240C          CALL WH1      and print it
21C1  47              MOV B, A      save it for look-up later
21C2  3E20            MVI A, ' '
21C4  CD240C          CALL WH1      and print a space
21C7  C9              RET
21C8  CD7E20  SECOND  CALL SMENU    offer alphabet
21CB  2A0E0C  SECONDS LHLD POS      get ready to backspace
21CE  2B              DCX H         and
21CF  220E0C          SHLD POS      do it
21D2  C9              RET
```

SUBROUTINE: PUNCT

The subroutine PUNCT handles offering the control codes (by calling another subroutine) and it offers the punctuation symbols. It uses one of the spelling subroutines to handle punctuation.

```
21D3  216B22  PUNCT   LXI H, CONTROLM-1
21D6  CDFF20          CALL MESSAGE    offer CONTROL?
21D9  CAEB21          JZ CONTROL
21DC  21AC22          LXI H, PSTART   starting address of punctuation
21DF  CDCB21          CALL SECOND     offer them
21E2  CDBA21          CALL OOP        print the chosen one
21E5  C34C20          JMP ELOP        go back to offer SPELLING?
21E8  000000          NOP NOP NOP     I took out something here
```

SUBROUTINE: CONTROL

CONTROL offers and executes the control commands.

```
21EB  217322  CONTROL LXI H, BACKSPACE?-1
21EE  CDFF20          CALL MESSAGE    offer backspace
21F1  C2FA21          JNZ TWO
21F4  CDCB21          CALL SECONDS    backspace
21F7  C30820          JMP ENTERL      back to offer SPELLING?
21FA  217D22  TWO     LXI H, ERASE LAST WORD?-1
21FD  CDFF20          CALL MESSAGE
2200  C21222          JNZ THREE
2203  2A0E0C          LHLD POS
2206  2D              DCR L           back up
2207  2D      MORE    DCR L           back up
2208  3EA0            MVI A, ' '
220A  BE              CMP M           have we reached a space?
220B  C24022          JNZ RUB
220E  23              INX H           leave the space
220F  C30820          JMP ENTERL      and go offer SPELLING?
2212  218D22  THREE   LXI H, SPACE-1
2215  CDFF20          CALL MESSAGE
2218  C22322          JNZ FOUR
221B  3E20            MVI A, ' '
221D  CD240C  END     CALL WH1
2220  C34C20          JMP ELOP        back to offer SPELLING?
2223  219422  FOUR    LXI H, NEXT LINE?-1
2226  CDFF20          CALL MESSAGE
2229  C23122          JNZ FIVE
222C  3E0D            MVI A, 0DH
222E  C31D22          JMP END
2231  219E22  FIVE    LXI H, ERASE SCREEN?-1
2234  CDFF20          CALL MESSAGE
2237  CA0020          JZ START        start all over
223A  CD8221          CALL ESCAPE
223D  C3EB21          JMP CONTROL
224D  36A0    RUB     MVI M, A0H      put blank on screen
2242  C30722          JMP MORE
2245  00              NOP
2246  00              NOP
2247  00              NOP
2248  00              NOP
2249  2181F8  PATCH   LXI H, F881H    initialize text address
224C  220E0C          SHLD POS
224F  C9              RET
```

In the listing below I haven't typed the hex equivalents for the ASCII (this listing was hand-assembled).

```
2250                  ESCAPE?
2257                  SPELLING?
2260                  PUNCTUATION?
226C                  CONTROL?
2274                  BACKSPACE?
227E                  ERASE LAST WORD?
228E                  SPACE?
2295                  NEXT LINE?
229F                  ERASE SCREEN?
22AC                  .'?;:!0123456789#$"%&()*+-
22C6  FB              DB FBH          end flag
```

ALPHABET LOOK-UP TABLE

Here is the look-up table for the various alphabets, in non-standard form.

```
22C7  C13023          A 2330H
22CA  C24B23          B 234BH
22CD  C36323          C 2363H
22D0  C47723          D 2377H
22D3  C59123          E 2391H
22D6  C6AC23          F 23ACH
22D9  C7C223          G 23C2H
22DC  C8DA23          H 23DAH
22DF  C9F123          I 23F1H
22E2  CA0C24          J 240CH
22E5  CB1324          K 2413H
22E8  CC2B24          L 242BH
22EB  CD4624          M 2446H
22EE  CE6024          N 2460H
22F1  CF7B24          O 247BH
22F4  D09624          P 2496H
22F7  D1AD24          Q 24ADH
22FA  D2B124          R 24B1H
22FD  D3CC24          S 24CCH
2300  D4E624          T 24E6H
2303  D50025          U 2500H
2306  D61A25          V 251AH
2309  D72825          W 2528H
230C  D83F25          X 253FH
230F  D95725          Y 2557H
2312  DA7025          Z 2570H
```

THE ALPHABETS

And here are the alphabets, once again without the hex.

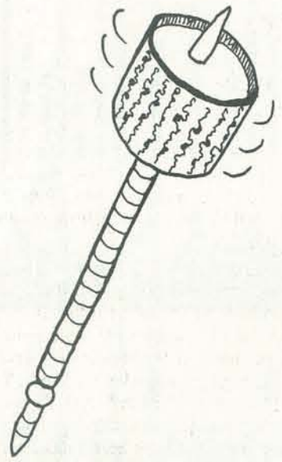
```
2315                  ASTART  TAOSWIHCBFPMRELNDUGYJVQKZX
232F  FB              DB FBH          end of alphabet flag
2330                  NTSRLDCIGVMYPBKUFWOJXHZEQA
234A  FB              DB FBH
234B                  EAOUYRISLJTVMBDWCGHNPFK
2362  FB              DB FBH
2363                  OEHATKILURCYSONDZMW
2376  FB              DB FBH
2377                  EIUARSOLMDGYNVJQWHEFTPKBZ
2390  FB              DB FBH
2391                  RSNDALMCETVFPXIGYOWUHQKBJZ
23AB  FB              DB FBH
23AC                  ORIFEAULTSYWBMGCHNJPD
23C1  FB              DB FBH
23C2                  EHROAIGSLUTNYMFDBWZJKPC
23D9  FB              DB FBH
23DA                  EIAOTURYLNWDSMBHQFPCGK
23F0  FB              DB FBH
```

```
23F1                  NSTOCMLAREDVGPFBKXUZQIJLWY
240B  FB              DB FBH
240C                  AEOUIJ
2412  FB              DB FBH
2413                  EISANLYOGFWTURDPMKBJCHV
242A  FB              DB FBH
242B                  EIALYODTSUFRMVWKPCBGNHJZXQ
2445  FB              DB FBH
2446                  EAOIPMUYSBLFNTHCDRWGJKVCZ
245F  FB              DB FBH
2460                  DTEGSCIAOYNLFVUKMJRQPHWXBZ
247A  FB              DB FBH
247B                  NFRUMPLTOWSDCVIBEYAKHJGXZQ
2495  FB              DB FBH
2496                  ROAELTSPIHMUYWFGKBNDCJ
24AC  FB              DB FBH
24AD                  UIO *
24B0  FB              DB FBH
24B1                  EIOATSYDMNURCLVKGPWBFHXQJZ
24CB  FB              DB FBH
24CC                  TEIOSHUCAPYKMWNLGQFBDRVJZ
24E5  FB              DB FBH
24E6                  HEIOARSTUYLWCFMNBPDZGKVJQ
24FF  FB              DB FBH
2500                  TSNRLCGPAEMDIFBOYZXUVKQJH
2519  FB              DB FBH
251A                  EIAOYUSRVZKGM
2527  FB              DB FBH
2528                  EAHIONRSLTDYKUPFBCMZWG
253E  FB              DB FBH
253F                  EPTICAHUYOQLNWFSVGBKMRD
2556  FB              DB FBH
2557                  EOSAITPMBLNWCRGDZHUFVXIK
256F  FB              DB FBH
2570                  EAZOYIUKTVWHJB
257E  FB              DB FBH          end of alphabets
257F  C1C1424C45C1424F AAbleAbout     beginning of vocabulary storage
      5554
```

TEXT AND EDITOR

```
3F00  CD200C  TEXT    CALL WHO        keyboard input
3F03  FE7F            CPI 7FH         is it rubout?
3F05  CA263F          JZ RUB
3F08  FE5F            CPI 5FH         is it shift 0?
3F0A  CA383F          JZ EDITOR
3F0D  FE1C    CTL     CPI 1CH         is it a control character?
3F0F  DA213F          JC CONTROL
3F12  FE20            CPI 20H         is it a control code?
3F14  D2193F          JNC PRINT       if not, print it
3F17  C6DF            ADI DFH
3F19  77      PRINT   MOV M, A        store it in memory
3F1A  CDE53F          CALL LPRINT     put it on TV
3F1D  23              INX H           next memory location
3F1E  C3003F          JMP TEXT        do it all over again
3F21  F6C0    CONTROL ORI C0H         make eighth bit high
3F23  C3193F          JMP PRINT       for 'capital' letters
3F26  CDE53F  RUB     CALL LPRINT     rubout on TV
3F29  2B              DCX H           back up in memory
3F2A  C3003F          JMP TEXT        go do it over
3F2D  2A800C  RETEXT  LHLD 0C80H      fetch starting address
3F30  3E0C            MVI A, 0CH      erase TV
3F32  CD240C          CALL WH1
3F35  C3003F          JMP TEXT
3F38  22800C  EDITOR  SHLD 0C80H      save start of current line
3F3B  3E0D            MVI A, 0DH
3F3D  CD240C          CALL WH1        start a new line
3F40  2A800C          LHLD 0C80H      fetch start of current line
3F43  EB              XCHG
3F44  CDD103          CALL DEOUT      print address in hex
3F47  EB              XCHG            restore address
3F48  3E20            MVI A, ' '
3F4A  CD240C          CALL WH1        print space
3F4D  7E      LOOP    MOV A, M        fetch character from memory
3F4E  CDE63F          CALL LPRINT     put it on TV
3F51  7E              MOV A, M
3F52  23              INX H
3F53  FEFB            CPI FBH         was it the end of a line?
3F55  DA4D3F          JC LOOP         if not, keep printing
3F58  CD200C  KEY     CALL WHO        wait until a key is pressed
3F5B  FE20            CPI ' '         is it a space?
3F5D  C2673F          JNZ M1          if not, keep checking
3F60  2A800C          LHLD 0C80H      fetch starting address
3F63  23              INX H           space skips one character
3F64  C3383F          JMP EDITOR      and reprints line
3F67  FE7F    M1      CPI 7FH         is it rubout?
3F69  C2873F          JNZ M2
3F6C  2A800C          LHLD 0C80H      fetch starting address
3F6F  E5              PUSH H          copy HL
3F70  D1              POP D           into DE
3F71  3EFF            MVI A, FFH      end of vocabulary flag
3F73  010000          LXI B, 0        start counting at zero
3F76  2B              DCX H
3F77  23      M3      INX H
3F78  0B              DCX B           count one byte
3F79  BE              CMP M           check for end flag
3F7A  C2773F          JNZ M3          keep counting if not the end
3F7D  2A800C          LHLD 0C80H      fetch starting address
3F80  23              INX H           we are moving one space
3F81  CD0001          CALL MOVE
3F84  C3383F          JMP EDITOR      display edited line
3F87  FE0D    M2      CPI 0DH         is it carriage return?
3F89  CA383F          JZ EDITOR       then display next line
3F8C  FE0A            CPI 0AH         is it line feed?
3F8E  C2A03F          JNZ M4
3F91  2A800C          LHLD 0C80H      fetch starting address
3F94  2B              DCX H           back up
3F95  2B      M5      DCX H           keep backing up
3F96  7E              MOV A, M
3F97  FEFB            CPI FBH         look for control flag
3F99  DA953F          JC M5           and keep backing up until found
3F9C  23              INX H           skip the flag
3F9D  C3383F          JMP EDITOR      and display previous line
3FA0  FE5F    M4      CPI 5FH         is it shift 0?
3FA2  CA2D3F          JZ RETEXT       if so, go to TEXT
3FA5  FE1C            CPI 1CH         is it a control character?
3FA7  DABE3F          JC M6           if so, it is upper case
3FAA  FE20            CPI 20H         could it be a control code?
```

Prayer Wheel Program



BY EDRID

When I finished building my computer, I wanted to do something far out with it to start off right. Having been a meditator for some time, I thought of a computer implementation of a Tibetan Prayer Wheel. I chose an ancient high mantra for the first thing my computer would do in its present incarnation.

We had the good fortune to meet Sonam Gyatso, a genuine Tibetan Lama. When told of my computer's 'recitations', he beamed brightly and said, characteristically, 'Oh my! Great Merit!'

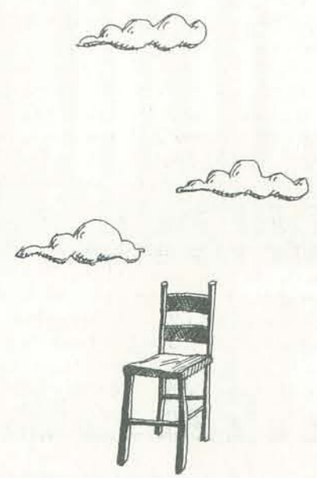
As of January 27, 1978 the number of recitations by my computer was 22,199,184. We encourage the spread of this program, and would like to know of other implementations.

Edrid
c/o Dynabyte
4020 Fabian Way
Palo Alto, CA 94303

```
10 REM * * * * MANTRA * * * *
20 REM * * A PRAYER WHEEL PROGRAM * *
30 REM * * WRITTEN BY EDRID * * * *
40 REM * * IN NORTH STAR BASIC * * *
100 OPEN #0, "MANTRAF"
110 READ #0, N: CLOSE #0
120 CHR$(12)
130 PRINT: PRINT: PRINT: PRINT
140 PRINT "MUMBLE. . . MUMBLE. . ."
150 DIM M$(18)
160 FOR M=1 to 1007: READ M$
170 RESTORE: NEXT M
180 DATA "OM MANI PADME HUM!"
190 PRINT M$
200 N = N + 1008
210 PRINT "                ", N
220 OPEN #0, "MANTRAF"
230 WRITE #0, N: CLOSE #0
240 GOTO 160
```

100-110 gets the number of past recitations of the mantra from the disk.
120-140 clears the screen and prints a message to give a hint of what is going on.
150-170 prepares a space in memory for the mantra, then puts it in there over and over for 1007 times, one less than the number of petals in the Crown chakra.
180 is the mantra.
190 prints the 1008th.
200 adds 1008 to the number of recitations.
210 prints the total number of recitations of the mantra.
220-230 puts the new total onto the disk, with the thought that some of the power of the 1008 recitations is within the number.
240 goes back to the beginning to do the whole thing over again, endlessly.

```
3FAC  D2C03F          JNC INSERT      if not, insert it as is
3FAF  FE1E            CPI 1EH         be sure it is not M or L
3FB1  D2B93F          JNC M7          if it's not, then control code ok
3FB4  C6B0            ADI B0H         make into L or M
3FB6  C3C03F          JMP INSERT      and insert it
3FB9  C6DF      M7    ADI DFH         make into control code
3FBB  C3C03F          JMP INSERT
3FBE  F6C0      M6    ORI C0H         make into capital letter
3FC0  2A800C  INSERT  LHLD 0C80H      fetch starting address
3FC3  F5              PUSH PSW        save character on stack
3FC4  3EFF            MVI A, FFH      end of vocabulary flag
3FC6  010000          LXI B, 0        start counting at zero
3FC9  2B              DCX H
3FCA  23        M8    INX H
3FCB  0B              DCX B
3FCC  BE              CMP M
3FCD  C2CA3F          JNZ M8
3FD0  54        M9    MOV D, H        move forward
3FD1  5D              MOV E, L
3FD2  13              INX D
3FD3  7E              MOV A, M
3FD4  12              STAX D
3FD5  1B              DCX D
3FD6  2B              DCX H
3FD7  0C              INR C           count one space
3FD8  C2D03F          JNZ M9
3FDB  04              INR B
3FDC  C2D03F          JNZ M9
3FDF  F1              POP PSW         get back character
3FE0  12              STAX D          and insert it
3FE1  23              INX H
3FE2  23              INX H
3FE3  C3383F          JMP EDITOR
3FE6  FE60    LPRINT  CPI 60H         is it upper case?
3FE8  D2240C          JNC WH1         print as is
3FEB  FE41            CPI 41H         is it lower case?
3FED  DA240C          JC WH1          if not, print as is
3FF0  C620            ADI 20          make it lower case
3FF2  C3240C          JMP WH1         and print it
```



COMPUTER CONTAGION

or Nightmare in the Submarine Shop

BY HOWIE DI BLASI

For 14 years, Howie DiBlasi has been teaching electronics. He is currently both Director of Vocational and Industrial Education and Media Director at Lake Havasu High School in Lake Havasu City, Arizona. He has developed software for inventory control as well as video tape retrieval information for the school's media services. He is now working on programs for storing and retrieving student information, including grades.

I walked into the store and looked around. I saw a man sitting in front of a typewriter watching a piece of paper come out of the roller. It had something printed on it. I walked closer so that I could get a better look at what was on the paper. The man moaned. Then silence. He typed something on the keyboard. The typewriter came to life again. It was typing by itself. What was this? He looked at the typed page. He groaned. He mumbled. Something about phasers, starships and dumb Klingons. Then he got up and walked away muttering something about blowing up the Enterprise. WOW! Was I in a strange place!

I walked over to the magazine rack and picked up an issue of *People's Computers*. As I thumbed through it I overheard two people talking about bits, bytes, assemblers, machine language, hex, IMSAI's, Poly's and SOL's.

I walked over to some equipment that was set up on a table and pretended to know what I was looking at. I wondered why the number pad on that machine went from 0-9 and A-F. What a strange way to count. Normal people count from 1 to 10. But I quickly remembered that I was not in a normal store. I wondered what they used all the computer stuff for. And it probably cost hundreds and thousands of dollars. Much more than our school district could ever afford.

Clockwise: Benny Santopietro, Joe Ruiz, Denise Burns and Cari Long try out the new microcomputer system built by the students at Lake Havasu High School.

Doug Browning (left) and Ed Dixon (right) put the finishing touches on the Seals 8K memory board.

Then it happened. Someone said, 'May I help you?' If only he knew. 'Help me', he said. Where did this nightmare start? I thought back.

I had been following the articles in *73* magazine about microprocessors. The articles had provided me with some good lecture material for my electronics classes at our high school where I teach. What I could not understand was how these people could get so excited about an electronic device. I have been teaching for 14 years and I really enjoy my work but these computer nuts were in another world. All this talk about being hooked on a computer. Bah! Humbug!

I had to go to Phoenix, Arizona for a teachers' conference and had heard that there was a Byte Shop there. Four weeks before I had thought a Byte Shop was a new submarine shop. I decided to drop in and see why all this excitement about microprocessors.

My mind came back to the present. The salesman asked me if I was OK. I said yes, and maybe he could answer a few questions for me. 'I'm Howie DiBlasi from Lake Havasu High School', I said, as we shook hands. 'Alan Hald, what can I help you with?' 'Well, tell me about these microprocessors, and if you could please, given me a demonstration.'

Alan explained that he and Jeff McKeever were the owners of the Byte Shops of Arizona. Two and a half hours later I had learned about computers for personal use, a few BASIC fundamentals, hexidecimal format, looked at several types of systems, played Star Trek and spent $35 on a few books and magazines so I could find out more about this computer business. I thanked Alan for his help and walked out of the store. I felt something byte me. Little did I know how infected that byte would get.

THE INFECTION SPREADS

During the next three weeks I read everything I could get my hands on. I devoured books and magazines like candy. At night my wife kept asking me if I was OK. I would just grunt and keep on reading. All my friends were getting tired of me talking about computers everytime I saw them. My electronics students were more sympathetic. We were studying about transistors and IC's and the information about microprocessors fit right in. The more I told the students the more they wanted to know. Now the students were reading all of my books and magazines. They were telling their parents about the computers. The infection was spreading.

Two weeks later I had to travel to Los Angeles for a conference on Audio Visual Education. I decided that I would stop in at a few computer stores when I had some free time. The more stores I went to the worse I got. I found out about schools using computers and many other applications for our electronics and Audio Visual programs. I spent some time talking to people who had built their own systems from kits. After hearing the pros and cons I decided that our electronics students could build a system.

THE PLUNGE

I returned to Lake Havasu convinced that we should include a computer in our electronics classes. I called the Byte Shop back in Phoenix and Alan and I discussed the items that would best suit our program. We wanted a product that was durable and would stand up under student use every day. Our classes could go through the 'how it works' process and also put the computer to work in as many applications as possible. An order was placed for the following equipment:

1 IMSAI Microcomputer
1 ACT-1 Terminal
1 National Multiplex 2SIO Interface
2 Processor Technology 4K RAM Boards
1 Seals 8K RAM Board
1 Educassette Recorder

I had a video monitor in our TV studio to get us started; later we would order a monitor for the computer.

All of the items had to be ordered from the factory; we were told they would arrive in a few weeks. We spent the next two weeks discussing IC's, memory devices, kit construction, logic, truth tables and binary math. Two of the items were delayed a week, but at the end of three weeks we had everything.

EPIDEMIC

I divided the class into small groups and had them assemble the kits together. Each student checked on others in the group to try and catch any mistakes. The students were so excited about the computer I could not believe it. Every free moment they had was spent in the lab working on the computer—this was over and above the time they spent in my class working on the kits. Students went from one kit to another checking to see what other groups were working on. We tried to keep each group working at the same pace so that things would fall into place when the kits were completed.

Kids kept working on the kits, reading, discussing—all of us were learning together. I could not give the information to the students fast enough. I had to run them out of the electronics lab almost every day. Our school has classes that start at 7:00 AM for students who like to take an early class. About 25% of the students elect to do this, the rest start at 8:00 AM. School is out at 2:40 PM but there were days when we did not leave school until 5:30. How about that! Kids staying after school because they want to learn more. What is this world coming to?! The infection was spreading to so many areas it was becoming an epidemic—I wondered if it could be kept under control.

DANGER, DANGER, MOS, MOS

When you pick up the memory chips for the RAM boards you cannot help but notice the big bright letters that say 'Danger, MOS Device'. That is enough to scare anybody. I decided that a little research would pay off. An excellent article by Joe Magee can be found in the February 1977 issue of *Kilobaud*. Another reference can be found in the August 1977 issue of *Popular Electronics*.

We took the first necessary precaution by talking about the MOS device and how sensitive it is to static electricity. We set up part of the lab area with a large piece of sheet metal and connected it to ground. We then connected a wire to a good earth ground with a 1 megohm resistor at the end. The other end was connected to the wrist of the student who was putting the MOS devices into the sockets.

Some more tips: It's important to check to see if you have on any synthetic clothes. It is best to wear only cotton when installing the devices. One other thing you can do if you are afraid that you have not taken enough precautions is to get a bucket of water and place your bare feet into it!

MOMENT OF TRUTH

We were all getting anxious. The moment of truth was getting very close. The IMSAI was complete. We put the three memory boards into the S-100 bus slots. The National Multiplexor 2SIO Board was in the slot. Cables were run to interface the ACT-1 to the SIO interface and to the cassette recorder. Video monitor cables were connected.

We all crossed our fingers. Power on. Nothing. Everything plugged in? Nope. Plugged in the IMSAI. Turned it on. Lights on the data bus and input came on! Checked to see if the ACT-1 terminal worked. Nothing on the screen. Connections OK? Yes, everything checked OK.

Two hours later I still could not get the terminal to display letters on the screen. I decided that it would be a good idea to inspect the wires and cables that connected everything together. It took ten minutes to find a loose wire on the National Multiplex 2AIO board. When it was soldered it was not connected properly and had come loose; a quick solder job solved that problem.

We were all ready to try it again. Power on. On the IMSAI I hit the switches: Stop, reset, examine, run. Well, I'll be! A question mark came up on the screen, just like it was supposed to. So far so good.

WHAT'S A KILDOT?

We went over the National Multiplex information that explained the Dump, Save, Load and other procedures to get data into the memory and out of the memory. It was time to see if everything would work when we put a program in. We had purchased a tape with BASIC and a few games on it from the Byte Shop in Phoenix. I spent the next hour trying to feed BASIC and Star Trek into memory.

Clockwise: Benny Santopietro, Denise Burns and Cari Long try their hands at programming the new microcomputer system built by the students at Lake Havasu High School.

Photos by Howie DiBlasi.

No luck. All I could get was parity errors. A phone call to Phoenix informed me that the heads on my tape recorder were not aligned the same as the heads on the machine the tape was made on. Alan went through the procedures of loading the tapes but nothing would work. He told me to send him the unit and he would align it properly and make new tapes.

Meanwhile, Alan suggested we could still try out 'Kildot', an assembly language program in the IMSAI manual. He spent 40 minutes on the phone going over the correct way to feed in a program in assembly language. He explained hex code and how to use it. We made a number of mistakes while doing the program but he was very patient with us and was really a big help. I sure am glad it was his dime for the phone call. We fed the program into memory, recalled it, examined the program and then saved it on tape. We had dots flashing on the LED's on the IMSAI and the kids had a lot of fun with their first program. The idea of the game is to catch a LED when it is on; then the program decreases the flash rate of each LED. If you do not hit the LED the speed increases. It provided me with an opportunity to discuss binary math and how to count.

UP AND AT 'EM

I sent the tape recorder to Phoenix and had it back in a week. I fed Star Trek in and had it running in a matter of minutes. The kids were jumping up and down. They could not contain themselves. Try to imagine the first time you saw Star Trek running on a micro. It is really something to see. Everything was going bananas. Kids were pushing each other to get a better look. One student got so excited he had to excuse himself to go to the bathroom. I got the kids calmed down and then went through the game of Star Trek and explained how to play it. It took about an hour to explain the game and answer their questions.

We set up a schedule so that each student could have a turn. As they were playing the game my mind started drifting. The computer enthusiasm had spread everywhere. Teachers were talking about it, kids were talking about it with their parents, everything I was reading had something to do with computers. My wife was sick and tired of me talking about micros and my friends were sure I had gone off the deep end.

NOW WHAT?

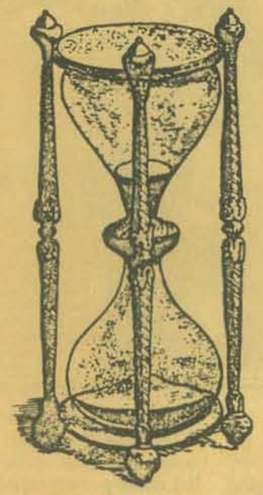
If I really put my mind to it we can start to use this computer for some really functional and educational things around the school. How about using it for:

- Keeping student schedules
- Storing student grades
- Inventory of school equipment
- Budget and purchase order control
- Mailing list of students
- Mailing list of teachers
- Word processing
- Student checks on subject matter
- Student standard mean and deviation
- Math plots and equations
- Educational simulations
- Audio visual materials available to teachers
- Video tapes available to teachers
- Films and movies available to teachers

Hey! What am I doing sitting here? I have a lot of work to do. Let's see, I think I will start with the program for the student schedules. Then maybe the principal will try to find me some more money to purchase another computer. With all the programs we are going to have we may need several. I wonder if computer bugs ever byte principals?

Measuring TIME on the PET and Other Microcomputers

BY LARRY TESLER



The operation of a microcomputer requires one or more "clocks". Some of the "clocks" in a computer are quite like the clocks we use to tell time. They count from zero to some maximum number at a regular rate and then start over at zero again. Other so-called "clocks" simply swing like a pendulum between two voltages. They are like a watch that only tells whether it is day or night, except that their rate of counting is a million or more times a second. In this article we will discuss the first kind of clock, and how a program can use such a clock to measure time intervals.

There are several reasons one might want to measure time intervals in a computer program. Among them are:

- to time portions of a slow program to determine where it needs work;
- to introduce delays into a fast program to match human reaction time;
- to time events that occur on external devices;
- to cause events to happen according to a schedule;
- to tell the time of day.

The PET has several methods of measuring time intervals. Similar methods are available on other microcomputers. Just as we use stopwatches to measure hundredths and tenths of seconds, ordinary watches to measure seconds, minutes, and hours, and calendars to measure days, weeks, and months, the PET has different measuring instruments to measure intervals of different magnitudes. Each of these instruments will be described below. In summary, they are:

- TI$ counts in hours, minutes, and seconds like a digital watch;
- TI counts in units of 1/60 second ("jiffies");
- PEEK(59465) counts in units of 1/3900 second (256 microseconds);
- PEEK(59464) counts in units of one millionth of a second (1 microsecond);
- PEEK(512) and PEEK(513) count in units of about 18 minutes and about 4 seconds, respectively.

If you have a computer but not a PET, consult its manual under the heading "Interval Timer" or "Real Time Clock" to find out how to measure time. [If the processor is a 6502 type, there is probably a 6522 PIA device like the one the PET uses for methods (3) and (4), but the exact memory addresses to use are probably not 59464 and 59465. Look for locations 8 and 9 in the description of the PIA.]

To demonstrate the use of each method, we will assume that there is a BASIC program in which one wants to measure the time between the performance of statements 100 and 200.

MEASURING HOURS, MINUTES and SECONDS

In January, Commodore began distributing a booklet called *An Introduction to Your New PET Personal Electronic Transactor*. This 38-page booklet includes a discussion of how the variable TI$ can be used to measure time in hours, minutes, and seconds. The value of TI$ is a string of the form "HHMMSS", e.g., to set the time to 2:35 p.m., use:

```
TI$ = "143500"
```

To measure the time DT$ between statements 100 and 200, one could use this program:

```
100 TI$ = "000000"
  .
  .
  .
200 DT$ = TI$
```

A result of DT$="011258" would mean that 1 hour, 12 minutes, and 58 seconds elapsed between statements 100 and 200. The maximum value of DT$ is "235959", after which it rolls over to "000000". This method is suitable for measuring intervals shorter than 24 hours. Note, however, that the crystal-controlled clock on your PET may not be adjusted perfectly, and it may not tell time as accurately as certain applications demand.

MEASURING UNITS OF 1/60 SECOND

The PET user's booklet explains the use of TI to measure jiffies (1/60 second units). The value of TI is an integer equivalent to the number of jiffies that have elapsed since TI$ was set to "000000". It is not possible to assign to TI directly.

To measure the time DT between statements 100 and 200, one could use this program:

```
100 TI$ = "000000"
  .
  .
  .
200 DT = TI
```

A result of DT = 375 would mean that 375 jiffies, or 6.25 seconds, elapsed between statements 100 and 200. The maximum value of DT is 5184000, after which it rolls over to 0. This method, too, is suitable for measuring intervals shorter than 24 hours.

One disadvantage of the two programs shown so far is that in both programs TI$ is reset. If you were using TI$ to keep track of the time of day, this can mess things up. The following program avoids that problem:

```
100 T = TI
  .
  .
  .
200 DT = TI - T
```

Here, we record the time before and after the interval, and subtract them to determine the elapsed time. But now there is another problem. If the clock strikes midnight between statements 100 and 200, TI will roll over to zero and DT will be negative. If you don't want your program to turn into a pumpkin, add the following statement:

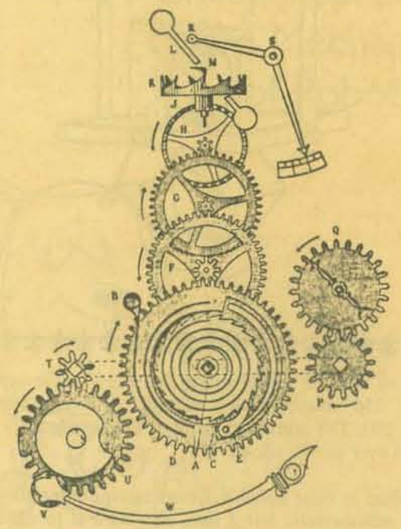
```
210 IF DT<0 THEN DT = DT + 24*60↑3
```

It will add 24 hours worth of jiffies to the negative number to yield the true elapsed time.

MEASURING UNITS OF 256 MICROSECONDS

Here is a technique not described in Commodore's booklet. It takes advantage of a 6522 PIA device inside the PET, which has an interval timer built into it. The timer is controlled by various registers addressable by the program. It counts *down* from 255 to 0 and then goes back to 255 and continues to count.

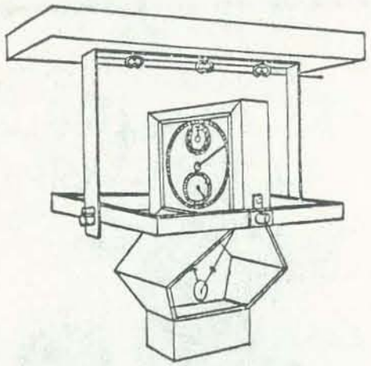
For simple applications, it is possible to access the interval timer from BASIC using the commands PEEK and POKE. PEEK(A) returns the contents of hardware address A, which could be either a memory location or an input register. POKE A, B stores B into address A, which could be either a memory location or an output register. The following program PEEKs and POKEs the interval timer to measure the time between statements 100 and 200:



```
1 DT=0: TH=59465: TA=246
```
Initialize these variables early to ensure that later accesses will take place in a rapid and predictable amount of time. The time intervals being measured are actually shorter than the time it takes BASIC to do many things, so this precaution must be taken to keep the measurements accurate.

```
  .
  .
  .
100 POKE TH,0
```
Reset the interval timer to zero.

```
  .
  .
  .
200 DT=TA-PEEK(TH)
```
Compute the elapsed time. Subtract it from the fudge factor TA to correct for time spent in executing the BASIC statements. The constant TA=246 has been calibrated to yield a DT of zero when no statements fall between 100 and 200. It is possible that on other PETs, especially with later releases of the system software, a different value of TA will be needed.



A result of DT=35 would mean that 8.96 milliseconds (35 units of 256 microseconds) elapsed between statements 100 and 200. The maximum value of DT is 246, after which it will roll over to –9, –8, etc.

This method is most suitable for measuring intervals shorter than 63 milliseconds, but I have also used it to distinguish more lengthy intervals as long as their difference is less than 63 ms.

MEASURING UNITS OF 1 MICROSECOND

The interval timer has a register at address 59464 that counts down every microsecond. Whenever it reaches zero, it causes the slower 256 microsecond timer to count down by one, and it starts itself counting down again from 255.

It is about as sensible to use the one microsecond timer in a BASIC program as it would be to use a stopwatch to time the growth of a carrot. Consider it only if you plan to write machine language programs (see Don Inman's *Data Handler* series in previous issues of *People's Computers.*)

MEASURING MISCELLANEOUS UNITS

The first release of the PET operating system uses the following memory locations to calculate TI and TI$:

PEEK (514) increments every jiffy. Counts from 1 to 255 in about 4 seconds.
PEEK (513) increments every 256 jiffies. Counts from 0 to 255 in about 18 minutes.
PEEK (512) increments every 65536 jiffies. Counts from 0 to 80 in 24 hours.

TIMING EVENTS ON EXTERNAL DEVICES

I purchased an inexpensive joystick constructed in such a way that variation of its position in either axis varied the resistance of a potentiometer in a proportional fashion. I then constructed a conventional joystick interface, in which the variable resistance is incorporated into a simple RC oscillator circuit controlled by a 560-type timer IC. Such a circuit has the property that the rise time of the capacitor is proportional to the resistance, and thus to the position of the joystick.

The PET user port, or any computer's parallel I/O port, can talk to such an interface in various ways. I arranged it so that the charging of the capacitor was started by a transition from 1 to 0 on the output pin PA1 of the user port, and so that the external circuit signalled input pin PA0 when the capacitor became charged beyond a preset threshold. The elapsed time between the two events varied between about 96 and 120 milliseconds depending on the position of the joystick in that axis. (The other axis used pins PA2 and PA3.)

User port pins PA0 and PA1 on the PET can be addressed from BASIC by PEEK and POKE to the 1- and 2-bits of memory address 59471. The corresponding bits in memory location 59459 determine which pin is input and which is output. The bit-pin correspondence is as follows:

Pin	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
Bit	128	64	32	16	8	4	2	1

I was surprised to discover that a BASIC program could be written to determine the position of the joystick. The program resets the interval timer to zero just before starting the oscillator cycle. It then uses the WAIT command to wait for the input signal to change to a 1. After 96 to 120 ms, when the input signal changes, the timer is in the midst of its second 63 ms countdown. Here is the program with annotations:

```
1 DT=0: TH=59465: TA=80: PP=59471
```
Initialize DT and the constants. The calibration factor TA has been selected for our particular joystick to make the center position come out DT = 0 and the range be –65 to +65.
```
2 POKE 59459, 2
```
Make PA0 be an input signal and PA1 be an output signal.

.
.
.

```
300 POKE PP,1: POKE TH,0: POKE PP,0: WAIT PP,2,2:
    DT=TA–PEEK(TH)
```
POKE to output a 1 on PA0.
POKE to reset the interval timer.
POKE to output a 0 on PA0 and thus to start the capacitor charging.
WAIT until bit 2 of location PP becomes nonzero.
Calculate and adjust the time DT that elapsed while waiting.

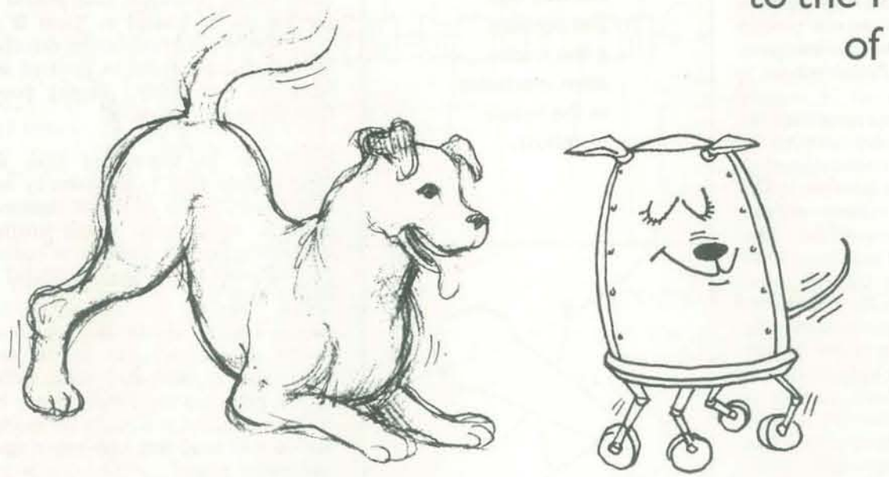
A result of DT=–65 would mean that the joystick is all the way to the left, DT=65 all the way to the right, 0 in the center, and other values in proportional positions.

Well, I hope these tips will help you have a good time with your PET! □

The Patter of Little Feet

A Cheap Approach to the Mechanics of Robotics

BY ROBERT ROSSUM



The name Rossum may be familiar to science fiction fans–it comes from the Čapek play, R.U.R. *The play is commonly cited as the source of the term 'robot' as it is commonly used ('R.U.R.' stands for 'Rossum's Universal Robots.') Members of the United States Robotics Society (USRS) are using the family name 'Rossum' as a kind of collective pseudonym for their publications. Members who prefer to be anonymous may publish through USRS under whatever 'Rossum-name' they reserve. Thus far, half a dozen names have been spoken for, e.g. 'S. A. Rossum,' 'D. I. Rossum,' and some folks whose real family name is Rossum have been listed.*

'Robert Rossum' writes books, articles, and non-theatrical motion pictures. He has spent most of the past 20 years working in research and developmental laboratories.

Additional background material on designing a robot may be found in our back issues–see Rossum's articles 'Robots as Household Pets' (Vol 5 No 4) and 'Pet Robots: New Capabilities' (Vol 6 No 1) and 'Careful Bull in the China Shop' (Vol 6 No 4).

We thank MITs for permission to reprint the figures in the article from the October issue of Computer Notes.

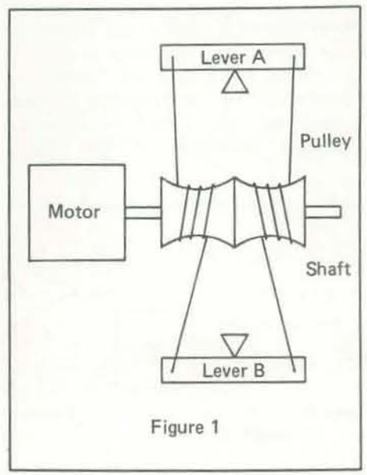


Figure 1

Recap: In the first of the two parts of this series, I described the double windlass mechanism as a simple, cheap approach to the design and construction of mobile systems by poverty-stricken non-engineers.

The basic idea is that a very small amount of energy applied to either Lever A or Lever B (See Figure 1) pulls the cord snug around the pulley and thereby draws energy from the main motor to move the opposite lever until tension on the cord is relieved. In this way, very small amounts of energy can be used for control of a large number of paired mechanisms along a shaft driven by a single main motor. The power of the motor is shared by various mechanisms as it's needed. So any particular mechanism can obtain as much energy as it needs up to the full capacity of the motor, while the average load on the main motor is comparatively low. The elements of the system are not much more complex than Tinkertoys, allowing the frustrated amateur to build his/her own experimental systems at a modest cost.

○ ○ ○

When Nature designed animals to move around, she gave them legs instead of wheels. For such animals as snails, caterpillers, and snakes, 'tracks' were provided. However, complex as they are, legs are the big winners in locomotion. Wheels just don't seem to work well unless the terrain is fairly level and smooth. It seems a shame to design robots with systems Nature has rejected, but we can usually control our robots' environments more easily than we can build flexible robots.

Perhaps the most discouraging fact to those who hope to develop anti-gravity devices is that no living creatures employ them. If antigravity were possible, surely some animals would have evolved practical anti-gravity capabilities. The competitive advantage of such capabilities would be enormous. In this case, even to the born optimist, the fact that something hasn't already been done strongly suggests that it can't be done. There's no evidence either, that animals can do well on home-grown wheels. Lovable as *Star Wars*' R2D2 may be, it's hard to believe that his little wheels carried him smoothly through those sand dunes. So flapping wings, waving fins, walking legs, and perhaps a few tracks seem inevitable in the future of robots.

Flapping wings, waving fins, walking legs, and perhaps a few tracks seem inevitable in the future of robots.

Since legs are the most practical, let's talk about them. We can construct the best design by copying Nature. 'Maybe the only sensible thing to do,' commented one engineer, 'is to find a dog skeleton and make an investment casting of his legs. That's a great design.' Basically, the dog's leg looks like the diagram in Figure 2.

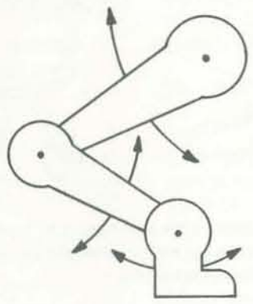


Figure 2

This is a simplified sketch, but it shows that only three moves are required to make the leg functional. Each section of the leg can be treated as 'Lever B' in the double windlass mechanism described here. Each section can be powered and controlled individually, drawing power from the single main motor.

Power can be transmitted from the rotating main shaft to the levers by way of a system similar to an old-fashioned dentist's drill; three double-windlass mechanisms operate together to control the movement of one leg, as indicated in Figures 3–6.

In this case, it may be desirable to use pulleys of different sizes on the shaft (see Figure 7), since the longest section of the limb may need more power but less speed applied to it, while the shortest section may work best with lots of speed and limited power.

Clearly, the coordination of movements for effective use of such a leg is complex, requiring accurate timing and logical operations dependent on feedback of information from the real world. This article deals only with the principles of the mechanical system, not with the logic required. The computerist can look forward to many years of developing logic that lets the clumsy baby mechanism grow up into a coordinated artificial animal. However, the logic necessary is not mysterious; it's largely a matter of timing and sequencing, influenced by feedback.

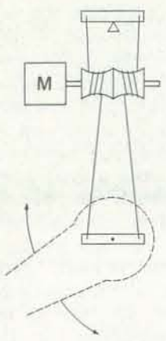


Figure 3

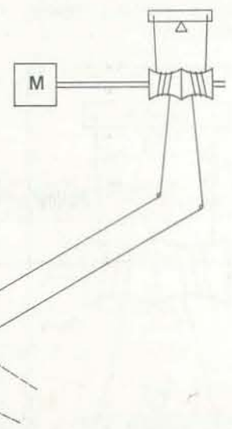


Figure 4

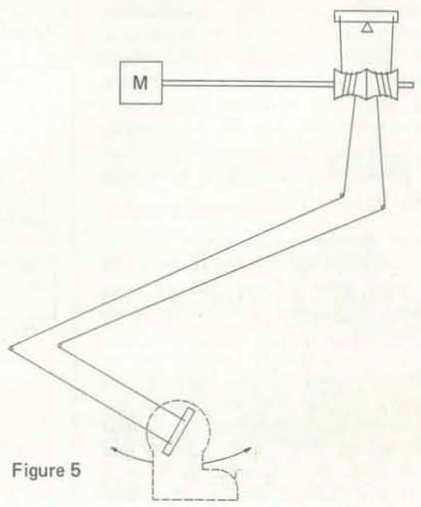


Figure 5

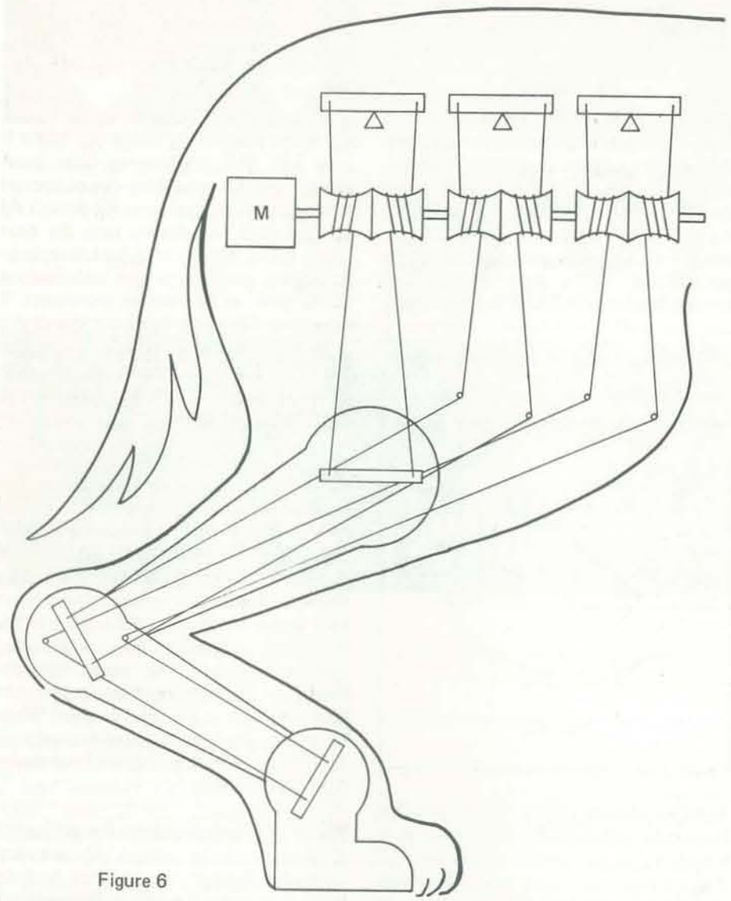


Figure 6

Commentators at the 1976 Olympics talked a great deal about the improbability that a tensed-up sprinter could get away from the starting blocks in as short a time as 10 milliseconds. That's enough time for 10,000 cycles of operation of a one-megahertz computer. Real-time operation seems like no great problem, especially since the robots may not need the coordination of Olympic athletes for some time.

Examining this proposed system, one enthusiast commented: 'It would be the essence of simplicity to make the robot tap dance!' The remark was followed by a strong remonstrance from Glenn Norris, president of USRS, who has made a career out of conquering hideously difficult projects, producing things that wild-eyed inventors regard as trivially easy. After Norris's heated rejoinder, the enthusiast said, 'Let me rephrase that. With enough effort, it could be made to tap dance.'

The fact is that dogs, from whom this leg design is borrowed, are seldom good dancers. Certainly, that's not only due to lack of interest, but also to the design of the leg, which has fewer degrees of freedom in its motion than does the human leg or arm. Even so, the dog can do a number of practical things that the average robot can't do. For example, a dog can walk around a hill without toppling over or walk up a flight of steps.

He can step over obstacles, squeeze between objects, squirm under fences. A robot might do worse than to emulate the dog. But by using this scheme, many configurations of the robot are possible.

The first important consideration is how many legs your robot needs. It's generally true that the larger and heavier an animal is, the less likely he is to have all four feet off the ground at once. An elephant moves fast, but he always has two feet on the same side on the ground at the same time. Smaller animals can more safely use a variety of gaits. While walking animals move their legs in synchrony much of the time, they tend not to move them in phase to avoid oscillation that could flip them right over.

Bugs with eight legs tend to turn like caterpiller tractors, stopping movement on one side while continuing movement on the other to cause pivoting. Of course, an animal with legs all around might be able to rotate smoothly around the center point of its body as wheeled robots (consider Papert's Turtle, Hollis's Newt) often do.

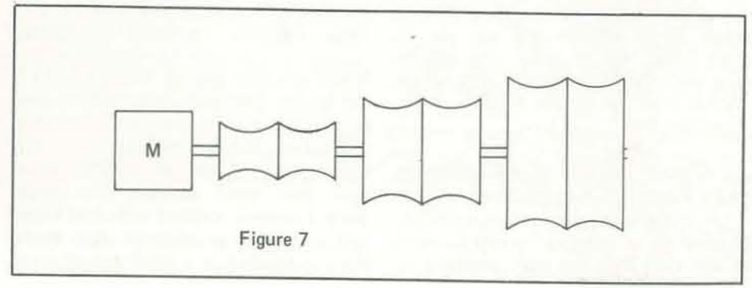


Figure 7

Legs-all-around may not be too difficult to construct. We've already discussed the use of a flexible shaft on our main motor. It may be possible to run that shaft all the way around so that it connects with the motor again at the other end (Figure 8). Pulleys could be arrayed along the circular shaft so that power could be pulled off at any point. But how many legs and how many mechanisms can be fitted along the loop?

Operating limbs can reach out in any direction from the main body of the creature. If you lean strongly to emulation of the dog, you may decide to give the beast a waggable tail. That would require only a single double-windlass mechanism. To give the tail more expressiveness, add another double-windlass so it can move up and down as well as back and forth. A dog with its tail between its legs is ordinarily indicating submission. You may want your robot to appear submissive when it's being scolded. The ability of a robot to express gloom as well as pleasure can prove extremely useful to its handler, who needs signs that let him anticipate what the robot will do next. Clues to 'emotional state' are extremely useful.

In past articles copyrighted by the USRS, robots have been evaluated as household pets; the point being that if robots are to be welcome in society, they must display features and characteristics that society has already accepted. Since household pets are clearly acceptable, roboticists may well study them in detail and make use of their physical and behavioral characteristics. For example, animals whose hooves scratch hardwood floors and damage carpets aren't often invited into the parlor. Animals that weigh more than 50 pounds ordinarily live outside. Those that move faster than about two miles an hour in the house are sent outside to release their energies.

Taking these things into consideration, the practical pet robot will probably weigh 30-90 pounds, will use one to three cubic feet of space, and will be more spherical than otherwise. This configuration suits the double-windlass mechanism very well.

An engineer listening to this discussion might suggest off the top of his head that a 1/6 to 1/4 horsepower motor is probably adequate to move such a robot around at not more than two mph, assuming no great acceleration demanding extra surges of energy is required. Moving upstairs and downstairs should be all the same to a machine designed to be in no particular hurry.

If you're eager for robots that slither or creep instead of walk, this same approach should work. Aquatic creatures with flippers and arboreal types with arms designed for tree-climbing are very similar. A flying robot with flapping wings seems improbable, considering the aeronautical failure with ornithopters, but the field is wide open to experiment.

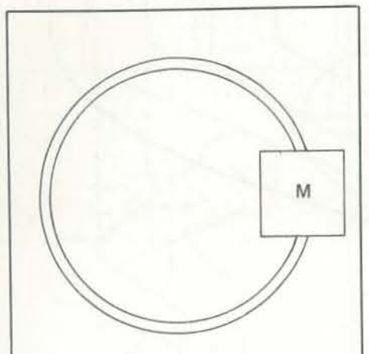


Figure 8

A major feature of the double-windlass is its push-pull capacity. It not only pulls a lever up in one direction, but can also push it back the other way. This reciprocating movement is very important to animals; Nature makes sure that they can back out of trouble as well as push into it, thus adapting to new situations. To survive, the robot must be able to put its foot down as well as pick it back up without cycling through a full rotation movement. Control of this action must be in the 'brain' of the robot, in its logic circuitry. Again, this involves philosophy and logic beyond the scope of this article, but the mechanical design must not preclude solutions to logical problems.

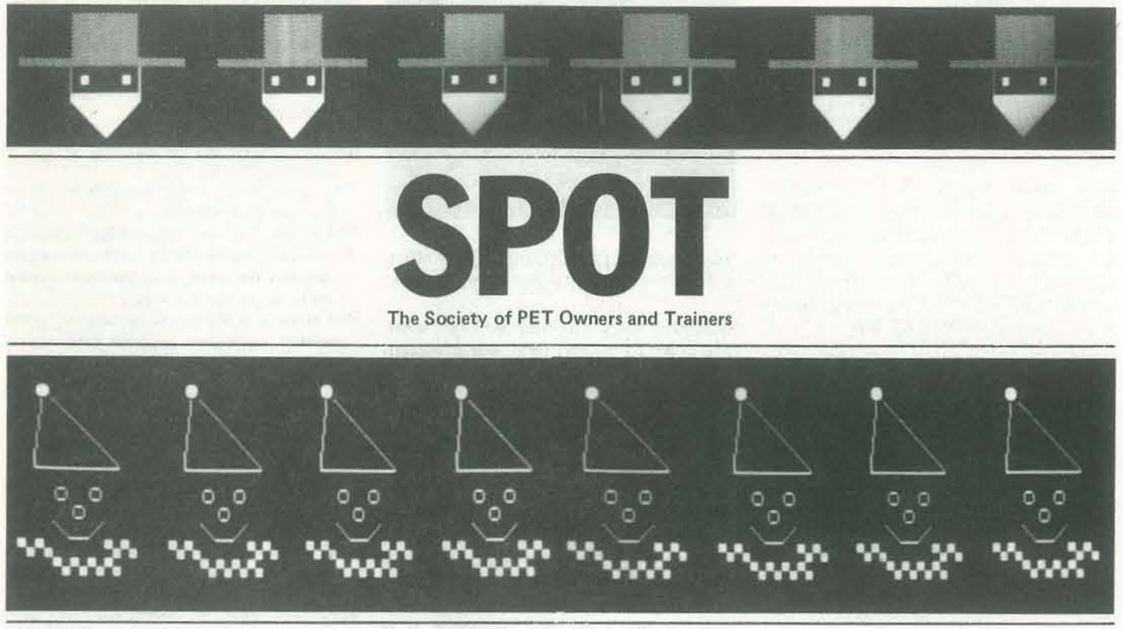
Briefly consider one of Nature's tricks—the human knee-jerk reflex. When you get a physical exam, the doctor taps just under your kneecap with his silly little rubber hammer and is gratified when your knee jerks. In fact, your brain doesn't become involved with that knee-jerk, except as an observer. The whole thing is handled in a subcircuit of your nervous system. You receive a sharp stimulus to a sensitive point in your knee, and your system responds by yanking on your lower leg. (If something is biting you, or you are being burned, this reflex tends to pull you out of danger.)

When the reaction is complete, where is your leg? Well, it's hanging there, loose again, ready for something else of interest to happen. Note that your leg doesn't fly up and kick the doctor into the next room. It also doesn't snap back sharply to its original position, or lock into position at the peak of its reaction movement. It does something your brain can detect as a protective move and then stops the reaction. Then it's ready for the next action, relaxed and not committed to anything in particular.

The 'nervous system' in the robot may be designed with similar reflexes. The mechanical system must be able to work in accordance with the nervous system. This leads to questions about 'normal' positions for limbs. A horse ordinarily sleeps on his feet, because he has a leg at each corner to keep him balanced, and his legs lock normally into a standing position. Similarly, he sleeps with his head erect. He must expend energy to put his head down to the grass to graze. When he relaxes, his head is pulled naturally up to a level at which he can see most things that attack horses.

What's the normal position for the robot? A design can be chosen to suit any rationale. Springs can be used to hold limbs in a 'normal' position. However, it's not necessary to use a single spring to hold a leg in position. Doing so expends a lot of the motor's energy in overcoming that spring whenever the limb moves. Instead, you can use paired springs that hold the limb at the point of equilibrium between them. When the motor pulls the limb, it fights one of the springs as it releases energy stored under tension. This wastes some energy, but the overall cost may be far outweighed by the savings in maintaining a normal 'power-off' position, which is better than having your robot collapse in a heap on the floor.

With small motors, pulleys, springs, and levers, your construction of a satisfactory mobile system at modest cost is possible. The patter of little feet around the house may soon be produced not by kids and cats, but walking machines trying their new legs. □



SPOT

The Society of PET Owners and Trainers

Commodore's PET is a factory assembled personal computer based on a 6502 microprocessor. The unit includes a keyboard, cassette tape unit, CRT, some graphics, upper and lower case, and an extended 8K BASIC. The system with 4K of user memory costs $595; the 8K model costs $795. Commodore has announced a 4 week delivery time for its $99.95 second cassette drive. Orders may be placed through Commodore, 901 California Ave, Palo Alto, CA 94304, (415) 326-4000.

PET fans will also be interested in the article on timing in this issue.

Photos in this article were taken of PET screens using a Polariod camera with a special hood to reduce glare.

We encourage readers to submit comments on and programs for the PET (or the TRS-80, or. . .). Please *submit only tested and debugged programs—we don't have time to debug programs here.*

SOFTWARE AVAILABILITY

Various ads for PET software have been appearing—we'd appreciate hearing from anyone who's tried any. Peninsula School software is offered under 'Software' in Announcements. We're still waiting for Commodore to begin selling software.

DOCUMENTATION

By now PET owners should have received their introductory manuals. Commodore will send, upon request, a collection of bulletins answering common questions PET owners have. A detailed PET User's Manual is now being written by Commodore; it will be sold for about $12—15. Data sheets on the chips used in the PET are available from KIM dealers.

PET NEWSLETTERS/GROUPS

In November, a professional-looking 7-page booklet bearing the title 'PET User Notes' was published by a PET enthusiast in Pennsylvania. Very general (but useful) information was provided. Future issues are scheduled to contain information on PET programs and systems offered for sale, and to devote much space to software exchange. The second of these bi-monthly publications should be out soon. To get on the mailing list, send $5.00 to PET User Notes, P O Box 371, Montgomeryville, PA 18936.

A PET Users' group is forming in Dallas to exchange ideas and information. Contact Carl Martin, 2001 Bryan Tower, Suite 3800, Dallas, TX 75201; (214) 742-5750.

Robert Elliott Purser (Box 466, El Dorado, CA 95623) is starting a reference list (to be published) of software available on cassettes for the PET, TRS-80, SOL and Apple II computers.

TAPE TIPS

At first we bemoaned the lack of a tape counter on the PET, as we suffered through hours spent searching for files on tape. But we've decided a tape counter wouldn't help that much, especially since we expect young children to use our PETs. Even with a tape counter you have to remember where on a tape a file is.

The solution we like best so far is to use short tapes—about 10 minutes long (5 minutes per side). That's enough tape to hold 2 copies of an 8K program on each side—but it's little enough tape that both sides of the tape can be used for program storage. (We gave up using more than one side of tapes longer than 15 minutes—it simply took too long to rewind.) We're pleased to announce we've come up with a source of inexpensive short tapes—see 'Cassette Data Tapes' under Other in Announcements.

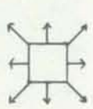
DRAW REVISITED

The PET drawing program has been expanded and modified by Larry Tesler and Dave Offen. We reproduce it here in full because portions of last issue's listing were pretty much unreadable.

How Draw Works. For new readers, we offer a brief summary of how DRAW works. For a more detailed explanation, see *People's Computers* Volume 6, Number 4; the program listed there will run on a 4K PET. The expanded version in this issue requires an 8K PET.

The program displays a dot as a drawing symbol at 'home', the center of the screen. The digit keys 1 through 9 are used to make the target move one cell in any of eight directions.

7	8	9
4	5	6
1	2	3

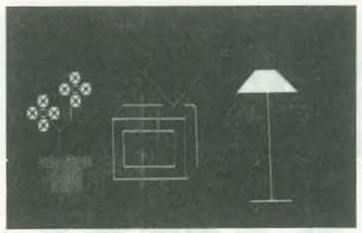


Pretend the target cell is on the '5' key. To move it left, press '4'; to move it up and right, press '9'; and so on. Whenever the target moves, it will inscribe the drawing symbol in its new cell. To change drawing symbols, press any graphic key; shift to get the graphics on keys 1-9.

RVS will reverse the drawing symbol. CLR and HOME work as usual. To erase, use SPACE. DEL erases the target cell without changing the drawing symbol. You can 'un-DEL' using the '5' key.

To move the target without changing the picture, get rid of the drawing symbol by pressing either of the CRSR keys, then use the digit keys to move the target. When you are ready to draw again, press a graphics key or RVS.

Dot Mode. The PET's graphics include small square dots which are each one-quarter size of a normal character. The 'dot mode' enables you to use the screen as a grid which can display up to 76 dots across and 48 down—twice as many symbols as the 'character' mode. To use the dots to draw, type shift->. The small square will become your drawing symbol and remain so until you press a graphic key, at which time you're back in 'character' mode. The new code used for dot



mode is in lines 4, 425, and 3100-3940; it is briefly described under 'Annotations.'

Repeating Keys. The number keys, when held down for a short time, will automatically start to repeat. For example, a line of symbols can be drawn across the screen by pressing the '6' key just once and holding it down for several seconds. The new code is in lines 25, 350, 2800–2850, and in the dot mode code. Memory location IB contains the number of characters in the keyboard input buffer. Memory location KD (discovered by Harry Saal for this purpose) contains 255 if no key is down and the row-column code of the down-key otherwise. It is possible that these locations may be different in future versions of the PET operating system.

Saving on Tape. The 8K version has been simplified to take advantage of the extra space. While the picture is being measured, a gray stripe creeps down the right side of the screen. While it is being stored into an array, the picture itself is covered with gray.

If, instead of a file name, an asterisk is typed (and a RETURN), then the picture will not be saved on tape. A BASIC program will be constructed that would print the picture left flush on the screen. That program is typed out with the word NEW above it and the cursor is put on the word NEW. If you wish to save the little printing program, press RETURN several times to erase the DRAW program and to enter each line of the printing program into memory. (You may want to change the line numbers first.) Finally, save the program the way any PET BASIC program is saved. Note: If the drawing is too wide, too high, or has too many reversals, then the printing program may not fit on the screen and none of this will work.

Lines 5510–5700 have been revised and lines 7000–7040 have been added.

Annotations for Dot Mode.

Q: quadrant number. For drawing small squares, each printing character is divided into four quadrants, numbered as shown:

1	4
2	8

QX: component of the quadrant in the X direction (0 or 1).

QY: component of the quadrant in the Y direction (0 or 1).

PT: pattern number. The pattern number is formed as the sum of all quadrant numbers required. For example, if quadrants 1 and 4 are filled, pattern 5 is used.

PN$ contains all characters comprised of small squares. Note that we print these special characters as words inside square brackets, since we have no way of printing graphic characters. Note: all characters we print inside square brackets are *shifted* (including SPACE) with the exceptions RVS, HOME, RIGHT (cursor right) and DOWN (cursor down). Lower case also requires shifting (as in line 30).

425 Enters dot mode when shift—> is typed.

3100-3140 clears screen, initializes variables, prints square in center of screen.

3150-3280 same as 100-900, for small squares.

3300-3330 same as 3000-3030, for small squares.

3400-3420 same as 4000-4020, for small squares.

3500-3520 same as 4500-4520, for small squares.

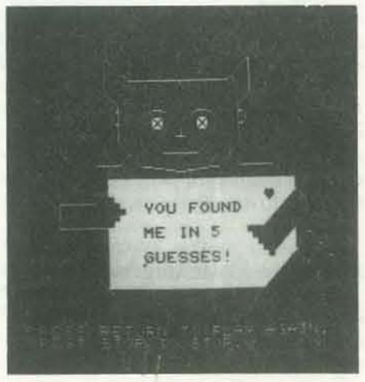
3600-3700 moves small square in direction indicated by number key.

3710-3780 reads current character from screen buffer so that other small squares within the same character will remain unchanged.

3790-3810 checks if target small square should be filled in (RV>0), blanked-out (RV<0) or left the same (RV=0).

3910-3915 replaces character on screen with pattern of small squares and blinks target square if necessary.

3920-3940 if number key is still down, moves again.




```
1 REM "DRAW 8K" FOR 8K PET
2 REM  COPYRIGHT 1977, 1978
       PENINSULA SCHOOL, MENLO PARK, CA
3 REM  PERMISSION TO USE, BUT NOT TO SELL
4 PN$="[OFF,SPACE,OFF,>,OFF,:,OFF,!,OFF,<,
  RVS,",RVS,?,RVS,COMMA,OFF,COMMA,
  OFF,?,OFF,",RVS,<,RVS,!,RVS,:,RVS,>,RVS,SPACE]"
5 LS="": I=0: POKE 59468,12
6 SC=32768: SP=32: SS=160: CV=191
7 BY=255: SH=120: US=127: QTS=CHR$(34)
8 LX=1: MX=38: LY=0: MY=24: DIM ES(MY)
9 HS="[LEFT]"+CHR$(0)+"[RIGHT]"
10 VS="[UP]"+CHR$(0)+"[DOWN]"
15 N1$="1": N9$="9": CR=13: RE=18: DE=20
20 WH=50: BL=53: WT=5: XR=164: IR=146
25 I1=525: KD=515
30 GRS="q": PRINT "[CLR]";
40 Y=INT((MY+LY)/2): X=INT((MX+LX)/2)
45 R=IR
50 PRINT "[HOME]";LEFT$("[12 DOWN]",Y);SPC(X);
55 IF PEEK(KD)<BY GOTO 55
60 GOTO 2300
100 GET CS: IF CS="" GOTO 3000
150 C=ASC(CS) AND US
200 IF FL>WH THEN GOSUB 4000
250 IF C=CR THEN FL=-1E8: GOTO 100
300 FL=WH-WT
350 RP=0
400 IF CS>=N1$ AND CS<=N9$ THEN 1700
425 IF CS="[>]" GOTO 3130
450 IF C>=SP GOTO 1000
500 IF C=DE GOTO 4500
600 IF C=RE GOTO 1200
700 IF CS="[DOWN]" OR CS="[RIGHT]" THEN R=IR: GRS=""
750 IF CS="[HOME]" THEN GRS="": GOTO 40
800 IF CS="[CLR]" GOTO 30
850 IF CS="[LEFT]" GOTO 5000
875 IF CS="[UP]" GOTO 6000
900 GOTO 100
1000 GRS=CHR$(C+SH): R=IR
1100 GOTO 1300
1200 IF GRS="" THEN GRS=PGS: R=PR
1250 R=XR-R: PRINT CHR$(R);
1300 GOSUB 1400: GOTO 100
1400 PRINT GRS;"[LEFT]";
1500 PGS=GRS: PR=R
1600 RETURN
1700 PX=X: PY=Y
1750 X=X+C+1-3*INT((C+2)/3)
1800 IF X<LX THEN X=LX
1900 IF X>MX THEN X=MX
2000 Y=Y+1-INT((C-49)/3)
2100 IF Y<LY THEN Y=LY
2200 IF Y>MY  THEN Y=MY
2250 PRINT MID$(HS,X-PX+2,1);
2275 PRINT MID$(VS,Y-PY+2,1);
2300 L=SC+40*Y+X
2325 PGS=CHR$(PEEK(L) OR SH)
2350 PR=IR-(PEEK(L) AND SH)
2400 IF GRS<>"" THEN GOSUB 1400: GOTO 2800
2500 GOSUB 4000
2600 FOR DL=1 TO WT: NEXT DL
2700 GOSUB 4000
2800 IF PEEK(IB)<>0 OR PEEK(KD)=BY GOTO 100
2850 RP=RP+1: IF RP=1 THEN FOR DL=1 TO 200: NEXT DL:
                GOTO 2800
2900 GOTO 1700
3000 FL=FL+1
3010 IF FL=WH THEN GOSUB 4000
3020 IF FL=BL THEN FL=0: GOSUB 4000
3030 GOTO 100
3100 PRINT "[CLR]";
3110 Y=INT((MY+LY)/2): X=INT((MX+LX)/2)
3120 PRINT "[HOME]";LEFT$("[12 DOWN]",Y);SPC(X);
3130 RV=1: C=53: Q=1: QX=0: QY=0
3140 GOTO 3600
3150 GET CS: IF CS="" GOTO 3300
3160 C=ASC(CS) AND US
3170 IF FL>=WH THEN GOSUB 3400
3180 IF C=CR THEN FL=-1E8: GOTO 3150
3190 FL=WH-WT
3195 RP=0
3200 IF CS>=N1$ AND CS<=N9$ GOTO 3600
3210 IF C>=SP GOTO 425
3220 IF C=DE GOTO 3500
3223 IF C=RE THEN RV=(-RV) OR 1: GOTO 3800
3227 IF CS="[DOWN]" OR CS="[RIGHT]" THEN RV=0
3230 IF CS="[HOME]" GOTO 3110
3240 IF CS="[CLR]" GOTO 3100
3250 IF CS="[LEFT]" GOTO 5000
3260 IF CS="[UP]" GOTO 6000
3270 IF C>=SP GOTO 1000
3280 GOTO 3150
3300 FL=FL+1
3310 IF FL=WH THEN GOSUB 3400
3320 IF FL=BL THEN FL=0: GOSUB 3400
3330 GOTO 3150
3400 PT=PT+Q-2*((PT) AND Q)
3410 PRINT MID$(PNS,PT+PT+1,2);"[LEFT]";
3420 RETURN
3500 PT=(PT) OR Q
3510 GOSUB 3400
3520 GOTO 3150
3600 XC=C+1-3*INT((C+2)/3): YC=1-INT((C-49)/3)
3605 PX=X: PY=Y
3610 XT=X+X+QX+XC
3620 X=INT(XT/2): QX=(XT) AND 1
3630 IF X<LX THEN X=LX: QX=0
3640 IF X>MX THEN X=MX: QX=1
3650 YT=Y+Y+QY+YC
3660 Y=INT(YT/2): QY=(YT) AND 1
3670 IF Y<LY THEN Y=LY: QY=0
3680 IF Y>MY THEN Y=MY: QY=1
3690 Q=2*(QX+QX+QY)
3700 PRINT MID$(HS,X-PX+2,1);MID$(VS,Y-PY+2,1);
3710 L=PEEK(SC+40*Y+X)
3720 LS=CHR$((L AND 63)+SH)
3725 PT=0
3730 FOR I=2 TO 16 STEP 2
3740 :IF MID$(PNS,I,1)=LS THEN PT=I/2-1: I=16
3750 NEXT I
3760 IF PT=0 GOTO 3800
3770 IF PT>=5 THEN PT=15-PT
3780 IF L AND SH THEN PT=15-PT
3790 IF RV=0 THEN GOSUB 3400: GOSUB 3400: GOTO 3920
3800 IF RV THEN PT=(PT) OR Q
3810 IF RV<0 THEN PT=PT-Q
3910 PRINT MID$(PNS,PT+PT+1,2);"[LEFT]";
3915 IF RV=0 THEN GOSUB 3400: GOSUB 3400
3920 IF C=RE OR PEEK(IB)<>0 OR PEEK(KD)=BY GOTO 3150
3930 RP=RP+1: IF RP=1 THEN FOR DL=1 TO 200: NEXT DL:
                GOTO 3920
3940 GOTO 3605
4000 PR=XR-PR: PRINT CHR$(PR);
4010 PRINT PGS;"[LEFT]";
4020 RETURN
4500 PGS=" ": PR=IR
4510 PRINT "[OFF] [LEFT]";CHR$(R);
4520 GOTO 100
5000 X0=MX: X1=LX: Y0=MY: Y1=LY: K=SC
5010 FOR Y=LY TO MY
5020 :FOR X=0 TO 39
5030 ::C=PEEK(K):K=K+1
5050 ::IF C=SP GOTO 5100
5060 ::IF X<X0 THEN X0=X
5070 ::IF X>X1 THEN X1=X
5080 ::IF Y<Y0 THEN Y0=Y
5090 ::IF Y>Y1 THEN Y1=Y
5100 :NEXT X
5110 :POKE K-1,102
5130 NEXT Y
5510 FOR Y=Y0 TO Y1
5512 :K=SC+40*Y+X0
5515 :RV=0: LS=""
5520 :FOR X=X0 TO X1
5525 ::C=(PEEK(K)-SP AND CV)+SS
5530 ::POKE K,102
5535 ::V=C>BY: IF V=RV GOTO 5545
5540 ::RV=V: LS=LS+MID$("[RVS, OFF]",V+2,1)
5545 ::K=K+1: LS=LS+CHR$(SH*RV+C)
5600 :NEXT X
5615 :ES(Y)=LS:
5620 NEXT Y
5630 INPUT "[CLR]SAVE FILE NAME OR *"; NM$
5635 IF NM$="*" GOTO 7000
5640 OPEN 1,1,1,NM$
5650 PRINT#1, Y1+1-Y0
5655 PRINT#1, X1+1-X0
5660 FOR Y=Y0 TO Y1
5665 :PRINT#1, QTS; ES(Y); QTS
5670 :ES(Y)=""
5675 :POKE 59411,53: T=TI
5676 :IF TI-T<3 GOTO 5676
5677 :POKE 59411,61
5678 NEXT Y
5690 CLOSE 1
5700 GOTO 30
6000 OPEN 1
6005 INPUT#1,Y1
6010 INPUT#1,X1
6015 IF ST GOTO 6100
6020 Y0=INT((MY+LY+1-Y1)/2)+1
6030 X0=INT((MX+LX+1-X1)/2)+1
6040 PRINT MID$("[CLR, 12 DOWN]",1,Y0);
6050 FOR Y=1 TO Y1
6060 :INPUT#1,LS
6070 :PRINT SPC(X0);"[LEFT]";LS;
6080 :IF Y<=MY THEN PRINT
6090 NEXT Y
6100 CLOSE 1
6110 GRS="*"
6120 GOTO 40
7000 PRINT "[CLR, 2 DOWN]NEW"
7010 FOR Y=Y0 TO Y1
7020 :PRINT 901+Y-Y0;"?";QTS;ES(Y)
7030 NEXT Y
7040 PRINT"[HOME]";: END
```

PET PROJECTS & PROPOSALS

Lud Braun, developer of the Huntington Project simulations, has a PET. We're pleased that he sent us a copy of the letter he sent to Commodore so that we can share his reactions and activities with our readers.

I have had my PET for about a month now and thought that I should write down some of my observations about it. Basically, I am *very* pleased with it but, in typical human fashion, there are things I'd like to see improved. Among them are:

1. For educational use there should be a composite video signal coming out of a UHF or BNC Connector (not the odd-ball one that Radio Shack has used). This permits the teacher to bring the PET into the classroom to work with his entire class. I suggested this to you when I visited in early August and had hoped that it would have been added. We have designed and installed a mixer etc. and now have a composite video signal on old #77. It takes about $2 worth of parts.
2. The PET should have a handle on it. One of the first things I did after I got ours was to go to the hardware store to buy a 99¢ handle. It is very helpful when I carry the PET into class or home or onto an airplane. (You should see the airport security people freak out when I present this strange looking TV set for inspection!) I'm convinced that kids will be borrowing PETs over weekends from school or signing them out from the local public library, but they need a handle.
3. We have designed and installed a simple 4-channel A/D converter to permit us to enter parameter values into programs during execution of a simulation. This capability has enormous potential in education. It permits the digital computer to act like an analog computer (but with none of the problems of the real analog computer) and permits enormously improved simulation experiences. This also costs about $2 in parts.
4. We have had 6-10 system crashes in the month we've had it. We haven't been able to identify the cause although I suspect that it may be a voltage spike riding in on the line. The system suddenly hangs up and the keyboard goes dead. It had always been easy to recover just by turning the PET off and on, but then the memory is zeroed and any program I was devel-

oping has disappeared. Is it possible to restart without zeroing memory? Such a capability would sure be nice.

5. The monitor doesn't have equal gain horizontally and vertically. As a consequence, when I draw a circle (in a math program) it comes out an ellipse. A square (equal numbers of horizontal and vertical elements) comes out a rectangle. I have found a height adjustment in the monitor but no width control. Is there one? If so, I could fix it. (Incidentally, such controls probably should come out the back.)
6. It really would be nice to be able in one vertical and one horizontal command to send the cursor to any place on the screen (e.g., POKE 245, V). With this, graphing would really be easy. This amounts to a two-dimensional TAB command.

Despite these criticisms I am very happy with the PET. I am very excited about this machine for educational applications because of its powerful BASIC, because of its price, and because of its portability. The latter property is especially exciting because it gives us teachers, for the first time, the opportunity to let our students take a computer home overnight or over a weekend to develop an idea through use of a simulation or to solve a problem normally beyond their capability.

Ludwig Braun
Professor and Asst. Director
for Educational Technology
State University of New York
Stony Brook, NY

PET PICTURES

I was interested to read in the November-December issue of your experiences with the PET. Shortly after receiving the issue your DRAW program was running and we are now planning a contest for art students in one of the schools in which I am consulting. Using high contrast film and a micro lens, we are able to capture the work for reproduction. It is a most interesting exerise in applying the computer in some of the less expected areas of the school curriculum.

In addition to my own PET, we are getting a number of machines for the Simulation Center in the College of Osteopathic Medicine here at Michigan State University. Having followed John Starkweather's work at San Francisco, from the early days of Computest on the 1620 to his current efforts, I was most interested to read of your development of a PET version of PILOT. Is there more information available on that implementation? Is there any way this version can be purchased?

Norman T. Bell
Director, Faculty Development Program
College of Osteopathic Medicine
Michigan State University
East Lansing, MI

We'd enjoy seeing results of your art contest–some of our PET graphics accompany this article. As for your questions on PILOT, see the 'PET Software' ad under Announcements.

POST-DOC WARRANTY?

As a new PET owner, I found your November article timely and informative. I can't speak for the other owners but for one who has an elementary knowledge of BASIC and is interested in learning more BASIC plus assembly language, the PET to date has been a let-down. Foremost is their lack of a comprehensive instruction book which treats the three PET features: BASIC, assembly language and graphics. I argue that the lack of such a book does not allow me to fully test out the PET and therefore my warranty should start only after I receive my book. What do the other PET owners think?

The PET is a great machine, but there is always room for improvement. The case in point is the lack of a counter on the tape unit. Looking for programs on a cassette is a real chore. A counter would make PET even better.

I think the PET section in *People's Computers* is filling a real need, and I urge you to keep with it. We, the interested and uninformed PET owners of the world, support you.

Phillip Gash
Redding, CA 96001

A warranty that begins when you receive documentation is an idea that raises interesting questions, such as who gets to decide whether documentation is suitably 'comprehensive'? What if documentation is buggy–does this affect the warranty?

CHECKBOOK PROGRAM

```
70 REM ADD UP YOUR CHECKBOOK OR HOW TO
80 REM FORMAT FOR DOLLARS AND CENTS
90 REM BY OWEN HAWKINS
100 PRINT"[CLR]":N=1
120 INPUT "AMOUNT IS ";A
130 A=INT(A*100+.5): T=T+A
140 PRINT "[UP]";TAB(12);
150 PRINT "          ";: XS=STR$(A)
160 IF ABS(A)<10 THEN
        XS=LEFT$(XS,1)+"0"+RIGHT$(X$,1)
170 PRINT TAB(33-LEN(XS));"$";
180 PRINT LEFT$(XS,LEN(XS)-2);".";RIGHT$(X$,2)
190 IF A=0 THEN IF N THEN A=T: N=0: PRINT:
                          PRINT "TOTAL": GOTO 140
200 ON N GOTO 120
999 END
```

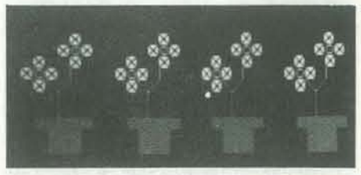
100 clears screen and sets GOTO flag N to 1.
120 rounds A to 2 decimal places and removes decimal point; computes cumulative total, T.
150 erases A on the screen and converts it to a string.
160 if A is less than 10 inserts a '0'.
170-180 prints the formatted number.
190 if finished (A was zero), then swaps T into A, sets flag to zero, and prints the total.
200 gets more numbers unless N=0.

For those of you who have not yet convinced your wife (and I suppose a husband or two) that buying the PET was a dire necessity, you might let her use the above program to add up the checkbook. It's considerably more fun than a pocket calculator and you can visually inspect your entries.

Unfortunately you are limited to numbers less than 10 million. If that presents a problem for you, I suggest that you simply do away with the cents by modifications to lines 140, 160 & 180. Now you can almost handle a billion. When you're ready for the total, enter '0' as the amount.

If you don't have a checkbook but like the formatted output, just use lines 140 to 180 in your own program. If you still don't like anything then at least take a closer look at line 190. The IF... THEN... THEN IF... etc can go fairly deep. And if you're big on commas (I'm not) the key is FROM 10 to 4 STEP –3.

Owen Hawkins
Menlo Park, CA



FROG RACE

BY B ERICKSON

Frog Race is a game based on races in which you put frogs in the center of a circle; the first frog to jump out of the circle is the winner. But the frogs don't know or care they are in a race, and may jump in any direction. In this program the frogs will jump in any direction, one jump at a time.

First you are asked 'How many players?'–there can be up to 8. Then odds will be printed out; they come pretty close to being right. Next each player is asked to pick a frog (numbered 1 to 8) and to bet (there is no limit on how much you bet).

After all bets have been made the computer prints the frogs in a circle (it's really a square). The circle and frogs are reprinted each time a frog jumps until a frog jumps out of the circle. Finally the computer prints the winning frog's number, the odds it paid, and what each player won or lost.

```
5 REM THIS PROGRAM IS FORMATTED FOR SOUTHWEST
10 REM TECHNICAL PRODUCTS CORP. 8K BASIC VERSION 2.0
15 REM USING THEIR TV TYPEWRITER II WITH THEIR CURSOR
20 REM COMMANDS OR A TELETYPE.
25 GOSUB 190:PRINT "WHAT ARE YOU USING"
30 PRINT "1=TV TYPEWRITER":PRINT "2=TELETYPE"
35 INPUT M:IF M=1 THEN 50
40 IF M=2 THEN 50
45 GOTO 25
50 DATA 2,2,3,3,4,5,8,10
55 DIM A(81),B(8),C(8),D(8),H(8)
60 LINE= 0:DIGITS=0
65 GOSUB 190:PRINT "WAIT"
70 FOR I=1 TO 81:A(I)=0:NEXT I
75 FOR I=1 TO 9:A(I)=9:A(I+72)=9:NEXT I
80 FOR I=10 TO 64 STEP 9:A(I)=9:A(I+8)=9:NEXT I
85 LET W=0:FOR I=1 TO 8:READ D(I):NEXT I
90 LET A(31)=1:A(32)=2:A(33)=3:A(40)=4
95 LET A(42)=5:A(49)=6:A(50)=7:A(51)=8
100 LET H(1)=31:H(2)=32:H(3)=33:H(4)=40
105 LET H(5)=42:H(6)=49:H(7)=50:H(8)=51
110 GOSUB 190:PRINT "HOW MANY PLAYERS ";
115 INPUT P:P=INT(P):IF P<=0 THEN 405
120 IF P<=8 THEN 130
125 PRINT "NO MORE THAN 8 PLAYERS":GOSUB 265:GOTO 110
130 GOSUB 190
135 PRINT "      FROG NO.        ODDS"
140 FOR I=1 TO 8
145 PRINT TAB(11);I;"------ ";D(I);"TO 1"
150 NEXT I:PRINT:FOR I=1 TO P
155 GOSUB 380:PRINT "PLAYER NO. ";I;"YOUR TURN"
160 PRINT "WHICH FROG DO YOU WANT TO BET ON";
165 INPUT B(I):IF B(I)>8 THEN 185
170 PRINT "HOW MUCH IS YOUR BET ";
175 INPUT C(I):NEXT I:GOSUB 190
180 GOSUB 195:GOTO 230
185 PRINT "THERE ARE ONLY 8 FROGS":GOTO155
190 PRINT CHR$(16);CHR$(22);:RETURN
195 PRINT CHR$(16);:PRINT:X=1:PRINT
200 FOR I=X TO X+8
205 IF A(I)=0 THEN PRINT "   ";:GOTO 220
```

Here is what the circle looks like (I told you it's a square).

```
+ + + + +
+ 1 2 3 +
+ 4   5 +
+ 6 7 8 +
+ + + + +
```

Remember that all input to the computer must be terminated by a carriage return.

This program is formatted for Southwest Technical Product Corporation's 8K BASIC, Version 2.0, using their cursor commands. It will also work on a teletype. This is a test program, so you can see if my programs run on your system. The program uses the commands I use in the rest of my programs–if this runs, they all run, if you have enough memory. I sell computer games in Standard BASIC as well as SWTPC BASIC.

B Erickson, P O Box 11099, Chicago IL 60611 □

```
210 IF A(I)=9 THEN PRINT " + ";:GOTO 220
215 PRINT " ";A(I);
220 NEXT I:PRINT:X=X+9:IF X>=81 THEN RETURN
225 GOTO 200
230 IF W=1 THEN 325
235 FOR I=1 TO 8
240 GOSUB 270
245 IF A(H(I)+X)=9 THEN 315
250 IF A(H(I)+X)<>0 THEN 240
255 LET A(H(I)+X)=I:A(H(I))=0:H(I)=H(I)+X
260 NEXT I:GOTO 180
265 LET K=2↑2:RETURN
270 LET X=INT(8*RND(0))+1
275 IF X=1 THEN X=-9:RETURN
280 IF X=2 THEN X=-8:RETURN
285 IF X=3 THEN X=-1:RETURN
290 IF X=4 THEN X=1:RETURN
295 IF X=5 THEN X=8:RETURN
300 IF X=6 THEN X=9:RETURN
305 IF X=7 THEN X=10:RETURN
310 LET X=-10:RETURN
315 LET J=I:W=1:A(H(I)+X)=I:A(H(I))=0
320 LET H(I)=H(I)+X:GOTO 180
325 GOSUB 265:PRINT:GOSUB 190
330 PRINT "FROG NO.";J;"WON--PAID ";D(J);"TO 1"
335 PRINT :PRINT "PLAYER";PRINT
340 FOR I=1 TO P:PRINT "NO.";I;:IF B(I)=J THEN 350
345 PRINT "YOU LOST ";C(I);"DOLLARS":GOTO 355
350 PRINT "YOU WON ";C(I)*D(J);"DOLLARS"
355 NEXT I
360 PRINT :PRINT "ANOTHER RACE YES OR NO ";:INPUT Q$
365 IF Q$="YES" THEN RESTORE:GOTO 65
370 IF Q$="NO" THEN 400
375 GOTO 360
380 IF M=2 THEN RETURN
385 PRINT CHR$(16);
390 FOR K=1 TO 10:PRINT CHR$(10);:NEXT K
395 PRINT CHR$(22); RETURN
400 PRINT
405 GOSUB 190:PRINT "BYE FOR NOW"
410 LINE= 48:DIGITS=0:END
```

IBM 370 MODEL 69 FEATURES AND DEVICES

(Downward compatible with IBM 360/69)*

This document appeared out of thin air—we kindly thank A.Nonymous for contributing it for our edification.

INPUT-OUTPUT

Early Care Lace
Feed Card and Jam
Read Card and Scramble Data
Backspace Card Reader
Rewind Card Reader
Backspace Disk
Read Print and Blush
Eat Card
Update in Place on Card
Read Invalid Data
Write Invalid Data
Erase Card Punch
Punch Disk
Punch Operator
Fruit Punch
Read Chaos
Read Unhappy Macnam
Stacker Upset
Print and Smear
Forms Skip and Run Away
Scatter Print
Print and Break Chain
Print and Cut Ribbon
Stacker Select Disk
Rewind and Break Tape
Stretch Tape
Make Tape Invalid
Write Wrong-Length Record
Change Tape Density in Mid-Record
Switch to Zero Density
Write Past End of Tape
Read and Write while Ripping Tape
Write Noise Record
Write New Hit Record
Read Inter-Record Gap
Update and Erase Record
Slip Disk
Seek Record and Scar Disk
Hide and Seek
Eject Disk
Write to Protected File
Read Count Key and Garbage
Write Count Key and Garbage
Garbage Count Key and Read
Burst Selector Channel
Scatter Multiplexor Channel
Skip to Random Channel
Scramble Channel
Change Channels
Reverse Drum Immediate
Snare Drum
Sharpen Light Pencil
Random Access Card I/O

ARITHMETIC

Accumulate Trivia
Triple-Pack Decimal
Add Improper
Add and Reset to Zero
Subtract and Reset to Zero
Multiply and Lose Precision
Divide and Overflow
Divide and Conquer
Abnormalized Floating Point
Sinking Point Arithmetic
Sliding Point Arithmetic
Vanishing Point Arithmetic

DATA MANIPULATION

Move and Lose Record
Move and Wrap Core
Move Continuous
Move Devious
Gulp and Store Bytes
Burp and Clear Bytes
Move Bowels
Move and Drop Bits
Circulate Memory
Convert to Garbage
Load and Clear Core
Memory Bank Holdup

SPECIAL AND CUSTOM FEATURES

1401 Incompatibility
407 Emulation
370 Emulation
370 Immolation
Read-In Only Storage
Erase Read-Only Storage
Chinese Character Set
Execute Invalid Op-Code
Concoct Data
Pessimising Software
Random Bug Generator
(Plant Installation Only)
Uncouple CPUs and Branch
Virtueless Memory
Memory Prosthesis
Memory Left Shift and Branch
Reduce Throughput
Convert to Roman Numerals (Italy Only)
Alfred E. von Neumann Architecture:
Shiftless Registers
CNIL Memory
Chocolate Chips
Trivalent Bits

LOGIC AND CONTROL

Illogical OR
Illogical AND
Why Immediate
Branch on Index Missing
Branch and Loop Continuous
Branch on Programmer Debugging
Lose Message and Branch
Develop Ineffective Address
Transfer and Lose Return
Branch on Power Off
Swipe "Emergency Pull" Knob
Branch on Burned-Out Indicator
Branch on Blinking Indicator
Branch on Bug
Bug on Branch
Branch on CE Ground
Halt and Catch Fire (Privileged Op)
Reinitialize Meter
Branch on Chip Box Full
Branch on Phase of the Moon
Branch on Donder on Blitzen
Load Operator
Byte Operator
Execute Operator
Ignore Supervisor Call
Call Supervisor Names
Trap Secretary and Halt
Byte and Run
Destroy Storage Protect Key
Scramble Program Status Word
Pack Program Status Word
Electrocute DP Manager and Branch
Inquire and Ignore
Reverse Parity and Branch
Branch on Operator Sleepy
Branch on Operator Desperate
Branch and Disconnect Memory
Invert Record and Branch
Evacuate Memory
Generate Machine Check
Generate Machine Check and Cash
Generate Machine Check and Bounce
Byte Baudy Bit and Branch

*Upward compatible with other 370 models.
Crossword compatible with other 360's.
Awkward compatible with earlier systems.

ANNOUNCEMENTS

□■□■□■□■□■□■□■□■□■□■□■□

HARDWARE

□■□■□■□■□■□■□■□■□■□■□■□

NEW TEI SYSTEM

TEI, Inc., has announced the MCS-PT112/32, a self-contained computer system with display, disk storage, keyboard and a 12-slot motherboard. It may be used either as a stand alone processor or as a processor terminal in a larger system.

Features include a 15" monitor, an upper and lower case ASCII keyboard with eight user designated special function keys and a 16-key numeric pad. One Shugart SA-400 mini-floppy disk drive is standard.

The mainframe contains an 8080 CPU board and a circuit that implements a start up 'jump to' routine to any user selected byte address. 32K static RAM memory is provided with additional RAM optional. The disk controller will handle three mini-drives. The video controller board uses a 24 x 80 format, and the I/O board provides three parallel and three serial ports with selectable baud rates. Outputs are RS-232C and TTL. Software provided with the system includes CP/M operating system and SuperBASIC, a 20K interpreter.

The MCS-PT112/32 fully assembled and tested is priced at $4795.00. Contact CMC MARKETING CORP, 5601 Bintliff, Suite 515, Houston, TX 77036 (713) 783-8880.

□■□■□■□■□■□■□■□■□■□■□■□

S-100 PET ADAPTER

HUH Electronic Music Productions has announced the PET'S 100 – a PET to S-100 Bus interface board. This S-100 sized card plugs into the mainframe of your choice and a cable connects it to your PET which then enables you to use the wide range of peripheral and memory cards available for the S-100 bus. The PET'S 100 emulates the true S-100 bus including DMA, both read and write wait states, I/O address mirroring, multiplexed status lines and much more. This means you can use Dazzlers, Byte-savers, slow memory (like 1702s), analog interfaces and a host of other tricky cards.

The PET'S 100 will be available in kit or assembled form for $199.95 or $279.95 respectively. Deliveries are scheduled to begin in April. Contact HUH Electronic Music Productions, P O Box 259, Fairfax, CA 94930 (415) 457-7598.

□■□■□■□■□■□■□■□■□■□■□■□

THE WRITEHANDER tm

A new typing keyboard has been designed that permits typing all 128 characters of the ASCII code with one hand. To use the Writehander tm, the typist places his four fingers on four press-switches and his thumb on one of eight press-switches. The four finger-switches operate as the lower four bits of the seven-bit ASCII code, selecting the group of characters (out of 16 groups) that contains the desired character. The group contains a choice of eight letters, numerals, symbols, etc. The thumb then presses the particular switch that selects the desired character from the choice of eight.

The Writehander does not require a computer to operate a terminal; it will directly operate terminals such as the Diablo HyType, Teletype ASCII-modified Selectric, or a video monitor that accepts parallel 7-bit ASCII signals.

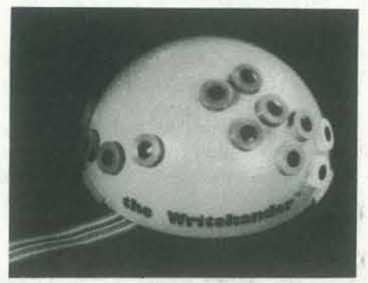
The Writehander is manufactured by the NewO Company of Palo Alto, CA, and sells for $98. Contact Mr Sid Owen at NewO, 246 Walter Hays Dr, Palo Alto, CA 94303; (415) 321-7979.

□■□■□■□■□■□■□■□■□■□■□■□

SS-50 MINIFLOPPY

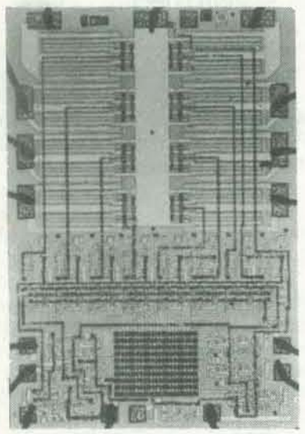
PerCom Data Company has introduced the LFD-400, a minifloppy disk system for the SS-50 bus. A 1-drive LFD-400 system includes a controller PC board, PROMware disk operating system, disk drive and drive power supply, interconnecting cable, two minidiskettes, an operator's manual, and an enclosure to house the drive and drive power supply. 2- and 3- drive systems are also available.

The controller board includes low-voltage-drop regulators, a proprietary 'bit shifting' compensation circuit, an inactivity time-out circuit to increase drive motor life, and provision for 3K bytes of PROM. The LFD-400 PROMware DOS, miniDOS tm, allows SS-50 bus owners to use their existing software with simple patches. The program includes load and save routines, and permits 'crash-proof' data storage and retrieval since the disk may be protected. It is contained in a 2708 EPROM, and is ready on power-up. Contact PerCom Data Company, Inc., 318 Barnes, Garland, TX 75042; (214) 276-1968.



EPROM PROGRAMMER

Smoke Signal Broadcasting announces a 2708 EPROM programmer. The POP-1 lists for $149 and is designed to interface to the company's P-38-1 and P-38-FF EPROM boards, which are SS-50 bus compatible products. Software is provided on audio cassette. An adaptive programming technique is used that allows most 2708's to be programmed in 15 seconds instead of the usual one and a half minutes. A self-contained power supply is used for the programming voltage, insuring sufficient current capability to program EPROM's from any manufacturer. Contact Smoke Signal Broadcasting, P O Box 2017, Hollywood, CA 90028; (213) 462-5652.



The chip layout of Signetics' new addressable peripheral driver integrated circuit.

DEVICE CONTROLLER

Two addressable peripheral driver integrated circuits which provide a means of triggering relays, lamps, LED displays and step motors incorporated into microprocessor-based systems have been developed by Signetics.

Designated the NE590 and NE591, the components are high-current latched drivers with 8 Darlington power outputs, each capable of 250 mA load current. They are similar in function to Signetics' 9334 address decoder, with which the NE590 is pin compatible.

The NE590 and 591 provide a simple replacement for the complicated circuitry, discrete transistors and Darlingtons presently required to turn on relays, lamps, LEDs and similar devices with commands from a microprocessor. In quantities of 100, the NE590N is priced at $1.95 and the NE591N at $2.45. Contact Signetics, 811 East Arques Ave, P O Box 9052, Sunnyvale, CA 94086; (408) 729-7700.

□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□

SOFTWARE

□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□

PET SOFTWARE

A PILOT interpreter, an Adventure-like game, our latest DRAW, and several other programs are now available for 8K PETs. For a licensing agreement and price list send a stamped self-addressed envelope to Computer Project, Peninsula School, Peninsula Way, Menlo Park, CA 94025.

□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□

6800 OBJECT CODE RELOCATOR

Technical Systems Consultants, Inc, is releasing a machine code relocator for the 6800. This program gives one the capability of moving assembly language programs from one area in memory to another. A feature is included which allows loading a Motorola Mikbug format tape directly into any part of RAM. This means programs located on tape where no RAM is available may still be loaded. Use of the relocator requires a knowledge of where the program to be moved starts and ends and all places in the program which contain data as opposed to executable code. All references to locations outside a range specified by the user will be left unchanged so that calls to monitor routines or other external routines will be properly relocated.

The 6800 Relocator requires just over 1K of RAM starting at 0200 hex, but since the program can relocate itself, it can be moved to any location. The relocator is self prompting and thus simple to use. The price of $8.00 includes a commented source listing, object code listing and a user's manual giving samples of use of the package. Contact TSC, P O Box 2574, W Lafayette, IN 47906.

□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□

OSI WORD PROCESSOR

Ohio Scientific has announced a new Word Processor. The OS-WP1 is a text editor which operates at both the character and line levels. It has internal GET and PUT file commands which transfer individual files from memory to disk. A full set of printer control commands can be used with virtually any impact or matrix computer printer or word processing printer. The formatted output mode allows the user to perform left and right justification of text without line numbers at a designated width of from 20 to 70 characters.

The OS-WP1 can be used directly with the Lear Siegler ADM-3A or with the Hazeltine 1500 and is adaptable to virtually any other conventional CRT terminal via documentation provided. The Word Processor package, two diskettes and a manual is now available for $79 for use on any disk-based Ohio Scientific computer system. Contact OSI, Hiram, OH 44234; (216) 569-7905.

□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□

GATHERINGS

□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□

JUNE 6-8
ANAHEIM, CA

The 1978 National Computer Conference will feature a Personal Computing Festival, to take place June 6-8 at the Disneyland Hotel complex in Anaheim, CA. A special program of papers and presentations relevant to personal computing will be presented. Both one-day and three-day registrations will be available for the Festival. Information on NCC 78 may be obtained from AFIPS Headquarters, 210 Summit Ave, Montvale, NJ 07645 or by calling (201) 391-9810.

□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□

□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□

JULY 22-23
ARLINGTON, VA

Several thousand people are expected to attend Amateur Computing 78, a July 22-23 microcomputer festival to be held at the Sheraton National Motor Hotel in Arlington, VA. Those interested in presenting a paper, participating in a panel discussion, displaying an amateur computer system or sponsoring a tutorial should submit a letter of intent along with a one-page abstract or outline by April 15 to John Wall Miller, Program Chairman, 6921 Pacific Lane, Annandale, VA 22003, telephone (703) 256-5702. This event is being sponsored by AMRAD, a technically oriented club of radio amateurs and computerists in the Washington, DC area. For further information, write AMRAD, Box 682, McLean, VA 22101.

□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□

AUG 22-25
BELLAIRE, MI

The International Conference on Parallel Processing, sponsored by IEEE Computer Society and Wayne State University, will be held August 22-25 in Bellaire, Michigan. Contact Professor G.J. Lipouski, Deptartment of Electrical Engineering, University of Texas, Austin, TX 78712.

□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□

OCT 10-12
SAN FRANCISCO, CA

The third USA-Japan Computer Conference will be held October 10-12, 1978 in San Francisco. This marks the first time this gathering is to be held on American soil. Contact Professor Edward J. McCluskey, Digital System Laboratory, Stanford University, Stanford, CA 94305.

□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□

NOV 3-5
LOS ANGELES, CA

Details on the Third Computer Faire may be obtained from Computer Faire, Box 1579, Palo Alto, CA 94302; (415) 851-7664.

□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□

COMPUTER CLUBS

CHIP'S (Computer Hobbyists in Processing—Syracuse). Computer Club with monthly meetings. For information contact: CHIP'S, c/o J A Green, General Electric Co, Court St Plant #3, Room 16, P O Box 4840, Syracuse, NY 13221.

Utah Computer Association. This club publishes *BITS*, a monthly newsletter. Meets 2nd Tuesday of each month at Murray High School, Rm 154, Salt Lake City, UT. Contact Larry or Holly Barney, 1928 S 2600 E, Salt Lake City, UT 84108.

□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□

OTHER

□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□

CASSETTE DATA TAPES

Microsette Company is selling boxed 50, 100, 200, and 300 foot data tapes, which are just slightly longer than the standard C10, C20, C40, and C60 tapes. Tapes come in packs of 4; 2 extra sets of labels are included. Prices per cassette for the 4 lengths are $.60, $.70, $.85 and $1.00. In lots of 100, there is a 10% discount; a 15% discount is offered in lots of 500 or more. An additional 10% discount is available if cash is paid for lots of 100 or more. Contact Microsette, 777 Palomar Ave, Sunnyvale, CA 94086; (415) 735-8821.

□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□

SUMMER WORKSHOP

Personal Computers for High School Science and Math Teachers, June 1978, 3 credits, Utah State University. Peter Grimes (San Jose Unified School District) is guest speaker. For information contact Conference and Institute Division, Utah State University, Logan, UT 84322.

□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□

APPLE 1 LIBRARY

The Apple 1 Software and Hardware Library is being started in Indiana to support the Apple 1 computer. Interested readers can write to Joe Torzerski, 51625 Chestnut Road, Granger, IN 46530.

□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□

PLANET CONFERENCING SERVICE

In the past several months, hundreds of people have been using a computer system to access each other's ideas and jointly manage projects. They are located throughout the Continental United States, Alaska, Canada and Western Europe. What they do could not be accomplished by mail, or by TELEX, by phone, or by facsimile. In many cases, it could not even be accomplished face-to-face. For lack of a better word, their activities are called 'computer conferencing.'

These people have demonstrated that it is possible to link human groups through computers. The link is practical and easy. It is called PLANET. Infomedia is the first commericial organization to offer such a service. The PLANET system is a carefully-engineered environment in which the exchange of ideas and information is the key. This includes a comprehensive service for documentation and training. It also includes an organization for maintenance of the system and support of the growing user community.

For information contact Rich Miller or Jacques Vallee at Infomedia Corp, 430 Sherman Ave, Palo Alto, CA 94306; (415) 321-2682.

□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□▪□

6800 HOME STUDY COURSE

Electronic Product Associates, Inc, announces that with the purchase of a basic Micro-68a you receive the complete home study course including User's Manual, 15-chapter Lab Manual, *Understanding Microprocessors*, M6800 Design Manual, the Motorola Programming Manual and the Instruction Summary Card. Total cost is $544.50. Contact Electronic Product Associates Inc, 1157 Vega Street, San Diego, CA 92110; (714) 276-8911.