Second Class Postage Paid at Menlo Park, CA
Address Correction Requested

$2.00 In Canada $2.50

Recreational COMPUTING

FOR THE IMAGINATIVE SMALL COMPUTER USER!

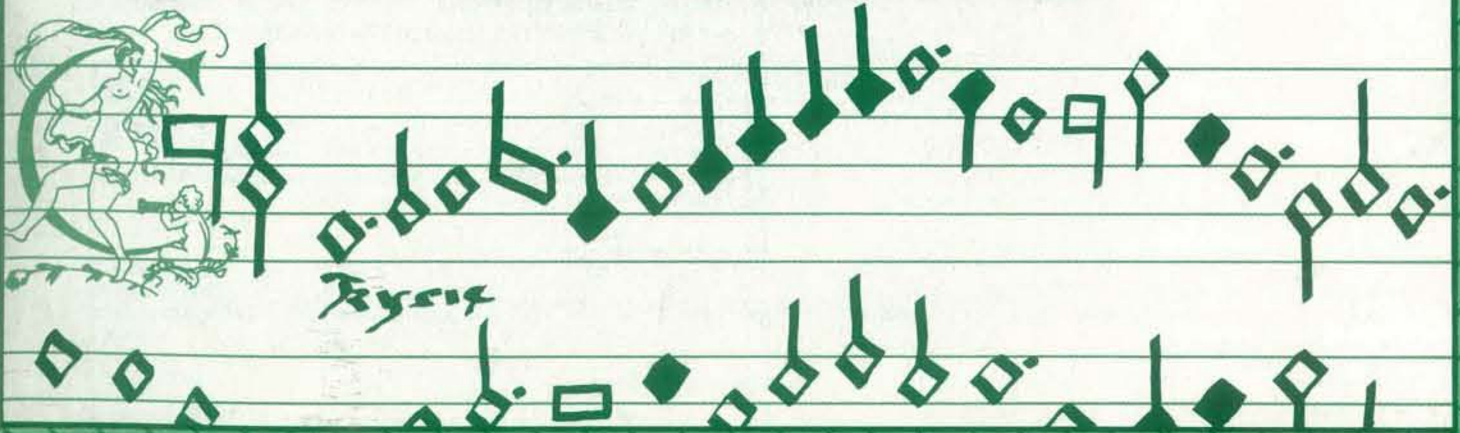
VOL. 8 NO. 6 ISSUE 45 MAY - JUNE 1980

Introduction to Computer Music

CBBS Phone Numbers

Five New Personal Computers

6502 Machine Language



Take the mystery out of programming

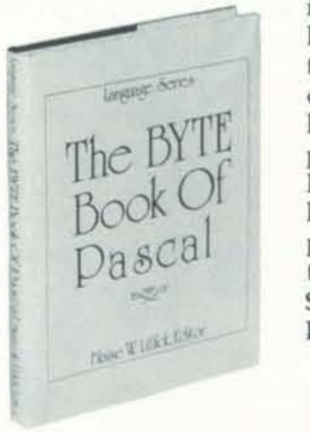
with the latest from BYTE Books™

The BYTE Book of Pascal

Edited by Blaise W. Liffick

Based on the growing popularity of Pascal as a programming language, numerous articles, language forums and letters from past issues of BYTE magazine have been compiled to provide this general introduction to Pascal. In addition, this book contains several important pieces of software including two versions of a Pascal compiler - one written in BASIC and the other in 8080 assembly language; a p-code interpreter written in both Pascal and 8080 assembly languages; a chess playing program; and an APL interpreter written in Pascal.

$25.00 Hardcover pp. 342
ISBN 0-07-037823-1



YOU JUST BOUGHT A PERSONAL WHAT?

by Thomas Dwyer and Margot Critchfield

Whether you are a novice programmer or an experienced computer user, this book is filled with practical ideas for using a personal computer at home or work. It will take you through the steps necessary to write your own computer programs, and then show you how to use structured design techniques to tackle a variety of larger projects. The book contains over 60 ready-to-use programs written in Microsoft and Level II BASIC in the areas of educational games, financial record keeping, business transactions, disk-based data file and word processing. **$11.95 pp. 256 ISBN 0-07-018492-5**

Beginners Guide for the UCSD Pascal System

by Kenneth Bowles

Written by the originator of the UCSD Pascal System, this highly informative book is designed as an orientation guide for learning to use the UCSD Pascal System. For the novice, this book steps through the System bringing the user to a sophisticated level of expertise. Once familiar with the System, you will find the guide an invaluable reference tool for creating advanced applications. This book features tutorial examples of programming tasks in the form of self-study quiz programs. The UCSD Pascal Software Systems, available from SofTech Microsystems Inc, 9494 Black Mountain Road, San Diego CA 92126, is a complete general purpose software package for users of microcomputers and minicomputers. The package offers several interesting features including:

- Programs which may be run without alteration on the General Automation or DEC PDP-11 minicomputers, or an 8080, 8085, Z80, 6502, 6800, or 9900 based microcomputers.
- Ease of use on a small, single-user computer with display screen and one or more floppy disk drives.

$11.95 ISBN 0-07-006745-7

These and other BYTE/McGraw-Hill books are available from BYTE Books or your local computer store.

Please send
☐ _______ copies of **The BYTE Book of Pascal**
☐ _______ copies of **You Just Bought a Personal What?**
☐ _______ copies of **Beginner's Guide for the UCSD Pascal System**
Add 75¢ per book to cover postage and handling.

Name _______ Title _______ Company _______

Street _______ City _______ State/Province _______ Code _______

☐ Check enclosed in the amount of $_______
☐ Bill Visa ☐ Bill Master Charge
Card No._______ Exp. Date._______

McGraw-Hill

BYTE BOOKS Division ● 70 Main Street
Peterborough, NH 03458

T50

Editors' Notes

Settling In

I cut it out of the newspaper and tacked it above my desk here at PCC—a comic strip, Shoe to be exact. The patter caught my eye: "What's a six letter word meaning odious reptile?" I tried 'Editor' and it didn't work. True words, I've learned already, and I'm new at the job.

I keep getting letters lamenting the fact we don't publish more software for the X machine. You fill in the manufacturer's name: Apple, Atari, PET, TRS-80, or whatever. And the mailbox is full of press releases telling of this or that new computer magazine specializing in this or that machine. Everyone is getting into the act. One wonders if there will continue to be a market for generalist magazines like *Recreational Computing*. I certainly hope so.

This issue has a general purpose fill-in-the-blanks form for those of you who want to participate in the magazine. Let us hear from you; what *Recreational Computing* becomes is as much your doing as mine.

5th Faire

Nearly everyone was there or so it seemed. Slightly under 20,000 people of all varieties. I ran about in my new capacity as editor/photojournalist trying to put together a photo-feature for *Recreational Computing*. I did manage to botch the camera settings and all my picture efforts (Jim Warren's nose being tweaked by the Microbot was but one) came to naught.

My major impression of the Faire is one of chaos. Too much hype, too much salesmanship, too many things. I was confused and lost—and I'm supposed to be an expert. I can't help but wonder what a novice would feel.

Double Digit Inflation

Inflation is everywhere. You see every price continuing to increase. Unfortunately *Recreational Computing* isn't immune from the pressures of general economics. Beginning with our next issue the cover price will increase to $2.50. Subscription rates will increase shortly thereafter. Maybe now's the time to consider renewals. You might also consider becoming a retaining or sustaining subscriber and get your name up in lights.

Sustaining Subscribers

Bill Godbout Electronics
Byte Publications
Paul, Lori and Tom Calhoun
Louis R. Patzke

Staff

Editor
Dennis Allison
Contributing Editors
Bob Albrecht
Tracy Deliman
Don Inman
Ramon Zamora
Copy Editor
Sara Le Forge
Publisher
Ann Merchberger
Production Manager
Carole Cullenbine
Artist
Aleeca Harrison
Typesetter
Phyllis Adams
Proofreader
Nancy Heubach
Marketing Director
Gavin W. Cullen
Advertising & Circulation Manager
Michael Madaj
Circulation Assistants
Peter Clark
Leah Dansby

Retaining Subscribers

Algorithmics, Inc.
Gerald Bowman
Robert Conners
David R. Dick
Mark Elgin
Joi Ellis
John B. Fried
Scott B. Guthery
Alan Hamilton
Brian Herring
William G. Hutchison, Jr.
W. A. Kelley
William M. Richman II
Phillip A. Smith
Joseph A. Weisbecker
Brett Wilson

ADVERTISING

ADVERTISING space is available in this publication. Please direct inquiries to the Advertising Manager, People's Computer Company, Box E, Menlo Park, CA 94025. (415) 323-3111.

SUBSCRIPTION INFORMATION

U. S. RATES
☐ $10/1 yr. (6 issues)
☐ Retaining subscription @ $25 ($15 tax deductible)
☐ Sustaining subscription @ $100+ ($90 tax deductible)
☐ Lifetime subscription @ $900+ ($700 tax deductible)

Please allow 6-9 weeks for your first issue to arrive.

FOREIGN RATES
Payments must be in $US drawn on a US bank.
☐ $17 Canada/Mexico First Class
☐ $23 Rest of World Airmail
☐ $14 World Surface Mail

Delivery of foreign mail is slow and unreliable. We strongly advise airmail.

BACK ISSUES

$2.50 each:	$3.00 each:
Vol. 6, Nos. 1, 2, 3, 4, 5	Vol. 7, Nos. 3, 4, 5, 6
Vol. 7, Nos. 1, 2	Vol. 8, Nos. 1, 2, 3, 4, 5

Outside the U.S. add $.50 per issue.

Recreational COMPUTING
May/June 1980
Volume 8, Issue 6

Cover by Aleeca Harrison

Recreational Computing (ISSN #0164-5846) 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, non-profit corporation, and donations are tax-deductible. Second class postage paid at Menlo Park, California, and additional entry points. Copyright 1980 by People's Computer Company, Menlo Park, California.

FOREIGN DISTRIBUTORS OF *RECREATIONAL COMPUTING*: **UK & EUROPE:** L P Enterprises, 8/11 Cambridge House, Cambridge Road, Barking, Essex, 1G11 8NT, Great Britain. Hofacker-Verlag, Tegernseer Strasse 18, D-8150 Holzkirchen, West Germany. Computerland/Computer Store AB, Box 7134, Kungsgatan 19, S-10387 Stockholm, Sweden. **CANADA:** RS-232, 186 Queen Street W., Toronto, Ontario M5V 1Z1, Canada. **ASIA & AUSTRALIA:** Electronic Concepts Pty Ltd., 55 Clarence Street, Sydney, NSW 2000, Australia. Computer Store, POB 31-261, 22B Milford Road, Milford, Auckland 9, New Zealand. ASCII Publishing, 305 Hi Torio, 5-6-7 Minami Aoyama, Minato-ku, Tokyo 107, Japan. Systems Formulate Corporation, Shinmaki-cho Building, 1-8-17 Yaesu, Chuo-ku, Tokyo 103, Japan. **AFRICA:** Computers, Etc., 170 Hendrik Verwoerd Drive, POB 51978, Randburg 2125, Republic of South Africa.

An Introduction To COMPUTER MUSIC HISTORY AND FUNDAMENTALS

C. Roads
Experimental Music Studio, Rm 26-311
Massachusetts Institute of Technology
Cambridge, MA 02139

Introduction

This article will introduce the interested reader to the fundamental concepts and techniques of computer music as it is practiced today. First, the history of automated music (older than you might at first imagine) will be surveyed. Next, the basics of sound synthesis *via* computer will be examined, and the technical requirements for high-fidelity sound will be outlined. Then sound synthesizers, both the older analog and the newer purely digital breeds, will be discussed. Composing programs which embody systems of musical intelligence will be introduced, along with some other applications of computers to music, such as music analysis and printing. Many references are provided for further investigation.

Although some technical detail is necessary in discussing specific methods of sound synthesis, this article is self-contained; any intelligent person should have little trouble understanding the fundamentals of computer music.

The Early History of Automated Music

Automated music goes back a lot further than most people think. But then it seems that music and symbolic thinking have long been associated. For example, the Greek Pythagoras (d. 497 B.C.) was both a number theoretician and a music theoretician. The germ idea of automatic music, *i.e.*, a music played without a musician, must be traced to the ancient Chinese wind-chimes [1] which, powered by gusts of wind, not only generated sound but also did not require compositional preprogramming. Melodies and harmonies were generated both by the hung structure of the chimes, and also by the wind patterns of the day.

By the Fourteenth Century, the Iranian astronomer and mathematician Jamshid ben Mas'ud ben Mahmud Ghiath ed-Din al-Kashi had invented a planetary computer for computing the longitudes of the Sun, Moon, and visible planets [2]. This was a very early and inventive development, and we should admire this scientist. However, in the 1200's some Dutch musical engineers, had devised some carillons (a carillon is a chime of bells) which could play preprogrammable melodies and harmonies automatically. These programmable carillons were driven by a revolving cylinder of wood or iron, studded with pegs which triggered a bell-ringing mechanism. The revolving cylinder was covered with holes (analogous to the binary "zero"), and by placing wooden pegs (analogous to the binary "one") in various holes, different melodies and chords could be programmed (digitally!). Some Flemish carillons had *ten thousand* locations (holes) for the insertion of note pegs [4].

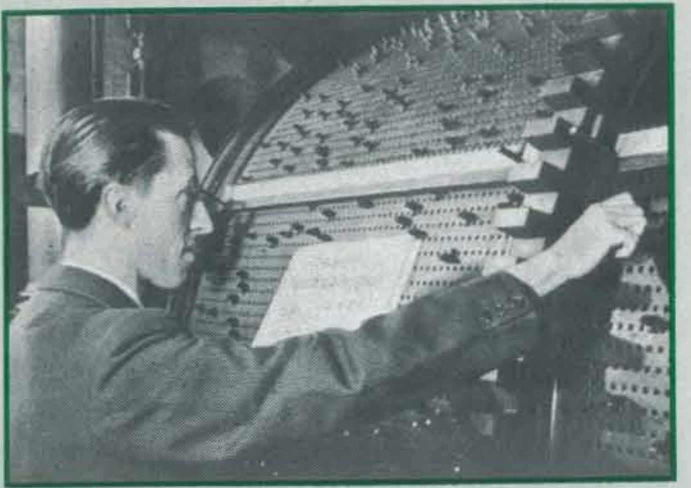
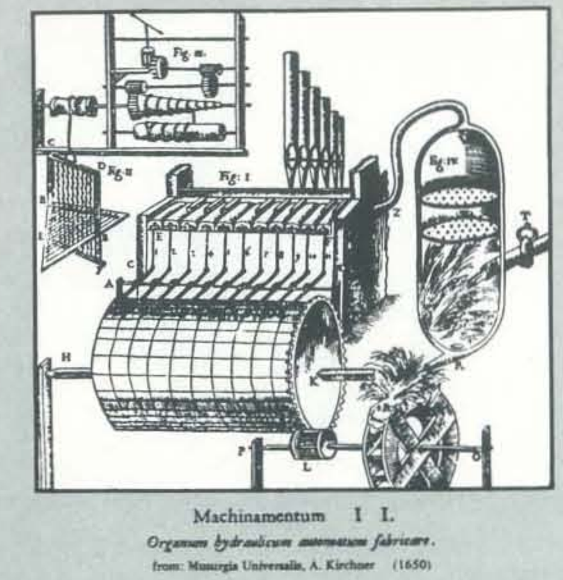


Figure 1. Eugene Uten, the carilloneur, reprogramming an early Belgian carillon. (Photograph from [3])

The Renaissance

As in most other technical matters, the Dark Ages in Europe produced nothing of note, and it is only with the Renaissance that new interest in automatic music began to burgeon. In the 1500's Leonardo Da Vinci constructed a mechanical spinet piano and drums [3, p. 16]. A. Kirchner's *Musirgia Universalis* (1650) details some fantastic plans for a steam-driven organ. [Figure 2]


Machinamentum I I.
Organum hydraulicum automatum fabricare.
from: Musurgia Universalis, A. Kirchner (1650)
Figure 2. A design for a steam-powered organ, taken from *Musurgia Universalis* (1650) by A. Kirchner.

Baroque Androides

It was in the Baroque period of the 18th century that some of the most extravagant automatic music devices were contrived. The French during this period exhibited a curious fascination with robots which played musical instruments, which Buchner [3] calls "androides." The most celebrated maker of musical androides was the mathematician and mechanic Jacques de Vaucanson (1709-1782), who exhibited a flute-playing androide in Paris in 1738. Audiences refused to believe that the robot was actually playing the flute; they thought that there was a mechanical instrument hidden in the body of the player. This flute-playing robot was reportedly a sensation for decades, being shown in many exhibitions. Another famous builder of musical robots was the father-son pair Pierre and Henry-Louis Jacquet-Droz. Their androide, "The Designer," drew a portrait of Marie-Antoinette, while their androide, "The Writer," wrote a letter of some fifty words. The Jacquet-Droz androide called "The Clavecin Player" played a keyboard in an animated fashion with breathing movements, head and eye movements, and bending towards her music as if to read it more clearly. [Figure 3] Pierre Jacquet-Droz achieved quite some fame for his inventions, and in 1758 he was invited by the King of Spain to Madrid. However, his work was not entirely sympathetically received, for he nearly was burned at the stake for practicing sorcery when the Inquisition came to power.

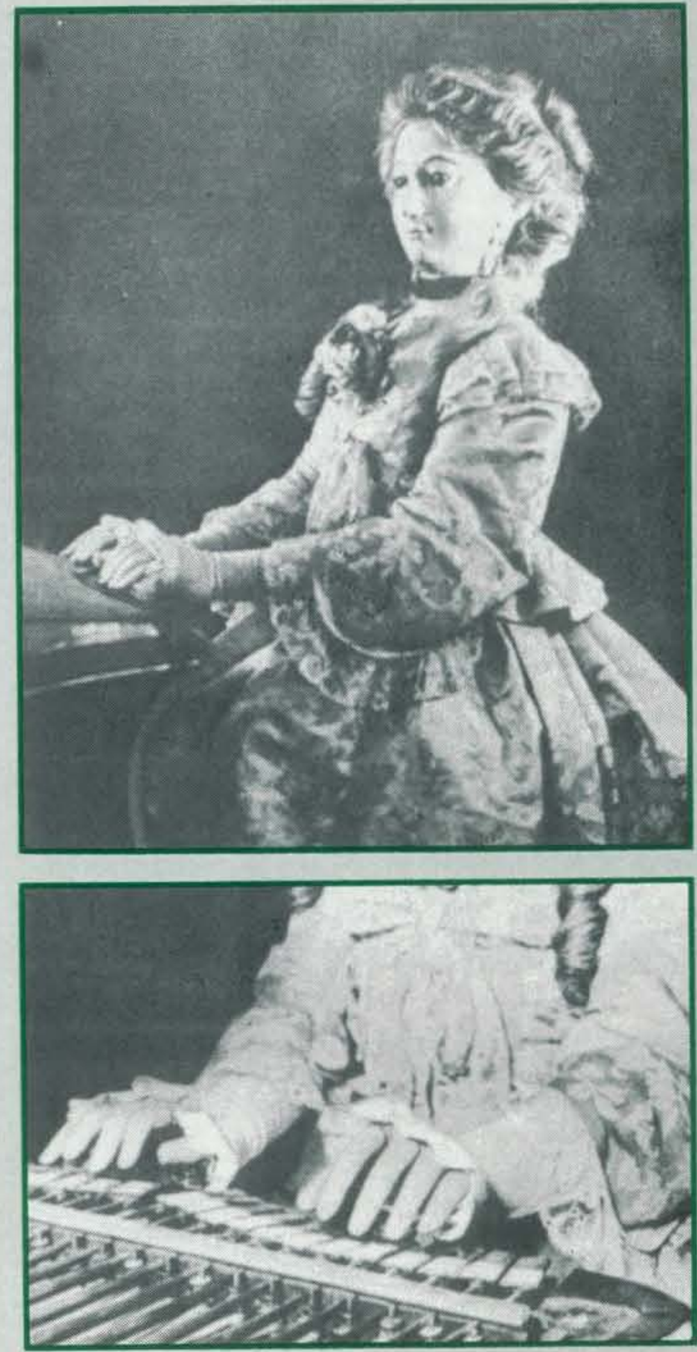


Figure 3. "The Clavecin Player" by Jacquet-Droz, a musical robot built in the 18 th century. (Photograph from [3].)

Another early mechanical music pioneer was F. Engramelle (1727-1781) who was one of the first to concern himself with *automatic music printing*. "His idea was to construct an apparatus able to write down automatically whatever was played on a piano" [4, p. 20]. The Londoner Rev. Creed presented a paper in 1747 to the Royal Society which demonstrated the possibility of automatic music transcription from a keyboard. By 1774, such a device had been constructed by the Berlin mechanician Hohlfeld. Tones played on the keyboard were recorded in a telegraphic notation which could be easily transcribed to common music notation.

Even earlier, the *first electric instrument* was built by the Jesuit Father La Borde in 1759. The device was an electric carillon (electric bell-chimes); by pressing a key on a keyboard, an electric circuit was activated which rang the appropriate bell.

Automatic Music Boxes

In the 19th century, automatic music boxes enjoyed great favor in both Europe and the United States. Indeed, only with the introduction of the phonograph in the early part of the 20th century was the heyday of the music box ended [5].

The earliest music boxes played metal cylinders – the same as most of the tiny boxes we see in gift and souvenir stores. In the 19th century however, music boxes were developed around *disk* technology taken to grand proportions. The German Paul Lochman introduced in 1885 a music box which played circular disks called a *Symphonium*. Another Leipzig outfit called Polyphon Music Works made a similar device soon thereafter. After some vicious lawsuits between the two over patent rights (Lochman lost) disk technology spread throughout Europe and most particularly to the United States. [Figure 4] The Regina Musical Box Company of New Jersey made a large 27-inch disk machine; "undoubtably the best disk machine made either before or since; it had two combs, comprising all 172 notes tuned in chromatic scale, and embracing over seven octaves. Time of playing this disk were over two minutes. *Nothing in automatic music had ever been heard before like the 27-inch Regina*, and they soon became popular the world over . . . The machine was displayed at the Crystal Palace in London" [5, p. 32].

Introduction of the Phonograph, End of the Mechanical Age

The next major innovation in music boxes occurred in 1896 when G. Brachhausen of the Regina Company patented a self-changing disk machine. The machine held a magazine of twelve disks. By inserting a penny, the selected disk could be played. By making the music disks randomly-accessible, the Regina Company hoped to stave off the growing popularity of the recently-introduced *phonograph* put out by Edison's company in the late 1890's The heavy-metal disks and instrumental gadgetry contained within the music boxes were, however, no match for Edison's system, which could record any performance, particularly vocal/instrumental combinations.

The last versions of the automated boxes were also the grandest in scale. These huge contraptions, called "Orchestrions" were the embodiment of the concept of the automated orchestra. [Figure 5]

By 1900, the age of mechanical music instruments was over. Even so, the composers Haydn, Mozart, Beethoven, and Debussy, among others, had all written pieces for mechanical instruments. The last sign of organized interest in mechanical automated music was a "Mechanical Music Festival" held in 1926 in Baden, Germany, which featured works by Hindemith, Toch, and Münch. One 20th century composer has continued to preoccupy himself with the possibilities of mechanical performance, specifically player-piano mechanisms. Conlon Nancarrow's piano music would be impossible to perform without mechanical means; it stands as an example of the extension of the domain of possible compositions through automated processes [7]. What mechanical instruments are left are today controlled by microprocessors [Figure 6].

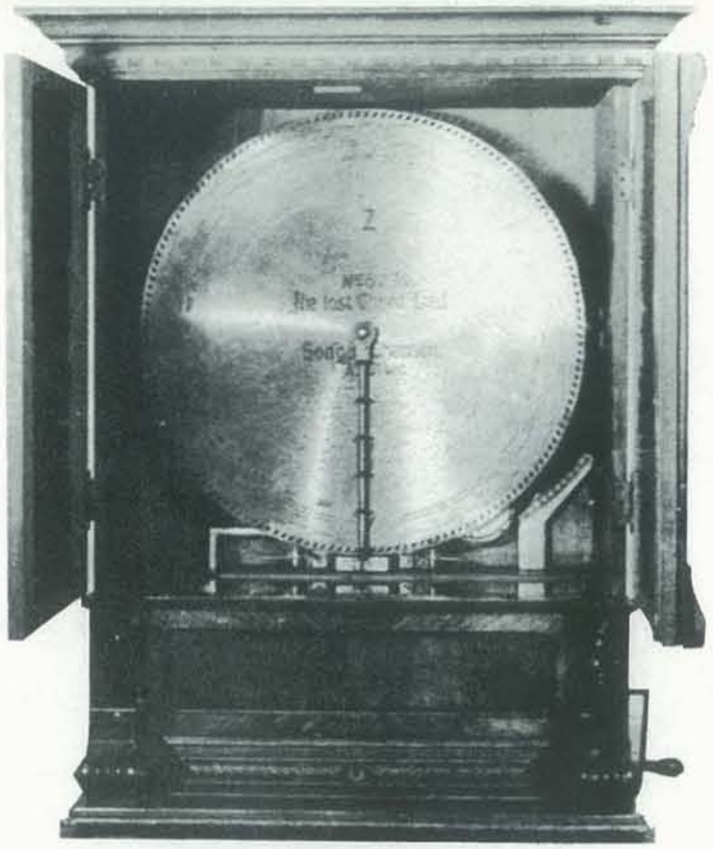


Figure 4. A large disk musical box – the Fortuna "Marvel" made in Leipzig. The machine plays 26-inch music disks, and holds within its cabinet a reed organ, drum, and triangle. The mechanism is activated by the insertion of a coin.

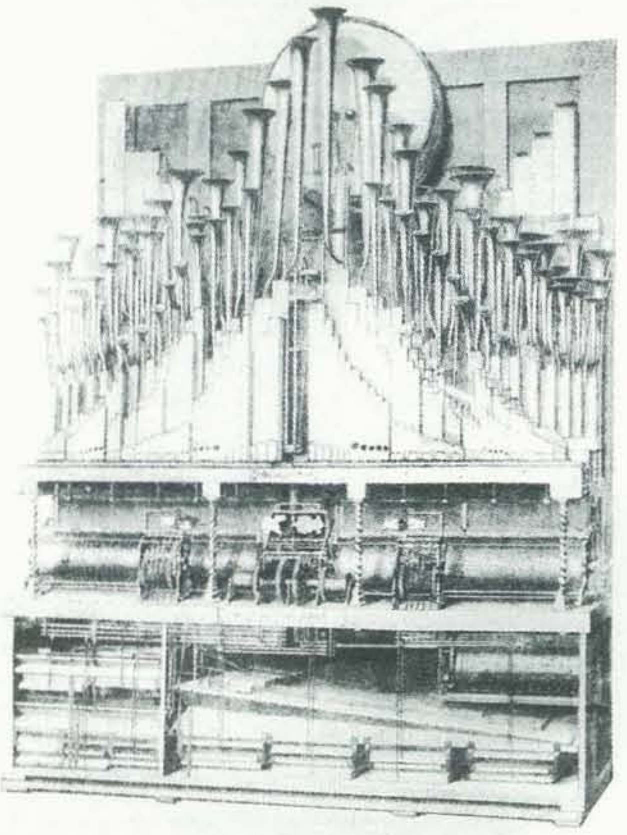


Figure 5. An "Orchestion" – automated orchestra – built by M. Wefte of Germany in 1862. Notice the vast range of pipes and bellows, and the bass drum at top center.

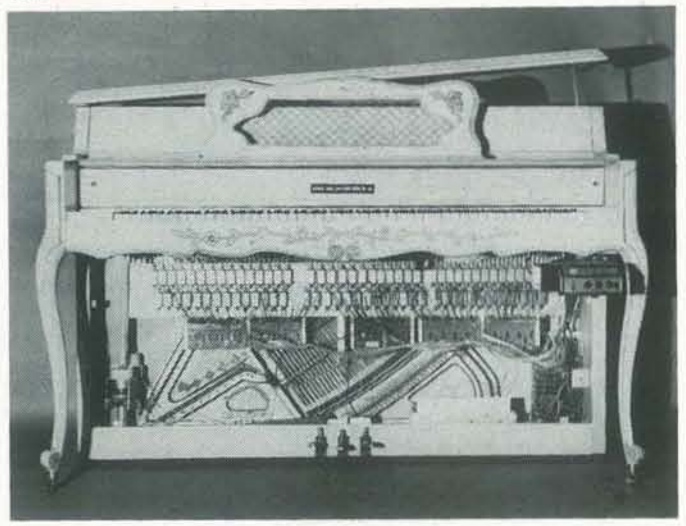


Figure 6. The Marantz reproducing piano, called the *Pianocorder*, equipped with a digital processor, control and driver circuitry, and a digital cassette machine. An input/output terminal may also be attached to the piano for progammable control. The device is capable of "listening" to one's piano performance and then reproducing it. Transformations on this recorded performance are then possible.

The Rise of Electronic Music

In 1865, Alexander Graham Bell hit upon the idea of transmitting speech by electric waves; he actually achieved his goal by transmitting intelligible speech in 1876. This could be seen as the beginning of serious efforts to link the audio domain to electric technology. The first major storage medium for audio was developed by V. Poulsen of Copenhagen, who introduced a steel-wire recorder in 1902.

By the turn of the century, pressures from within music itself were leading to an expanded view of possible musical sounds. Concurrently, technical means were improving, making such notions as an expanded musical sound universe feasible. By far the grandest conception from around this period (1903) was the gigantic *Teleharmonium* invented by Dr. Thaddeus Cahill. Cahill's plan was to broadcast electronic music on a subscription basis over the telephone wires being installed so rapidly around the United States. However, inductive interference created by the Teleharmonium's sound signals led to complaints from regular telephone customers. Despite an enormous investment, the project had to be cancelled. By this time, Cahill had assembled "thirty carloads" of tone generators, whose tones could be combined to produce music [6].

In 1913, the Italian painter Luigi Russolo issued his famous manifesto in which he called for a new sonic art form based on noises. He implemented his ideas by building a collection of noise-making instruments.

Meanwhile, the development of the vacuum tube in 1908 by Dr. Lee de Forest led to a spate of electronic music instruments, including the *Theremin* (1919), the *Trautonium* (1930), the *Ondes Martinot* (1928), and others too numerous to mention here. Some visionary composers were anxious to begin experimenting with these resources; however, most of these new instruments were in scarce supply. The composer Edgard Varèse traversed the country vainly trying to interest engineers and the companies they worked for in the possibilities for new, electronic musical instruments [8]. Composer Henry Cowell conceived of an electronic device for automatically performing complicated cross-rhythms around 1916; he finally paid for its development in the 1930's [9].

Although the tape recorder was invented in 1935 by AEG, under the German name *magnetophone*, it was not until the 1940's that its tremendous musical potential began to be tapped. In 1944-49, the Frenchman Boisselet composed a symphony for a large orchestra with piano, organ, Ondes Martinot (electronic instrument), harp, oscillators, and "magnetophone" [10, p. 62].

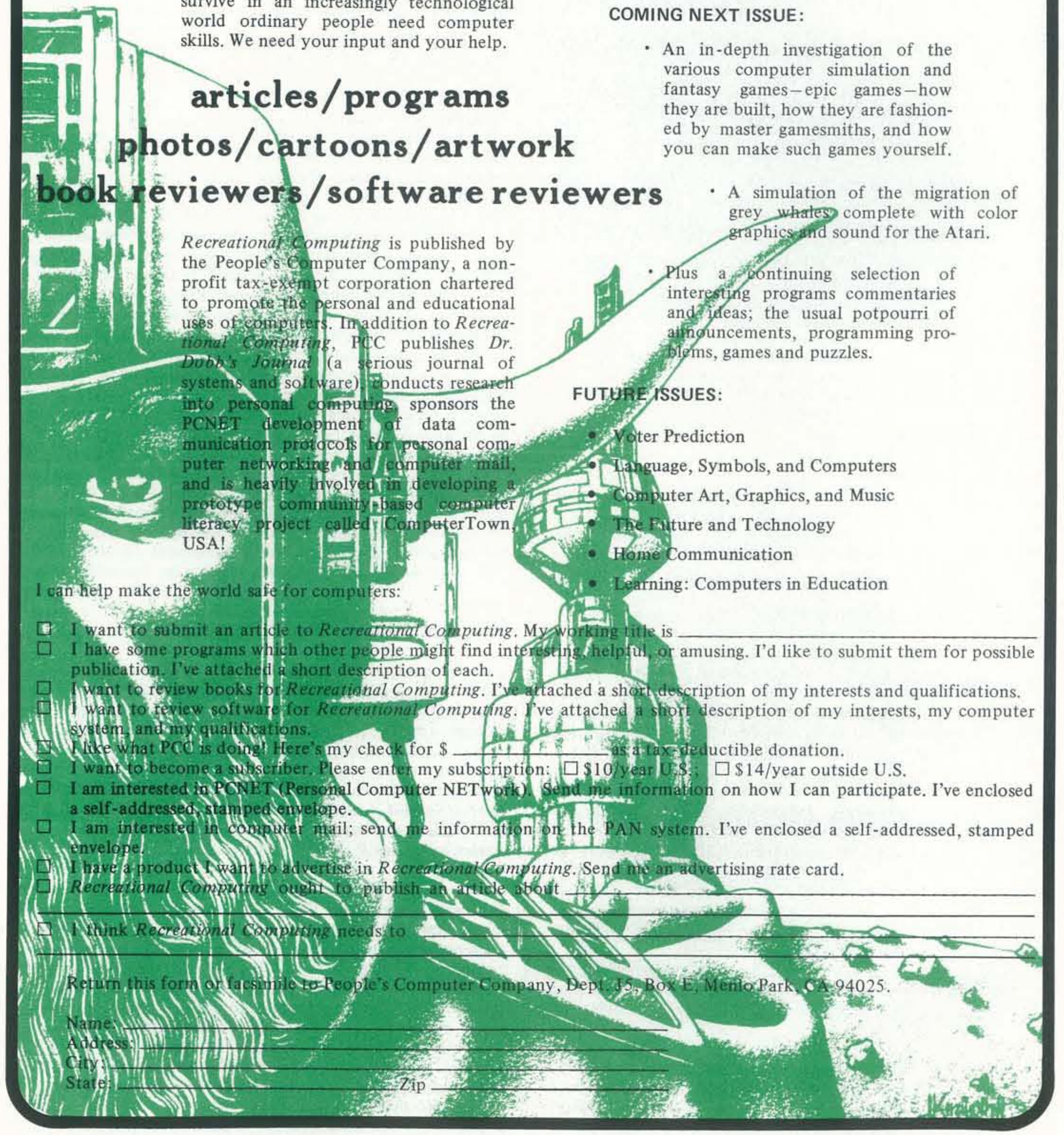
In 1948, the French engineer and composer Pierre Schaeffer presented a "concert of noises" over French radio. Interestingly, Schaeffer's pioneering experiments with recorded, natural sounds (which he called "musique concrête," a term which is still extensively used) were based on phonograph record technology. Another significant event in the history of electronic music was a visit in 1949 to Bell Laboratories by the German physicist and acoustician Werner Meyer-Eppler. There Meyer-Eppler saw demonstrated the Bell Labs *Vocoder*, which had been developed in the late 1930's. This device, which so impressed Meyer-Eppler, was capable of extracting the *formant* information from one signal (say, a human voice) and using this information to process another sound, such as a flute playing a melody. The result was the sound of a "talking flute." Meyer-Eppler used the Bell Labs Vocoder in lectures in Germany; a representative from the German radio attended one of these lectures and was intrigued by the possibilities of setting up a studio for experimentation with *elektronische musik*. The first broadcast of electronic music experiments occurred in 1951. In America, the composer Vladimir Ussachevsky gave the first public demonstration of *tape music* (using the newly-developed Ampex tape recorder) on May 9, 1952. Louis and Bebe Barron began to use electronic sound generating equipment to produce film scores for such science-fiction classics as *Forbidden Planet* (1954) [11].

Automatic Electronic Music

Shortly after the development of the first electronic computer, the *Enaic* in 1945, organ manufacturers in the United States began introducing options for rhythmic accompaniment, based around sequencer (relay) type technology. These very primitive "automated rhythm sections" produced incessantly repeating percussive sounds. The closest things to a "programmable" analog music synthesizer were the *Mark I* (1955) and *Mark II* (1964) synthesizers built by Olson and Belar of RCA. Although they were internally entirely analog, they were driven by musical specifications coded on a wide, punched paper-tape.

The introduction into the marketplace of *voltage-controlled synthesizers* in the mid-1960's brought new possibilities for electronic music expression to a much wider group of musicians. [Figure 7] By varying the control-voltages applied to the various sound-producing modules, it was possible to change the character of the sound being produced. Nearly every manufacturer of analog music synthesizers offered a more-or-less crude *analog sequencer* module as an option. Using an analog sequencer, a *sequence* of voltages for controlling some musical parameter (such as frequency) could be preset. The sequencer could then cycle through these voltages to produce a repeating melody. [Figure 8]

Continued on pg. 38



HELP CREATE THE FUTURE!

Help make *Recreational Computing* an exciting magazine about the world and computing. We are for everyone not just for kids or just for computer wizards. Computers are part of the future. To survive in an increasingly technological world ordinary people need computer skills. We need your input and your help.

articles/programs
photos/cartoons/artwork
book reviewers/software reviewers

Recreational Computing is published by the People's Computer Company, a non-profit tax-exempt corporation chartered to promote the personal and educational uses of computers. In addition to *Recreational Computing*, PCC publishes *Dr. Dobb's Journal* (a serious journal of systems and software), conducts research into personal computing, sponsors the PCNET development of data communication protocols for personal computer networking and computer mail, and is heavily involved in developing a prototype community-based computer literacy project called ComputerTown, USA!

COMING NEXT ISSUE:

- An in-depth investigation of the various computer simulation and fantasy games—epic games—how they are built, how they are fashioned by master gamesmiths, and how you can make such games yourself.
- A simulation of the migration of grey whales complete with color graphics and sound for the Atari.
- Plus a continuing selection of interesting programs commentaries and ideas; the usual potpourri of announcements, programming problems, games and puzzles.

FUTURE ISSUES:

- Voter Prediction
- Language, Symbols, and Computers
- Computer Art, Graphics, and Music
- The Future and Technology
- Home Communication
- Learning: Computers in Education

I can help make the world safe for computers:

☐ I want to submit an article to *Recreational Computing*. My working title is ______________________
☐ I have some programs which other people might find interesting, helpful, or amusing. I'd like to submit them for possible publication. I've attached a short description of each.
☐ I want to review books for *Recreational Computing*. I've attached a short description of my interests and qualifications.
☐ I want to review software for *Recreational Computing*. I've attached a short description of my interests, my computer system, and my qualifications.
☐ I like what PCC is doing! Here's my check for $ ____________ as a tax-deductible donation.
☐ I want to become a subscriber. Please enter my subscription: ☐ $10/year U.S.; ☐ $14/year outside U.S.
☐ **I am interested in PCNET (Personal Computer NETwork). Send me information on how I can participate. I've enclosed a self-addressed, stamped envelope.**
☐ I am interested in computer mail; send me information on the PAN system. I've enclosed a self-addressed, stamped envelope.
☐ I have a product I want to advertise in *Recreational Computing*. Send me an advertising rate card.
☐ *Recreational Computing* ought to publish an article about ______________________
____
☐ I think *Recreational Computing* needs to ______________________
____

Return this form or facsimile to People's Computer Company, Dept. J5, Box E, Menlo Park, CA 94025.

Name: ______________________
Address: ______________________
City: ______________________
State: ____________ Zip ____________

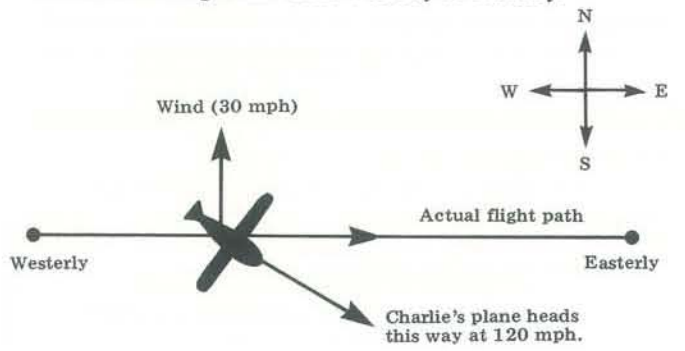
PROGRAMMING PROBLEMS

BY BOB ALBRECHT, DON ALBERS AND JIM CONLAN

PROBLEM #15 ONE WAY TO FLY

The town of Westerly is 100 miles due west of the town of Easterly. Charlie L., an experienced pilot, flies a plane from Westerly to Easterly.

* Charlie's plane cruises at 120 mph.
* The wind is from the south at 30 mph.
* Since Charlie is an experienced pilot, he heads south of east by just enough to compensate for the wind. His plane travels in a straight line from Westerly to Easterly.



Questions.

Q1. What is the heading of the plane? The heading is the direction the plane is pointed, measured in degrees clockwise from north.
Q2. What is the actual west to east speed of the plane, along the flight path?
Q3. How long does it take Charlie to fly from Westerly to Easterly?

Write a program to compute the heading, the west to east speed and the time of flight given the speed of the plane and the speed of the wind. A RUN of your program might look like this.

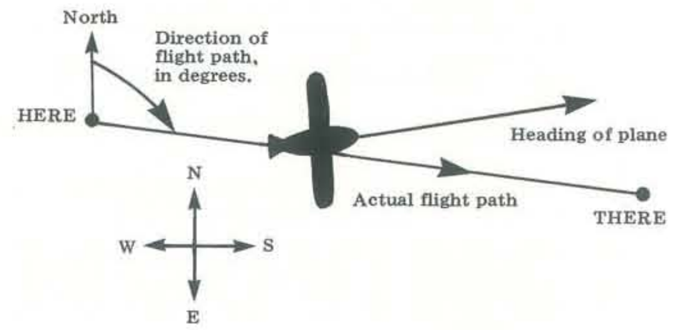
```
RUN
SPEED OF PLANE? 120
SPEED OF WIND? 30

HEADING IS 104.478 DEGREES
FLIGHT PATH SPEED IS 116.19 MPH
TIME OF FLIGHT IS .860663 HOURS
```

Remember, Charlie is flying from Westerly to Easterly, a distance of 100 miles. Strange, but true, the wind is always from the south. The independent variables are the speed of the plane and the speed of the wind.

PROBLEM #16 FLYING FROM HERE TO THERE

We assume that you have already read, and perhaps solved, PROBLEM #15 ONE WAY TO FLY.

Well, Charlie moves around a bit. Unfortunately, our program for PROBLEM #15 is useful only for west to east flights of 100 miles when the wind is from the south. So, let's make a more useful program . . .flying from HERE to THERE with the wind from any direction.



Charlie, as you may recall, is an experienced pilot. He points his plane just enough into the wind so that it travels in a straight line from HERE to THERE.

You are given –

* Direction of desired actual flight path from HERE to THERE, measured in degrees clockwise from north.
* Distance from HERE to THERE.
* Speed of Charlie's plane *in the direction in which it is pointed.*
* Direction the wind is *from*, measured in degrees clockwise from north.
* Speed of the wind.

Write a program to compute the heading of Charlie's plane (in degrees clockwise from north), the speed along the actual flight path and the time of flight from HERE to THERE. For the flight from Westerly to Easterly (PROBLEM #15), a RUN of your program might look like this.

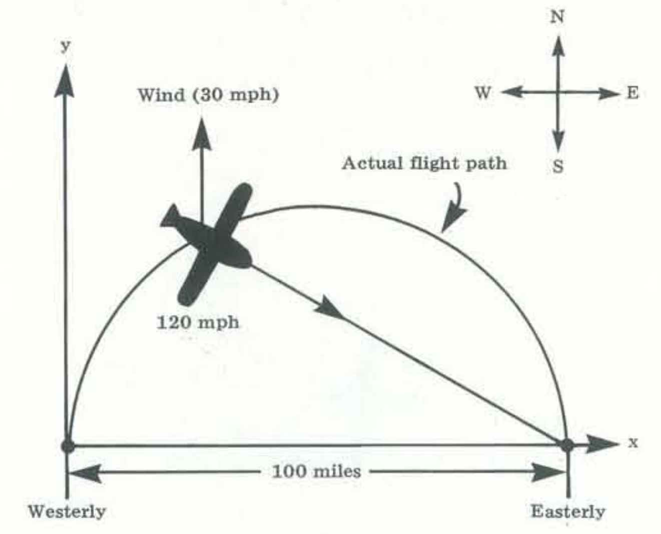
```
RUN
DISTANCE FROM HERE TO THERE? 100
DIRECTION FROM HERE TO THERE? 90
SPEED OF PLANE? 120
DIRECTION WIND IS FROM? 180
SPEED OF WIND? 30
```

```
HEADING IS 104.478 DEGREES
FLIGHT PATH SPEED IS 116.19 MPH
TIME OF FLIGHT IS .860663 HOURS
```

PROBLEM #17 ANOTHER WAY TO FLY

Before reading this problem, first read PROBLEM #15 ONE WAY TO FLY, where you will learn about the towns of Westerly and Easterly, where the wind is always from the south.

Archie B., who soloed recently, flies his plane from Westerly to Easterly. Archie is a very straightforward and direct sort of person. He simply points his plane directly at Easterly, ignoring drift due to the wind from the south. His plane, of course, drifts to the north. Does this bother Archie? No, he simply keeps changing headings so that his plane is always pointed directly at Easterly. Therefore, he flies a curve as indicated in the following diagram.



Write a program to help you answer the following questions.

Q1. How long does it take Archie to fly from Westerly to Easterly?
Q2. In the diagram, note the xy coordinate system. Complete the following table showing the x and y coordinates of Archie's plane as functions of time.

t (minutes)	x (miles)	y (miles)
10		
20		
30		
etc.		

Keep 'em flying!

PROBLEM #18 MORE THAN PERFECT?

One is one and all alone and . . . Yes, one (1) is unique in many ways. It is the only positive integer with exactly one divisor which (of course) is one. It is also the only positive integer for which the sum of its divisors is equal to the number itself.

* The sum of the divisors of 1 is 1.

For all positive integers greater than one, the sum of the divisors of the number is greater than the number.

NUMBER	DIVISORS	SUM OF DIVISORS
2	1,2	3
3	1,3	4
4	1, 2, 4	7
5	1, 5	6
6	1, 2, 3, 6	12

The sum of the divisors of six is twelve, which is exactly two times six. Six is called a *perfect number.*

* A *perfect number* is a positive integer for which the sum of the divisors of the number is exactly twice the number.

Six is the first perfect number. The second perfect number is 28.

DIVISORS OF 28: 1, 2, 4, 7, 14, 28
SUM OF DIVISORS: 1 + 2 + 4 + 7 + 14 + 28 = 56

But, of course, some people said, "What is so *perfect* about that?" Others pointed out that, if you added only the *proper* divisors, you got the number itself.

PROPER DIVISORS OF 6: 1, 2, 3
SUM OF *PROPER* DIVISORS: 1 + 2 + 3 = 6

And again:

PROPER DIVISORS OF 28: 1, 2, 4, 7, 14
SUM OF *PROPER* DIVISORS: 1 + 2 + 4 + 7 + 14 = 28

Amazing, delightful and . . . perfect? Well, anyhow, interesting. The first five perfect numbers are:

6, 28, 496, 8128 and 33550336.

That is *not* our problem.

Our problem is to find positive integers for which the sum of the divisors is exactly equal to *three* times the number.

The first such number is 120.

DIVISORS OF 120: 1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 20, 24, 30, 40, 60, 120

SUM OF DIVISORS: 360 (360 = 3*120)

The second such number is 672. *You* compute the divisors and the sum of the divisors.

Is there a third such number? If so, what is it? Is there a fourth number? If so, what is it? and so on . . .does it ever stop?

Is there a number (positive integer) for which the sum of divisors is equal to *four* times the number? More than one such number? How about five times the number . . . oh well, this could go on forever!

Please send answers to any of the above questions. Tell us how you got the answer. We would like to see programs that you used to get these answers, especially if they run on home/school/personal computers.

New from Osborne/McGraw-Hill

6502 Assembly Language Programming
by Lance Leventhal

"the book that will probably get the reputation as being the 6502 bible"
Interface Age
January 1980

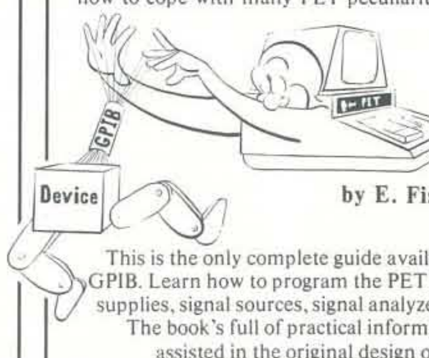
Learn to increase the capabilities and performance of your 6502-based computer by programming in Assembly Language.

#27-6 $12.50

TO ORDER, please specify book title, order #, quantity, and your return mailing address. Enclose check or money order for price of each book ordered, plus 75¢ per book for shipping. California residents please include sales tax. Allow 4 weeks for delivery. For faster shipment or credit card order, call (415) 548-2805.

PET® Personal Computer Guide
by C. Donahue & J. Enger
New this Spring

A step-by-step guide for the computer novice who wants to learn to program and operate the PET computer. It covers PET programming techniques, graphics, operation, and how to cope with many PET peculiarities. #30-6 $15.00



PET® and the IEEE 488 Bus (GPIB)
by E. Fisher and C. W. Jensen

This is the only complete guide available on interfacing PET to GPIB. Learn how to program the PET interface to control power supplies, signal sources, signal analyzers and other instruments. The book's full of practical information, as one of its authors assisted in the original design of the PET-GPIB interface.

®PET is a registered trademark of Commodore Business Machines #31-4 $15.00

Mail to: OSBORNE/McGraw-Hill
630 Bancroft Way, Dept. RC
Berkeley, CA 94710

OSBORNE/McGraw-Hill

Programming The 6502 In Machine Language

BY ERNEST GOLLUX

The processing heart of most personal computers on the market today (the Apple, the Atari, the PET, the Kim, and more) is the 6502. Its low cost and simple design have made it the workhorse microprocessor of the personal computer industry. Most personal computer users don't have much contact with the 6502—they program their computer primarily in BASIC. But if you really want to understand computers, eventually you have to contend with the raw machine.

Programming in machine language is quite different from programming in a high level language. The programmer has to keep track of many more details. He must worry about the representation of the information and how to access it. In return, he gets some additional flexibility and additional performance.

The 6502 is an eight-bit processor, which means that it processes information eight bits at a time. All data, insofar as the processor is concerned, are a string of eight bits. Some instructions interpret those bits as a binary number with values ranging from 0 to 255 decimal (unsigned) or −128 to 127 (signed). The programmer can interpret the number in any way he wants. For example, the number 65 could represent the ASCII character "A" in one context, and it could be $2^6 + 1$, a value in some computation, in another. The 6502 manipulates data by moving them from memory into registers internal to the processor, performing the desired operation or operations and then moving them back into memory. The register complement of the 6502 is shown in Figure 1. The accumulator is used to perform all arithmetic and logical operations on data. The program counter contains the address of the next instruction in memory. It is incremented as each instruction is processed so instructions are executed from low addresses to high sequentially. The flow of execution can be changed by branch instructions, jump instructions, or jump subroutine instructions. In the latter case, execution can resume at the instruction following the jump subroutine since the program counter is saved on a last-in-first-out list or stack. This stack uses the stack pointer to keep track of where the most recent return address is stored. The two registers labeled X and Y are index registers and are used to access data from memory. The flags register contains the status of the most recently executed operation and is used to control conditional branching.

Both the program and data are stored in the computer's memory. For the 6502 the maximum memory size is 16 bits or 65536 bytes. That's the largest value which can be represented in 16 bits or two 8-bit bytes. Since the 6502's world is really 8 bits wide, memory is divided into 256 pages of 256 bytes each. A single byte is adequate to specify a location within a page. Addresses are 16 bit numbers. They could be stored in either of two ways in the computer memory: most significant byte then least significant byte or the reverse. The assembler (Figure 2) provides ways to store data in both forms, but the convention for addresses is for the least significant byte (LSB) to be stored first (lower address) and then the most significant byte (MSB). With that ordering the processor does not need to save anything temporarily when computing an address.

The processor has certain built-in expectations as to how memory is to be used. Page zero is a good place to store frequently used data since the processor has a space efficient way to access them; in addition, by using a pointer in page zero, data elsewhere in memory can be accessed efficiently. Page one is preempted for use as the stack. Some locations in high memory (addresses $FFFA through $FFFF) are used to indicate what the processor is to do when power is applied or an interrupt occurs. An interrupt is a signal from the outside world to the processor indicating that it should suspend whatever it is doing and pay attention.

Each personal computer has different expectations about the use of memory. Often some addresses are preempted for the BASIC interpreter. Certainly some page zero locations are used by the system and should not be modified by the programmer. Other memory is reserved to communicate with peripheral devices; still other memory may be mapped into the user's terminal as a display. To know what memory you can use, consult your particular machine's documentation.

If you're going to do any serious amount of programming in the 6502's native language, you should have an assembler. An assembler is a program which converts mnemonic representations of operations and operand references into machine instructions. If you're just playing around, you can do the assembly yourself by hand by looking up the opcode in a machine reference manual and adding the appropriate addresses. It's a good exercise but rapidly becomes tedious.

Figure 2 summarizes the features of most 6502 assemblers. We will use those conventions in the programs we develop below. Figure 3 lists the 6502's instruction set and indicates which condition flags (if any) are affected by the instruction and which addressing modes are provided.

The operation set is rather sparse. You can add and subtract (carry included) 8-bit numbers. You can shift right or left one bit; you can rotate right or left one bit through carry. You can perform bit-wise and's, or's, and exclusive-or's. And that's about all. More complex operations like multiplication must be constructed from sequences of these operations. But that's typical of most eight-bit processors.

Addressing is, on the other hand, unusually rich for a microprocessor. There are, all in all, thirteen different ways an address can be specified. Not all modes exist for all instructions however. The formats of instructions are shown in Figure 4. To be able to program effectively, we must understand how each addressing mode determines its effective address.

Implied Addressing. The address is part of the instruction. These are all one-byte instructions.

Accumulator Addressing. Used to modify the result currently in the accumulator. All instructions are a single byte.

Absolute Addressing. Allows you to specify the address in the instruction. All are three bytes long: an operation byte followed by the low order byte of the address then the high order byte.

Zero Page Addressing. Just like the absolute address mode but only for page zero. Instructions are two bytes long.

Immediate Addressing. Rather than specify the address of a constant and then demand storage space be allocated for the constant too, the 6502 allows the constant to appear in the instruction. These instructions are two bytes long: one operation byte and the immediate data byte.

Relative Addressing. Used for branches only, relative addressing specifies a location −128 bytes backwards or 127 bytes forward from the instruction following the branch instructions. These instructions are two bytes long. Relative addressing has the property that it need not be modified if the program is moved to another location in memory since it depends only upon the relative position with respect to the program counter.

Indexed Page Zero Addressing. This is a two-byte instruction. The second byte of the instruction is added to the contents of either the X index register or the Y index register. The result, truncated to eight bits, is used as the address.

Indexed Absolute Addressing. An instruction using this form is three bytes long. The specified index register (either X or Y) is added to the second byte to form the low order effective address; any carry is then added into the third byte to form the high order portion of the address.

Indexed Indirect Addressing. A two-byte instruction. The second byte and the X register are added together to give an address in page zero. The contents of that location and the one following it (which must be in page zero) is taken to be the desired address.

Indirect Indexed Addressing. Another two-byte instruction. The second byte specifies a location in page zero. The value at that location and Y are added to give the low order byte of the effective address. The high order byte of the address is the word following in page zero with the carry (if any) incorporated.

Absolute Indirect. A three-byte instruction. The second and third bytes store and specify the address of the address.

The sequence of instructions executed is called the flow of control. In any interesting program the flow of control depends upon the data entered. The 6502 captures in its flag register some of the state information produced by each operation on data. For example, if an addition produces a zero result, the Z bit in the flag register is turned on. This bit can then be tested with a BEQ or a BNE instruction to modify the flow of control for the program. Branches in the

Figure 2.
6502 Assembler Conventions

Each instruction appears on a separate line. Each line has the form

LABEL OPCODE OPERANDS COMMENTS

The LABEL and COMMENTS are optional. Some opcodes do not require an OPERANDS field. The LABEL, if it appears, must be the first item on the line. COMMENTS must start with a semicolon (';').

The form of the OPERANDS field determines the addressing mode of the instruction. The various forms are:

address	*absolute or page zero*
address,X	*absolute or page zero indexed by X*
address,Y	*absolute or page zero indexed by Y*
#constant	*immediate*
(address,X)	*indexed indirect*
(address), Y	*indirect indexed*
(address)	*absolute indirect*

The assembler will generate page zero or absolute addressing as needed.

Assemblers also provide directives to control allocation, initialize data areas, and maintain assembly-time variables. Those provided by standard 6502 assemblers are:

.BYTE	$41	*initialize byte data*
.WORD	444	*initialize word data* (LSB, MSB)
.DBYTE	555	*initialize word data* (MSB, LSB)
.TEXT	"hello"	*initialize* ASCII *text*
.END		*end of assembly*
A=B+5		*equate (used for allocation and manifest constants)*
=+4		*equate used for allocation*

Constants may be expressed in a variety of different bases:

$50	hexadecimal (base 16) data
80	decimal number system
@120	octal number system
%1010000	binary number system
'A	ASCII character
*+7	offset to location counter

Most assemblers allow address expressions anywhere addresses are allowed. Some provide for relocation, binding of externals, macros, and other advanced features.

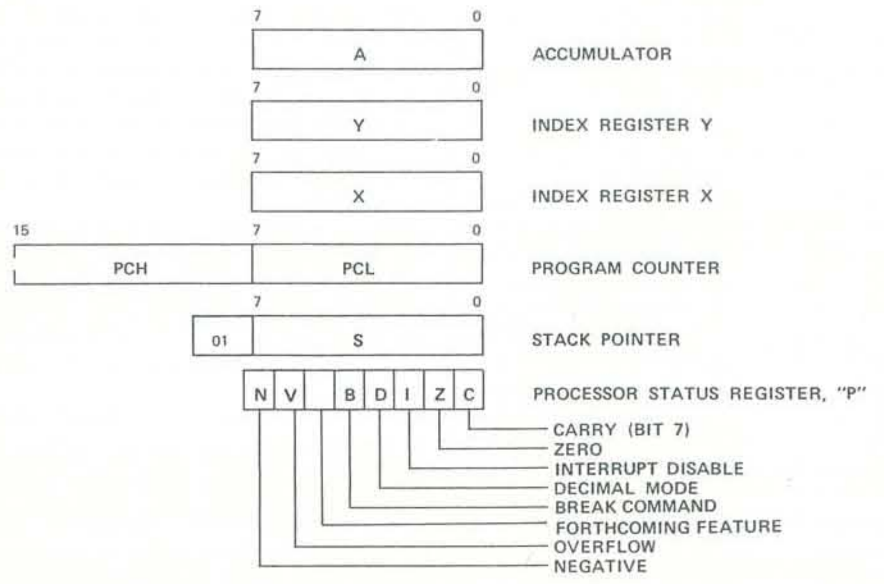


Figure 1.

The register complement of the 6502 includes a 16-bit program counter, an 8-bit accumulator in which arithmetic and logical operations are performed, two 8-bit index registers, an 8-bit stack pointer internally adjusted to address memory locations 0100_{16} through $01FF_{16}$, and an 8-bit flags register which holds the current status of the machine.

op code	operation	N	V	B	D	I	Z	C	ACCUMULATOR	IMMEDIATE	ZERO PAGE	ZERO PAGE, X	ZERO PAGE, Y	ABSOLUTE	ABSOLUTE, X	ABSOLUTE, Y	IMPLIED	RELATIVE	INDEXED INDIRECT	INDIRECT INDEXED	ABSOLUTE INDIRECT
ADC	Add with Carry to A	*	*				*	*		●	●	●		●	●	●			●	●	
AND	And to A	*					*			●	●	●		●	●	●			●	●	
ASL	Shift Left One Bit	*					*	*	●		●	●		●	●						
BCC	Branch on Carry Clear																	●			
BCS	Branch on Carry Set																	●			
BEQ	Branch on Zero Result																	●			
BIT	Test Bits in Memory with A	*	*				*				●			●							
BMI	Branch on Minus Result																	●			
BNE	Branch on Result not Zero																	●			
BPL	Branch on Result Positive																	●			
BRK	Force Interrupt or Break			1		1															
BVE	Branch on Overflow Clear																	●			
BVS	Branch on Overflow Set																	●			
CLC	Clear Carry Flag							0									●				
CLD	Clear Decimal Mode				0												●				
CLI	Clear Interrupt Disable					0											●				
CLV	Clear Overflow Flag		0														●				
CMP	Compare Memory and A	*					*	*		●	●	●		●	●	●			●	●	
CPX	Compare Memory and X	*					*	*		●	●			●							
CPY	Compare Memory and Y	*					*	*		●	●			●							
DEC	Decrement Memory By One	*					*				●	●		●	●						
DEX	Decrement X by One	*					*										●				
DEY	Decrement Y by One	*					*										●				
EOR	Exclusive-or Memory with A	*					*			●	●	●		●	●	●			●	●	
INC	Increment Memory by One	*					*				●	●		●	●						
INX	Increment Index X by One	*					*										●				
INY	Increment Index Y by One	*					*										●				
JMP	Jump to New Location													●							●
JSR	Jump and Save Return													●							
LDA	Load Acc from Memory	*					*			●	●	●		●	●	●		●	●		
LDX	Load X from Memory	*					*			●	●		●	●		●					
LDY	Load Y from Memory	*					*			●	●	●		●	●						
LSR	Shift Right One Bit	0					*	*	●		●	●		●	●						
NOP	No Operation																●				
ORA	Or Memory to A	*					*			●	●	●		●	●	●			●	●	
PHA	Push Acc onto Stack																●				
PHP	Push Status onto Stack																●				
PLA	Pull Acc from Stack	*					*										●				
PLP	Pull Status from Stack	*	*	*	*	*	*	*									●				
ROL	Rotate One Bit Left	*					*	*	●		●	●		●	●						
ROR	Rotate One Bit Right	*					*	*	●	●	●	●		●	●						
RTI	Return from Interrupt	*	*	*	*	*	*	*									●				
RTS	Return from Subroutine																●				
SBC	Subract Memory and Carry from A	*	*				*	*		●	●	●		●	●	●			●	●	
SEC	Set Carry Flag							1									●				
SED	Set Decimal Mode				1												●				
SEI	Set Interrupt Disable Status					1											●				
STA	Store Acc in Memory										●	●		●	●	●			●	●	
STX	Store X in Memory										●		●	●							
STY	Store Y in Memory										●	●		●							
TAX	Transfer A to X	*					*										●				
TAY	Transfer A to Y	*					*										●				
TSX	Transfer SP to X	*					*										●				
TXA	Transfer Index X to A	*					*										●				
TXS	Transfer Index X to SP																●				
TYA	Transfer Index Y to A	*					*										●				

Figure 3.
The 6502 operations, the condition codes they affect, and the addressing modes provided for each operation.

6502 are all relative and are limited in range to −128 bytes back or +127 bytes forward from the instruction following the branch. Longer branches, if they are needed, are constructed by inverting the branch condition (not-equal rather than equal, for example) and branching over an unconditional jump to the desired location. Jumps are not conditional and may be either direct or indirect.

Another mechanism allows for subroutines or procedures. A JSR instruction places the address following itself onto the stack and then jumps to the specified location. A RTS instruction executed later causes the program to resume execution at the location following the most recent JSR. The stack is limited to 256 bytes (one page) so the maximum subroutine nesting level is 128. The stack can also be used to store data using the PHA and PLA, PHS and PLS instructions. We can now try our hand at a few small 6502 programs to see exactly how these instructions can be fit together to accomplish useful work. We'll write things out in assembler language rather than machine language so that they are human readable to some extent. We'll use variable names like A and TEMP, but you should remember that *variable names in an assembler represent addresses not values.* The value of a name, say A, is its place in memory and not the contents of that place in memory. To get the contents you have to load the value into a working register like the accumulator.
Add a constant to a variable. Suppose we want to add 5 to the value of byte variable A. This can be done using the instructions

```
CLC
LDA     A
ADC     #5
STA     A
```

The clear carry instruction is necessary since we do not know what the value of carry is when we enter this sequence. If we had wanted to add one, we could have used

```
INC     A
```

which is provided in the instruction set because it (and decrement by one) are common programming operations.

Add two 16-bit variables. Suppose we want to have two 16-bit numbers stored in A and A + 1 and B and B + 1. We want to add them and store the result back into the storage space for A. Furthere, let's assume that A contains the least significant byte and A + 1 contains the most significant byte. To form the sum we use

```
CLC
LDA     A
ADC     B
STA     A
LDA     A + 1
ADC     B + 1
STA     A + 1
```

Notice that the carry is propogated from the low order byte addition to the high order byte addition just as it should be. This same approach can be generalized to additions of arbitrary length.

Coin Flipper. This is a subroutine which is meant to be called to get the result of flipping a coin; it appears in a different form in the Programmer's Toolbox in this issue. Executing the instruction

```
JSR     FLIP
```

sets the accumulator to either zero (heads) or one (tails). Somewhere in memory there is a two-byte number, SEED, which has been initialized to some non-zero value. The FLIP routine is

```
FLIP  CLC
      ROL     SEED
      ROL     SEED+1
      BCC     FL1
      LDA     SEED
      EOR     #$03
      STA     SEED
      LDA     SEED +1
      EOR     #$80
      STA     SEED + 1
      LDA     #1
      RTS
FL1   LDA     #0
      RTS
```

Now this is a bit more complicated. The sequence of ROL's double the value of SEED in memory. The high order bit is tested and if it is on, the seed is exclusive-or'd with the constant $8003 and the result one is returned. If the high order bit is off, then the result zero is returned.

In actual fact the information we want is stored in the high order bit of the constructed SEED and could be recovered uniformly to produce a more compact routine. That is left as an exercise for the reader.

Code Translation. Sometimes one wants to convert from one character set to another, say from ASCII to EBCDIC or vice-versa. The routine below does that efficiently. The character to be converted is stored in the X register and then the routine is called with a JSR. The routine replaces the value in the X register with the translated value.

```
CONVERT    PSA
           LDA     TABLE,X
           TAX
           PLA
           RTS
```

The address TABLE is the first (0-th) element of a table of codes whose value is the new (translate) code and whose ordinal position is the old code. For character data this table would be either 256 or 128 bytes long, depending upon the local conventions about parity.

The stack operations PSA and PLA are used to save and restore the value of the accumulator to prevent the user from inadvertently destroying useful data.

String Copy. Strings are simply linear arrays of characters. In one useful representation one associates a string with its starting address and agrees that all strings will be terminated by a zero byte (NUL). Passing around variables containing the address of the start of the string is then equivalent, in some sense, to passing around the string itself.

The problem at hand is to copy one string into another. The routine below does this using the powerful addressing modes of the 6502. Two pointer variables are allocated in page zero. These are S and D; both are 16-bit values. We initialize S to the address of the first character of the string we want to copy, and we initialize D to the address of the area in memory into which we want to copy the string. Then we call the routine with a JSR

```
SCOPY      LDX     #0
LOOP       LDA     (S,X)
           STA     (D,X)
           BEQ     DNE
           INC     S
           BNE     SC1
           INC     S + 1
S1         INC     D
           BNE     LOOP
           INC     D + 1
           BNE     LOOP
DNE        RTS
```

This program destroys both the A and X registers. It uses the indirect addressing modes to load characters and store them. The increment and branch operations add one to the 16-bit values of the pointers to select the next element. The value of the condition code is set by the load and tested to determine whether the routine should terminate.

Machine language programming is really simply a matter of fitting together some complicated low level primatives to make higher level operations, and then using those operations to make even higher level primatives. It has its own fascinations; one of the joys of programming is to find a way to construct a really elegant (and often, sad to say, inscrutable) way to solve a problem on a particular machine. Enjoy.

Figure 4.
6502 Instruction Formats by Addressing Mode

OP	operation code
A_L	low order address byte
A_H	high order address byte
D	displacement

example	format	mode
ROL	OP	ACCUMULATOR
ADC #1	OP DATA	IMMEDIATE
ADC $F010	OP A_L A_H	ABSOLUTE
ADC $0F	OP A_L	ZERO PAGE ADDRESSING
ADC $0F, X ADC $0F, Y	OP A_L	INDEXED ZERO PAGE ADDRESSING
ADC $F010, X ADC $F010, Y	OP A_L A_H	INDEXED ABSOLUTE ADDRESSING
TSX	OP	IMPLIED
BEQ * + 5	OP D	RELATIVE ADDRESSING
ADC ($0F, X)	OP A_L	INDEXED INDIRECT ADDRESSING
ADC ($0F),Y	OP A_L	INDIRECT INDEXED ADDRESSING
ADC($F010)	OP A_L A_H	ABSOLUTE INDIRECT

THE ELECTRIC PHONE BOOK

A Directory of 144 Computerized Bulletin Board Systems

A computerized bulletin board works just like an ordinary bulletin board system except that instead of paper and thumbtacks it uses a terminal, a computer, and the dial-up telephone network. It's a place to leave messages for everyone or for some particular person who, you know, browses the bulletin board occasionally.

The list below was developed from several sources including the Peripheral People in Mercer Island, Washington and the People's Message System in Santee, California. It is being maintained by PCC's PCNET project, our effort to bring computers and telecommunications into the hands of everyone. While this is the most complete listing we have as of this writing, we would appreciate additions and corrections. Send them to PCNET, PCC, P.O. Box E, Menlo Park, CA 94025 or leave them on the PCNET/PCC Bulletin Board System for Dave Caulkins, (415) 948-1474.

All the bulletin board systems listed here can be accessed by telephone using a 300-baud ASCII terminal and a Bell 103 modem. Most use carriage-return as a speed recognition character, after which they are self-teaching. All are free to anyone who calls, unlike the Arpanet, which is restricted, and The Source and MicroNet which cost money. The list has been sorted by area code; consult your local telephone directory for geographical correspondence.

We are compiling a mailing list of people interested in computers and telecommunication. If you would like to be on it, send us a note at the address above. If you are interested in PAN, PCNET's computer mail system for the PET, send us a stamped, self-addressed envelope and we'll send you back information.

(201) 457-0893	(213) 795-3788	(314) 838-7784	(604) 687-2640	(714) 730-1206
(201) 835-7228	(213) 799-1632	(316) 746-2078	(607) 754-5571	(714) 739-0711
(201) 874-6833	(213) 799-6514	(319) 353-6528	(609) 983-5970	(714) 751-1422
(201) 891-7441	(213) 826-0325	(319) 557-9618	(612) 929-8966	(714) 772-8868
(201) 968-1074	(213) 828-3400	(404) 394-4220	(614) 272-2759	(714) 898-1984
(202) 337-4694	(213) 843-5390	(404) 733-3461	(614) 649-7097	(714) 962-7979
(205) 945-1489	(214) 288-4859	(404) 790-8614	(615) 254-9193	(801) 753-6800
(206) 244-5438	(214) 634-2668	(404) 939-1520	(617) 354-4682	(803) 270-5372
(206) 482-5134	(214) 634-2775	(404) 939-8429	(617) 431-1699	(803) 270-5392
(206) 524-0203	(214) 641-8759	(404) 953-0723	(617) 649-7097	(803) 279-5392
(206) 723-3282	(216) 745-7855	(405) 528-8009	(617) 864-3819	(803) 771-0922
(206) 937-0444	(305) 566-0805	(408) 241-1956	(617) 897-0346	(805) 484-9904
(209) 638-6392	(305) 689-3234	(408) 263-0248	(617) 963-8310	(813) 223-7688
(212) 448-6576	(305) 772-4444	(415) 348-2139	(702) 873-9491	(816) 861-7040
(213) 276-4276	(305) 821-7401	(415) 348-2396	(703) 281-2125	(816) 931-3135
(213) 316-5706	(305) 989-9647	(415) 493-7691	(703) 734-1387	(817) 855-3916
(213) 329-3715	(312) 255-6489	(415) 661-0705	(703) 750-0930	(817) 855-3918
(213) 340-0135	(312) 269-8083	(415) 792-8406	(703) 893-9474	(817) 923-0009
(213) 349-5728	(312) 337-6631	(415) 948-1474	(703) 978-7561	(901) 276-8196
(213) 360-6332	(312) 420-7995	(417) 862-7852	(713) 693-8080	(901) 362-2222
(213) 394-1505	(312) 528-7141	(451) 948-1474	(713) 977-7019	(901) 761-4743
(213) 395-1592	(312) 622-9609	(502) 245-8288	(714) 449-5689	(904) 243-1257
(213) 396-3905	(312) 767-0202	(503) 646-5510	(714) 463-0461	(904) 243-8565
(213) 424-3506	(312) 964-7768	(512) 657-0779	(714) 526-3687	(913) 764-1520
(213) 428-4718	(313) 288-0335	(513) 874-2283	(714) 537-7913	(913) 782-5115
(213) 459-3177	(313) 465-9531	(516) 938-9043	(714) 565-0961	(915) 584-5393
(213) 566-8035	(313) 477-4471	(602) 866-0258	(714) 571-5550	
(213) 657-8803	(313) 484-0732	(602) 886-0258	(714) 582-9557	
(213) 675-8803	(313) 569-2063	(602) 955-1486	(714) 582-9957	
(213) 787-4004		(602) 957-4428		

AIM 65

SYM-1

KIM-1

MICRO™

The 6502 Journal

ATARI

PET

APPLE

OSI

MICRO is the quality reference journal devoted to the 6502 and 6502 based systems. MICRO has been published regularly since 1977, is now monthly and has over 10,000 readers throughout the world. MICRO is much more than 'just another computer magazine'. It is a continuing handbook of information relevant to the 6502 microprocessor, covering applications, systems, software, peripherals, reference materials, and more. If you are serious about your 6502 system, then you really should be getting MICRO.

Ordering Information: Circle selections and put total on bottom line

"**The BEST of MICRO Volume 1**" contains all important articles from issues 1–6 in book form: Surface $7.00 / Air $10.00

"**The BEST of MICRO Volume 2**" contains all important articles from issues 7–12 in book form: Surface $9.00 / Air $13.00

Individual back issues are available from issue 7. Check box for information on availability and price for each issue: ☐

Subscription rates: One Year = 12 issues.

Surface: US $15.00 / All Other Countries Surface $18.00

Air Mail: Central America $27.00 / Europe/So. America $33.00 / All Other Countries Air Mail $39.00

All payments in US dollars. Make checks payable to **MICRO**. Foreign payments by International Money Order. **Total: $** ....

Name: ....

Address: ....

....

City: .... State: .... Zip: ....

Country (if not U.S.): ....

Help **MICRO** bring you the info you want by completing this short questionnaire.

Microcomputers Owned/Planning to Buy: AIM SYM KIM PET APPLE OSI Other: ....

Peripherals Owned/Planning to Buy: Memory Disk Video Printer Terminal Other: ....

Microcomputer Usage: Educational Business Personal Control Games Other: ....

Languages Used: Assembler BASIC FORTH PASCAL Other: ....

MICRO
PO Box 6502
Chelmsford, Mass 01824
617-256-5515

MICROPROCESSOR CONTROLLED BRA

BY C. S. D. CLANTON, M.D.

SAFETY FIRST

D. Green Electronics of Glasgow, Scotland, may soon market microprocessor-controlled brassieres that determine the safe and unsafe periods in a woman's ovulation cycle by analyzing variations in breast temperature. The micro would automatically measure and analyze variations in breast temperature, providing a real-time readout of fertility. Not yet finalized is the location for the display.

From *EE Times*, December 10, 1979.

Because I am both a physician and a computer scientist, I try to keep up with interesting developments in both areas. When Dennis Allison called the other day to ask about a prospective microcomputer product for a medical application, I must admit I had not heard about it. The product, located in a brassiere, is intended to provide an indication of the safe and unsafe periods in a woman's menstrual cycle by analyzing variations in breast temperature. My immediate reaction, similar to that of many people when it comes to considering the contents of brassieres, was one of excitement. It seemed an answer to two problems – finding something useful to do with microprocessors and solving a significant social problem. After a few seconds reflection, my enthusiasm waned. The principle of operation of this device does not seem practical.

The device evidently measures breast skin temperature in an attempt to predict ovulation. There is a half degree rise of basal body temperature at the time of ovulation which can be used with some confidence for family planning, in particular the spacing of children, though it is not sufficiently reliable to prevent pregnancy with the same confidence as other more widely used birth control methods. Also, it IS still a rhythm method. That is, some other method of contraception must be used for much of the cycle. (Abstention is sometimes the method chosen for the unsafe period though many people find it less than ideal.) The basal body temperature merely keeps the beat somewhat better than counting backwards fourteen days from the time the next menstrual period is expected. Women who chart their basal body temperatures often find that their period does not invariably come exactly fourteen days after the temperature shift anyway, a demonstration of the difficulty of predicting the present from the future. For many people, any available form of contraception is often better than none. So if it worked, this device might still serve a role as one of the birth control options. In addition, though I saw no mention of it in the news release, the ability to predict fertility is especially useful for women who are trying to get pregnant. The theory of operation seems correct except for one small hitch.

The basal body temperature is the body's temperature at complete rest, immediately upon waking up, before any activity whatsoever except turning off the alarm clock and reaching for the thermometer. Further, it is a body core temperature. This device presumably measures breast skin temperature and does some form of integration of this temperature over time. If that presumption is true, it presents a serious problem for at least two reasons.

In ordinary life outside the research laboratory, a reasonably accurate core temperature can be obtained with a rectal thermometer. An oral thermometer gives a temperature about 1 degree F lower but is a fairly reliable indicator of the core temperature as long as it is properly placed, no hot or cold food or liquids have been recently consumed, and the mouth is kept closed. A temperature taken from the armpit is lower still and even more difficult to acquire with any accuracy or reliability. The armpit is too far from the body core and too well insulated. Well, the skin is even worse, as those who have seriously tried to correlate forehead temperature with fever know, and the breast poses even more problems for the testing of core temperature. Most of the breast is fat. The actual milk-producing glandular tissue in the breast is not very large. It is the rather firm, somewhat lumpy tissue that can be readily felt with the fingers as the breast is (gently!) rolled over the ribs in a small-breasted woman. (Science can be fun.) The rest is fat. Fat is a good insulator of body heat because it does not have much blood flow to convey heat through it. Thus, the skin of the breast overlies a layer of insulating fat that is different for every woman. This in itself might not be too serious a problem. Since it is only necessary to detect the rise in basal body temperature, the absolute values are of little importance.

Unfortunately, the skin temperature varies by a great deal more than the half degree of basal body temperature change mentioned above. The skin has an important role in the maintenance of a constant core temperature. Thus, when you take a hot bath, your skin turns red. That is due to the marked increase in blood circulating through the skin. This superficial blood radiates heat readily, thus eliminating the extra body heat acquired from the hot water. If you go out naked on a cold day, you will see the opposite effect. Your skin will blanch. The blood flow has been markedly decreased to minimize heat loss to the environment. If you stay out long enough, you will start shivering – an involuntary muscular activity to increase heat production to help maintain your core temperature.

While it is true that a woman's core temperature is higher over the days following ovulation so that she must on average be conserving more heat through her skin than before, I would expect this difference to be very small compared to the effects on skin temperature of the weather or sleeping with more or less covering or adjacent warm bodies. Another interesting influence on breast skin temperature is sexual activity. A woman's breasts engorge with blood early in sexual arousal; it is one of the causes of nipple erection. About half of young women have a sex tension flush over their breasts just prior to orgasm, though this occurs less frequently in older women. Again, these changes in blood flow pattern result in marked changes in skin temperature. They all may be randomly distributed, but over the course of such a short period as a month, they would almost certainly swamp any small regular variation in skin temperature.

Finally, the most telling blow of all relates to the carrier for this device – the brassiere. Obviously, this apparel, if worn at all, is only located over the breast during the day while the woman is active. The day may be cold or hot. She may have remembered her sweater or not. The man next to her on the bus may have been extremely attractive, or she may have been too sleepy to notice. While the measurement of her breast skin temperature under all these circumstances may be scientifically valuable and personally interesting, its relevance for information about fertility seems dubious. The woman undoubtedly will never even wear this special purpose garment when she is in a basal metabolic state, after hours of sleep.

Of course, there may be some interesting algorithm or unusual property of the breast skin measurement that has escaped my attention. I certainly do not know everything about breast skin temperature. However, before risking possible unwanted pregnancy, thorough experimental data would be needed to convince me. Such experiments would have to be conducted in a substantial number of women over some period of time, to accumulate sufficient experience to establish statistically valid estimates of reliability and precision. I also tend not to find studies by those who have something to gain from the outcome very persuasive. In fact, I am generally reluctant both personally and professionally to use medical products that are new to the market. I prefer to wait a year or so to see how they work on other physician's patients and what the real problems are before I use them on myself or anyone else. Usually in that year the product that was heralded by everyone as "the best ever" at the time of its announcement has proven to be no better and often worse then the old tried and true. For contraception, I use and recommend the barrier methods primarily – condom, diaphragm, and foam – because of their lack of complications and effectiveness if used correctly (yes, there is a wrong way to put on a condom – without leaving any room in the tip). These can be combined if very high reliability is desired. Sometimes, I support other methods – the birth control pill, IUD, rhythm with or without basal body temperature or cervical mucous testing as adjuncts, or even the ever faithful withdrawal. It all depends on the objectives and values of the woman and her sexual partner. While I am most interested in both computers and sex (though not combined in the manner of "Demon Seed"), I remain skeptical of this particular application.

THE BASAL BODY TEMPERATURE METHOD OF BIRTH CONTROL

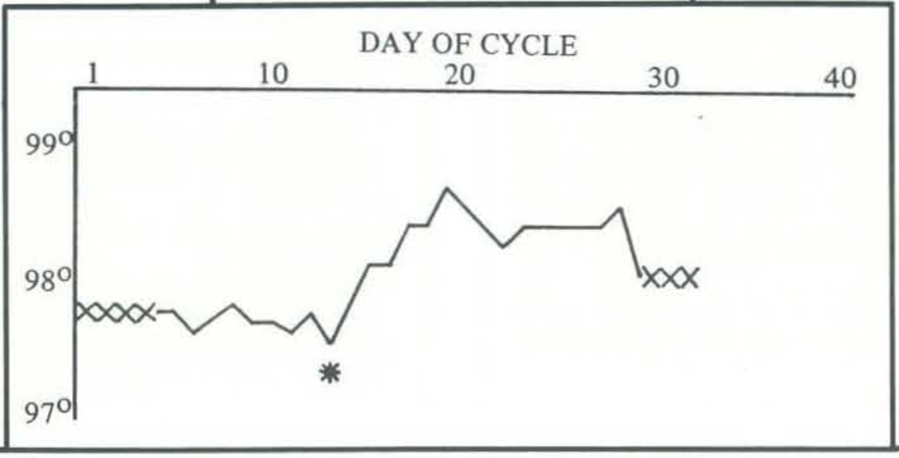
As a natural method of birth control, an accurate record of the basal body temperature (BBT) taken over a period of 3-4 months can reliably determine the end of the interval of fertility – the safe, infertile time after ovulation. BBT is defined as the lowest temperature reached by a healthy, rested body during waking hours. Immediately preceding ovulation, the BBT of the woman drops (about 0.2°F). During the following 24-72 hours, a reproductive hormone released after ovulation causes the temperature to rise several tenths of one degree (0.6-0.8°F) above the normal BBT. To avoid pregnancy, a woman should refrain from unprotected intercourse until her temperature has remained elevated for three consecutive days.

REFERENCES:

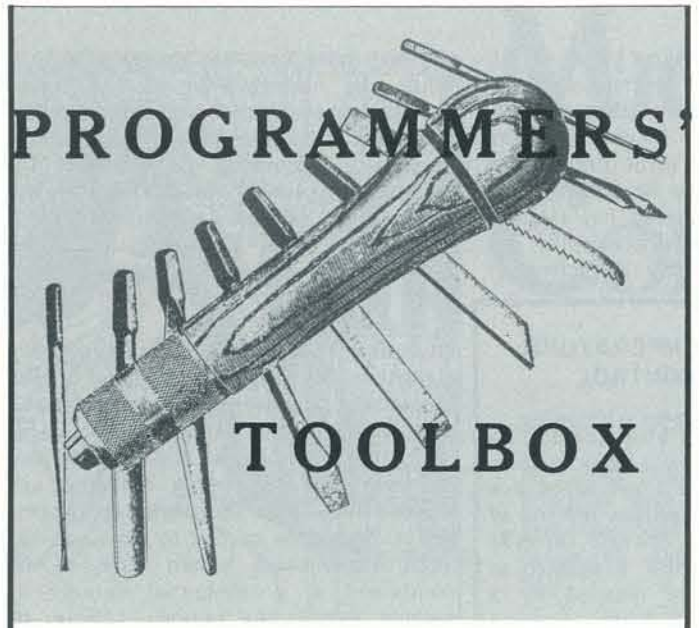
Contraceptive Technology 1978-1979, Robert A. Hatcher, M.D., et al., Irvington Publishers, Inc., New York. $3.25

A Cooperative Method of Natural Birth Control, Margaret Nofziger, The Book Publishing Company, Summerton, TN. $3.25

These books are available from the Education Department, Planned Parenthood Association of Santa Clara County, 17 North San Pedro St., San Jose, CA 95110 (prices include postage) or from your local Planned Parenthood Association.



The Basal Temperature will rise noticeably after ovulation



PT 19 : CREATING AND SAVING SHAPE TABLES FOR THE APPLE II

The Apple II and Apple II Plus provide powerful graphics capabilities, especially when shape tables are defined and manipulated. Unfortunately, there is no available software for the storage and retrieval of shape tables from disk, only from cassette. Additionally, Apple II Plus is not designed to operate in machine language. This renders invalid most of the discussion in the APPLESOFT Reference Manual about storing shape tables. What is required is the use of POKE to store the shape table as well as the address at which the shape table begins.

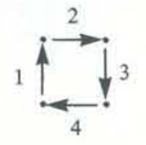
The following programs were designed to simplify the development of shape tables as well as their storage and retrieval. They were written in BASIC Applesoft and will run, as is, on a 48K Apple II or Apple II Plus with a DOS. If less memory is available, the value of the starting location of the shape table should be changed to a non-interfering memory location (lines 50, 60, 70, and 145 of SAVE SHAPE TABLES).

Prior to executing the program to save a shape table, the decimal values to be stored in the shape table need to be stored in a text file on disk. These values can be determined through the procedures described on pages 92-95 of the Applesoft BASIC Programming Reference Manual and using the BINARY to DECIMAL program defined below. The values can be stored using the MAKE TEXT program provided with the Apple II. Note that the values which should be stored for this program are DECIMAL, not HEXIDECIMAL. BINARY to DECIMAL will allow you to convert the binary code (described in the Applesoft Manual) to decimal. SAVE SHAPE TABLES will store the data in appropriate RAM locations and define the necessary pointers.

If you do not have a disk, these programs can still be used to place a shape table in memory by deleting lines 20, 22, 30, 40 and 140 of SAVE SHAPE TABLES and changing line 70 to I = 37888.

```
 10 REM BINARY TO DECIMAL
 20 INPUT A$
 25 T = 0
 30 X = 1
 40 I  = LEN (A$)
 45 IF I < = 0 THEN END
 50 FOR L = I TO 1 STEP - 1
 60 IF MID$ (A$, L, 1) = "1" THEN T = T + X
 65 X = X * 2
 70 NEXT L
 80 PRINT
 90 PRINT "DECIMAL VALUE = "; T
100 GOTO 20
110 END
              *    *    *    *    *
 10 REM SAVE SHAPE TABLES
 20 D$ = CHR$ (4)
 22 INPUT "NAME OF TEXT FILE?"; A$
 25 REM  OPEN DISK FILE
 30 PRINT D$"OPEN"; A$
 40 PRINT D$"READ"; A$
 45 REM SET SHAPE TABLE
 46 REM  POINTER
 50 POKE 232,0
 60 POKE 233,148
 70 I = 37887
 75 REM  READ AND STORE
 76 REM SHAPE TABLE VALUES
 80 INPUT D
 90 IF D > 255 THEN GOTO 140
100 POKE I, D
110 PRINT "AT ADDRESS"; I; "IS VALUE"; D
120 I = I + 1
130 GOTO 80
140 PRINT D$"CLOSE "; A$
143 REM SET HIMEM TO BEGINNING
144 REM OF SHAPE TABLE
145 HIMEM: 37887
150 END
```

EXAMPLE:
Shape Table Consisting of one square shape (4 moves)



Binary values for shape definition

```
00101100
00111110
00000000
```

SAMPLE RUN	COMMENTS
RUN BINARY TO DECIMAL ?00101100 DECIMAL VALUE=44 ?00111110 DECIMAL VALUE= 62 ? (Return)	Convert binary shape definition to decimal equivalents
MAKE TEXT Type String #1 : ?1 Type String #2 :?0	(Supplied with Apple II) Index to shape table (see page 95 of Applesoft Manual)
Type String #3 : ?4 Type String #4 : ?0 Type String #5 : ?44 Type String #6 : ?62 Type String #7 : ?0 Type String #8 : return	Shape definition
WHAT FILE NAME? SHAPE TABLE N RUN SAVE SHAPE TABLES NAME OF TEXT FILE? SHAPE TABLE N	

CONTRIBUTED BY RON LAUGHERY

PT 20 : A COIN FLIPPING ROUTINE

Many games and simulations need to make only binary decisions rather like flipping a coin. This routine is a good way to generate such sequences and, unlike a coin flipper based upon testing the value of RND( ), the number of runs generated will be close to that expected. It is presented below in the programming language C since it is very difficult to express in BASIC. The variable *seed* is a 16-bit integer which is initialized to some random number. The procedure *flip* does the flipping and returns a one or a zero (heads or tails, say) each time it is called. The operation '=+' adds *seed* to itself and stores the result back into *seed*; likewise '=^' performs the bit-wise exclusive-or of *seed* with the constant and stores the result in *seed*. The constant is expressed in octal in the program; in decimal it is 32771.

The program is based upon linear shift register sequences. See Stone, *Discrete Mathematical Structures and Their Applications*, Science Research Associates, Chicago, 1973 or Knuth, *Seminumerical Algorithms*, Addison-Wesley, Reading, Mass., 1969, for a discussion of the theory.

```
/*     a coin flipping routine based upon linear shift register
       sequences.
 *     the primitive polynomial here is X^15 + X + 1
 *
 *     cycle length is 32768
 *
 */

int seed; /* the random number seed */

flip ( ){
       seed =+ seed;
       if (seed <0)    {seed = ^ 0100003; return (1);}
       return (0);
 }
```

CONTRIBUTED BY DENNIS ALLISON

PT 21 : PRONOUNCEABLE NAMES

Here's a name generator for Microsoft style BASIC in the fashion of PT14 published in the Jan/Feb 1980 *RC*. It runs on the PET but should work with minor changes on most Microsoft BASICs. It may be necessary to change the RND function call.

```
480  REM  NAME GENERATOR SUBROUTINE
490  :
500  REM  MAKES UP NAMES YOU CAN
510  REM  PRONOUNCE. LENGTH IS
520  REM  RANDOMLY CHOSEN BETWEEN
530  REM  'SA' AND 'SB' SYLLABLES.
540  :
550  REM  NAME IS RETURNED AS 'A$'.
560  :
570  A$ = ""
580  FOR SY = 1 TO SA+(SB-SA+1) * RND (1)
590  : A$ = A$ + MID$ ("BCDFGHJKLMNPRSTVWYZ", 1+
     19* RND (1), 1)
600  : A$ = A$ + MID$ ("AEIOU", 1 + 5 * RND (1), 1)
610  NEXT SY
620  RETURN
```

One can still improve on this by changing the call to MID$ in line 600 to

MID$("AEEIOU" , 1 + 6 * RND (1))

The frequency of "E" will be closer to that in real names. Adding

605 IF RND (1) < 0.2 GOTO 600

inserts an occasional two-vowel syllable and a very occasional three-vowel one.

CONTRIBUTED BY JOHN TRENHOLME

PT 22 : PET INPUT WITHOUT BREAKS

Here are two techniques for improving the human factors for PET BASIC programs. The first prevents control from returning to BASIC when an empty line is input; the second allows a default input in response to a return. This trick, with a bit of careful program design and renumbering programs to start at 60,000 (you *do* have a Toolkit for your PET, don't you), will protect most first-time users from needless grief and confusion.

FOR PET

```
100 REM  THIS SHOWS HOW TO GET INPUT FROM THE KEYBOARD WITHOUT BREAKING OUT OF
110 REM  BASIC IF RETURN IS PRESSED WITHOUT TYPING ANYTHING ELSE
120 :
130 :
140 :
150 REM  METHOD 1 - NO DEFAULT INPUT
160 :
170 INPUT "HOW MANY EGGPLANTS ▮▮▮ "; A$
180 IF A$ = " " GOTO 170
190 A = VAL (A$)
200 :
210 REM  THE SECRET IS THAT THE STRING AFTER "EGGPLANTS" IN LINE 170 IS:
220 :
230 REM  [SPACE] [SPACE] [SHIFT-SPACE] [BACK] [BACK] [BACK]
240 :
250 REM  AND THE STRING CONSTANT IN LINE 180 IS A SHIFT-SPACE (PRESS SHIFT,
260 REM  HOLD DOWN WHILE PRESSING SPACE).
270 :
280 REM  THIS WORKS BECAUSE SHIFT-SPACE LOOKS LIKE A SPACE ON THE SCREEN, BUT
290 REM  THE "INPUT" CODE READS IT AS A NON-BLANK CHARACTER. IT IS PLACED SO
300 REM  THAT ANY INPUT THE USER TYPES WILL OVERWRITE IT.
310 :
320 :
330 :  REM  METHOD 2 -- DEFAULT INPUT DESIRED
340 :
350 Z = INT (1+100 * RND(1) ) -50 {RANDOM DEFAULT FOR THIS EXAMPLE}
360 PRINT SPC (7) Z
370 INPUT "▮WHICH"; A$
380 A = VAL(A$)
390 :
400 REM  HERE THE VALUE IN SPC (LINE 360) IS THE LENGTH OF THE QUESTION (HERE
410 REM  "WHICH" HAS LENGTH 5), PLUS 2. THE QUESTION THEN HAS A CURSOR-UP IN
420 REM  FRONT OF IT, TO GET BACK UP ON THE SAME LINE AS THE DEFAULT. IF THE
430 REM  USER PRESSES RETURN WITHOUT TYPING ANYTHING, THE DEFAULT IS ENTERED.
READY.
```

CONTRIBUTED BY JOHN TRENHOLME

SOFTWARE

NUMBER TRANSLATION PROGRAM

BY JEFFREY C. RUBLE

Most people have little difficulty in reading numbers with up to nine places to the left of the decimal (hundred millions). However, many people don't know the names of the numbers beyond nine places. Because of this I decided to write a BASIC program for level II 16K TRS-80 that will accept a number as input and yield as output the number's English translation. The program translates positive or negative numbers with or without digits to the right and/or left of the decimal point.

The basic strategy of the program is quite simple. The number to be translated is input as a string. If there are digits on both sides of the decimal points, the string is broken into two parts and each part is translated separately. The number of groups of three digits in each part is then determined. Each group represents a number from zero to nine hundred ninety-nine. After a group of three digits is translated, its order of magnitude (hundreds, thousands, etc.) is appended to the right of the three-digit translation. The place values and orders of magnitudes are stored in a table at the beginning of the program for later retrieval.

As an example, an input of –1021123789 .46237 would yield an output of "NEGATIVE ONE BILLION, TWENTY-ONE MILLION, ONE HUNDRED TWENTY-THREE THOUSAND, SEVEN HUNDRED EIGHTY NINE AND FORTY-SIX THOUSAND TWO HUNDRED AND THIRTY-SEVEN HUNDRED THOUSANDTHS."

```
10 REM            ****************************************
20 REM            *            NUMBER/BAS                *
30 REM            *        TRANSLATES NUMBERS            *
40 REM            *           INTO ENGLISH               *
50 REM            *          JEFFREY C. RUBLE            *
60 REM            *      PORT ANGELES, WA  02/27/80      *
70 REM            ****************************************
80 CLS:CLEAR100:DIMU$(20),T$(8),F$(5)
90 FORI=1TO20:READU$(I):NEXT:FORI=1TO8:READT$(I):NEXT
100 FORI=1TO2:READB$(I):NEXT:FORI=1TO5:READF$(I):NEXT
110 DATA"ONE","TWO","THREE","FOUR","FIVE","SIX","SEVEN","EIGHT"
120 DATA"NINE","TEN","ELEVEN","TWELVE","THIRTEEN","FOURTEEN"
130 DATA"FIFTEEN","SIXTEEN","SEVENTEEN","EIGHTEEN","NINETEEN"
140 DATA"TWENTY","TWENTY","THIRTY","FOURTY","FIFTY","SIXTY"
150 DATA"SEVENTY","EIGHTY","NINETY"," ","THOUSAND"
160 DATA"TENTHS","HUNDRETHS","THOUSANDTHS","TEN-THOUSANTHS","HUNDRED-THOUSANDTHS"
170 PRINT"THIS PROGRAM WILL TAKE A NUMBER THAT YOU INPUT AND TRANSLATE"
180 PRINT "IT INTO ENGLISH.  YOU ARE LIMITED TO 6 PLACES TO THE LEFT"
190 PRINT "OF THE DECIMAL AND/OR 5 PLACES TO THE RIGHT.  NO COMMAS ARE"
200 PRINT "ALLOWED."
210 M$="":F$="":PRINT:INPUT"WHAT IS YOUR NUMBER";M$:PRINT
220 IF LEFT$(M$,1)="-" THEN PRINT "NEGATIVE":M$=RIGHT$(M$,LEN(M$)-1)
230 GOSUB 520
240 IF VAL(M$)=0 AND VAL(F$)=0 THEN PRINT "ZERO":GOTO 590
250 R=LEN(M$)-FIX(LEN(M$)/3)*3
260 IF R=1 THEN M$="00"+M$:GOTO 280
270 IF R=2 THEN M$="0"+M$
280 FOR I = 1 TO (LEN(M$)/3)
290 N$=MID$(M$,1+3*(I-1),3):IF VAL(N$)=0 THEN 310
300 GOSUB 450  :PRINT " ";B$(LEN(M$)/3-I+1)
310 NEXT I
320 IF F1$<>"Y" THEN 590
330 IF VAL(F$)=0 THEN 590
340 IF VAL(M$)=0 THEN 360
350 PRINT"AND"
360 R=LEN(F$)-FIX(LEN(F$)/3)*3
370 IF R=1 THEN F$="00"+F$
380 IF R=2 THEN F$="0"+F$
390 FOR I = 1 TO (LEN(F$)/3)
400 N$=MID$(F$,1+3*(I-1),3)
410 IF VAL(N$)=0 THEN 430
420 GOSUB 450  :PRINT " ";B$(LEN(F$)/3-I+1)
430 NEXT I
440 PRINT F$(X):GOTO 590
450 REM                'FIND THE THREE PLACE VALUE
460 H=VAL(LEFT$(N$,1))
470 IF H=0 THEN 490
480 PRINT U$(H);" HUNDRED ";
490 R2=VAL(RIGHT$(N$,LEN(N$)-1)):IF R2<21 THEN PRINT U$(R2);:GOTO 510
500 M=VAL(MID$(N$,2,1)):R1=VAL(RIGHT$(N$,LEN(N$)-2)):PRINT T$(M-1);"-";U$(R1);
510 RETURN
520 REM                'BREAK NUMBER UP INTO WHOLE AND FRACTIONAL PARTS
530 FOR I = 1 TO LEN(M$)
540 IF MID$(M$,I,1)="." THEN F$=RIGHT$(M$,LEN(M$)-I):M$=LEFT$(M$,I-1):F1$="Y":X=LEN(F$):GOTO 560
550 NEXT I
560 IF LEN(M$)>6 THEN PRINT "SORRY! ONLY 6 PLACES TO LEFT OF DECIMAL ALLOWED":GOTO 210
570 IF LEN(F$)>5 THEN PRINT "SORRY! ONLY 5 PLACES TO THE RIGHT OF THE DECIMAL ALLOWED":GOTO 210
580 RETURN
590 PRINT:INPUT"ANOTHER NUMBER";K$:IF LEFT$(K$,1)<>"N" THEN CLS:GOTO 210  ELSE CLS:END
```

The original program translated numbers with thirty places to the left of the decimal (100-octillions) and/or twenty-nine places to the right of the decimal (100-octillionths). For the sake of brevity I have included a listing for a version of the program which handles numbers with six places to the left and five places to the right of the decimal. To modify this version to perform as my original program, change line 100 so that in B$(I) I ranges from 1 to 10 and in F$(I) I ranges from 1 to 29. Add to line 150 the data "MILLION," "BILLION," . . . , "OCTILLION." To line 160 add the data "MILLIONTHS," "TEN-MILLIONTHS," . . . , "HUNDRED-OCTILLIONTHS." Lines 180, 190, 560 and 570 would have to be redone to reflect the extended range. Finally, in line 80 more string space should be "CLEARED" and F$(5) changed to F$(29).

BANAL TUNE GENERATOR FOR THE ATARI

BY DENNIS ALLISON

In the February 1956 issue of *Scientific American*, Richard C. Pinkerton published an article, "Information Theory and Melody," in which he analyzed the nature of nursery rhyme tunes using probability theory. One result of his analysis was a state-transition table for composing simple nursery-rhyme-like tunes. The program below not only composes the tune, but plays it as well, using the Atari's built-in music generator. The tunes it generates sound a bit like nursery rhymes but certainly fit Pinkerton's characterization – "banal."

All songs are in the key of C and are single octave only. While these restrictions were plausible when the tune generation was to be done by hand, today we could do far better. Still, the simplicity of this scheme is appealing.

In the program below, the value of T determines the duration of each note, the value of I specifies the current state, and the value of N the next note to play. The arrays S and F store the state-transition table with two entries per state. The next state is selected by randomly selecting either the value of S(I) or S(I+1) and playing F(I) or F(I+1), respectively, to get there. A zero value of the note stored in the F array implies that there is to be no change from the previous value.

When you have the program working, try adding additional voices (the Atari has a total of four) which are harmonically related to the first. In the SOUND statement, the first parameter gives the voice, the second the note, the third the tone distortion (10 for a pure tone), and the last the amplitude. Add new SOUND statements with note values of N+N and N/2. Change the relative amplitudes as well so that the total amplitude (that is, their sum) does not exceed 32. You might also experiment with the distortion field; distortion values can be any *even* number between 0 and 14.

```
050 DIM  S(24),  F(24)
100 REM          INITIALIZE
110 T = 20
120 N = 121
130 FOR I = 1 TO 24
140 READ U, V
150 S(I) = U
160 F(I) = V
170 NEXT I
180 I = I
190 REM NOW CYCLE THROUGH STATES
200 SOUND 0, N, 10, 10
210 FOR J = 1 TO T: NEXT J
220 N = F(I)
230 K = I
240 I = S(I)
250 IF RND (0) >0.5  THE I = I+1
260 IF F(K) <=0 THEN GOTO 210
270 GOTO 200
490 REM TABLE OF STATES AND NOTES
500 DATA 3, 121, 23, 0
510 DATA 5, 121, 5, 121
520 DATA 7, 121, 19, 0
530 DATA 9, 0, 9, 0
540 DATA 11, 108, 15, 121
550 DATA 1, 121, 1, 121
560 DATA 23, 0, 23, 0
570 DATA 13, 108, 13, 108
580 DATA 15, 121, 11, 108
590 DATA 9, 0, 17, 96
600 DATA 19, 82, 7, 108
610 DATA 21, 0, 21, 0
```

TWISTER MOVE GENERATOR

BY ROD HALLEN

Twister is a game produced by the Milton-Bradley Company. I have seen it used as an icebreaker at both adults' and children's parties. Our Twister mat has been gathering dust lately because the move selector has gotten warped and the spinner always stops at the same place.

Why not let your computer generate moves for you? The program accompanying this article does just that. Each time the enter key is pressed, a new move is displayed. I would expect that the computer generated moves are at least as random as those selected by the original spinner board.

This program was written for the Level II TRS-80 but should transport easily to other systems.

```
 10   REM * TWISTER MOVE SELECTOR *
 20   REM * ROD  HALLEN,  PO  BOX  73
 30   REM * TOMPSTONE  AZ  85638
 40   C$(1) = "RED" : C$(2) = "BLUE"
 50   C$(3) = "YELLOW": C$(4) = "GREEN"
 60   E$(1) = "LEFT FOOT"
 70   E$(2) = "LEFT HAND"
 80   E$(3) = "RIGHT HAND"
 90   E$(4) = "RIGHT FOOT"
100   CLS
110   PRINT  "COMPUTERIZED  TWISTER"
130 PRINT "HIT ENTER TO SEE NEXT MOVE"
140 X = RND(4) : Y = RND(4)
150   PRINT @ 448, " "; : INPUT  X$
160   PRINT  @  454,  E$ (X) ; "^" ; C$ (Y) ; " " ;
170   GOTO 140
180   END
```

ROLLING TITLES

BY GRAHAM K. JENKINS

SUMMARY

This article describes and illustrates the use of a BASIC subroutine for generating large (7X5 matrix) headings on a display screen. Headings whose width exceeds that available on the screen are rolled horizontally across it in a manner similar to that employed in bank and travel service promotional displays.

AN END TO DOT AND SQUARE TEDIUM

At some time or another, most of us have had a need to display one or more words in large characters, either for exaggerated emphasis, or to enable the display to be read at some distance.

Usually the programmer must sit down with a piece of graph paper and (very essential!) an eraser, and place dots or asterisks in appropriate squares so as to form a matrix version of his large-character message. That done, he proceeds to code it directly in PRINT statements. The whole exercise amounts to a case-study in tedium.

To those of you who have been caught in that sort of situation, the subroutine in this article will seem like the answer to a prayer. All you need do is place the string for large-character display in the variable L$, then execute the subroutine. If the string is short enough to be accommodated within the screen width, it goes up and stays there; otherwise, it rolls horizontally across the screen from left to right just like the dot-matrix displays used in banks and travel agencies. The first few lines of the program listing show how it is done. (The business with C and C$ is to create a form-feed character and clear the screen; many computers will provide a simpler mechanism than this.)

COMPRESSED CHARACTER STORAGE

Our subroutine employs a 7x5 matrix for display (as shown in the sample output) of the characters whose ASCII representations lie between 32 (decimal) and 90 inclusive. In other words, it can display all the numerals and uppercase alphabet characters, and most of the punctuation and arithmetic characters available on the standard keyboard. Characters outside the range (e.g. lowercase alphabet characters) are mapped to appropriate equivalents.

It would, of course, be possible to store each of the 59 keyboard characters as a set of 7 row strings, each containing 5 characters. Thus, the "H" in our sample output would be stored as "$ $", "$ $", "$ $", "$$$$$", "$ $", "$ $", "$ $". This technique would utilize more than 35 bytes per character (allowing for string termination overheads), and would use up an awful lot of memory. We have elected instead to store each character as three 14-bit integers. The first integer contains the first column of the character in bit form, while the remaining two integers each contain two columns. Thus, our "H" is represented (in binary) as 0000000 1111111, 0001000 0001000, 0001000 1111111 – or 127, 1032, 1151 (decimal – see line 820).

CHARACTER DISPLAY

Characters are unpacked and built into line strings for display as required in a series of nested loops. All variables (both string and integer) modified in the subroutine commence in either "R" or "S", to help minimize name conflicts. A screen with a width capacity of 80 characters can accommodate 11 whole matrix-form characters (each separated by two spaces for readability), together with small parts of preceding and subsequent characters; line strings are therefore constructed for groups of 13 characters, then pruned at each end for displaying appropriate segments. The compressed characters are read directly from the data area in the R4 loop (line 475) each time they are required, the entire set being read each time for a constant time delay. For simplicity, and to ensure an even motion in rolling displays, the line strings are rebuilt in entirety for each display step. Lines 645 through 655 of the subroutine are used to move the cursor upwards on the screen at the end of each display step except the last; some modification to these may be necessary according to implementation.

MAKING IT SLOWER – OR FASTER

If the characters happen to roll too quickly on your system, it is a simple matter to insert an additional delay loop within the innermost (R6) loop. To speed up the character roll, you could read the character data once into an appropriate array upon entering the subroutine, then directly access the required data in the array whenever it is required thereby eliminating the R4 loop. Alternatively, or as well, you might build the line strings for the initial 13 characters once upon entry, then subsequently delete the first character in each string and build onto the end of it in each pass within the R2/R3 loops.

```
---------------------
|  PROGRAM LISTING  |
---------------------

100 REM   ************************************************************
105 REM   *        ROLLING TITLE DEMONSTRATION PROGRAM               *
110 REM   * .. IMPLEMENTED BY GRAHAM K JENKINS IN MELBOURNE,         *
115 REM   *   AUSTRALIA DURING AUGUST, 1979  ..                      *
120 REM   ************************************************************
125 REM
130 REM
135 REM   SET THE MARGIN, CLEAR THE PAGE -
140 MARGIN #0,80
145 C(0) = 1
150 C(1) = 12
155 CHANGE C TO C$
160 PRINT C$;
165 REM
170 REM   ONE-LINE STATIC DISPLAY -
175 L$ = "  HELLO!"
180 GOSUB 300
185 PRINT
190 REM
195 REM   HORIZONTALLY ROLLING DISPLAY -
200 L$ = "WELCOME TO OUR DEMONSTRATION; WE HOPE YOU ENJOY IT."
205 GOSUB 300
210 GO TO 160
215 REM
220 REM
300 REM   ************************************************************
305 REM   *          LARGE CHARACTER DISPLAY SUBROUTINE              *
310 REM   ************************************************************
315 REM   * THE CHARACTERS IN L$ ARE DISPLAYED IN EXPANDED (7X5      *
320 REM   * MATRIX) FORM. IF THERE ARE MORE THAN 11 CHARACTERS,      *
325 REM   * THE CHARACTERS ARE ROLLED HORIZONTALLY. THE CURSOR       *
330 REM   * POSITION ON ENTRY DETERMINES THE LINE AT WHICH DISPLAY *
335 REM   * COMMENCES. ON EXIT, THE CURSOR IS LEFT AT THE LAST       *
340 REM   * DISPLAY LINE.                                            *
345 REM   ************************************************************
350 REM
355 DIM R$(7)
360 R$(0) = L$ & "              "
365 REM
370 REM   THE NEXT 13 CHARACTERS ARE TAKEN AND MAPPED TO UPPER CASE;
375 REM   SEVEN ROW STRINGS ( R$(1) .. R$(7) ) ARE THEN CONSTRUCTED,
380 REM   AND THESE ARE APPROPRIATELY PRUNED AT EACH END, THEN
385 REM   DISPLAYED. THE STRINGS ARE REBUILT IN ENTIRETY FOR DISPLAY
390 REM   IN THE NEXT (HORIZONTALLY SHIFTED) POSITION, THEREBY
395 REM   PROVIDING APPROPRIATE AND CONSTANT DELAYS. STRING
400 REM   COMPONENTS ARE DETERMINED (EACH TIME) FROM THE DATA LIST,
405 REM   SO AS TO AVERT THE NEED FOR A LARGE ARRAY.
410 FOR R1 = 1 TO LEN(L$)
415   FOR R2 = 1 TO 7
420     FOR R6 = 1 TO 7
425       R$(R6) = ""
430     NEXT R6
435     FOR R3 = R1 TO R1+12
440       S$ = SST(R$(0),R3,1)
445       CHANGE S$ TO S
450       IF S(1) <= 90 THEN 460
455       S(1) = S(1)-32
460       IF S(1) >= 32 THEN 470
465       S(1) = 32
470       RESTORE
475       FOR R4 = 32 TO 90
480         READ R(2), R(4), R(6)
485         IF R4 <> S(1) THEN 575
490         REM
495         REM   UNPACK CHARACTER DATA COLUMNS INTO ROW STRINGS -
500         FOR R5 = 2 TO 6 STEP 2
505           R(R5-1) = INT( R(R5)/128 )
510           R(R5)   = R(R5) - 128*R(R5-1)
515         NEXT R5
520         R(7) = 0
525         FOR R5 = 1 TO 7
530           FOR R6 = 1 TO 7
535             S$ = " "
540             IF R(R5) < 64 THEN 555
545             R(R5) = R(R5) - 64
550             S$ = "$"
555             R$(R6) = R$(R6) & S$
560             R(R5) = R(R5) * 2
565           NEXT R6
570         NEXT R5
575       NEXT R4
580     NEXT R3
585     REM
590     REM   ROW STRINGS COMPLETED; DISPLAY APPROPRIATE PORTIONS
595     REM   THERE-OF -
600     PRINT
605     FOR R6 = 1 TO 7
610       S$ = SST( R$(R6),R2,80 )
615       PRINT S$
620     NEXT R6
625     IF LEN(L$)    < 12 THEN 685
630     IF LEN(R$(0)) < 14 THEN 685
635     REM
640     REM   MOVE THE CURSOR BACK UP TO THE START POSITION -
645     S(1) = 17
650     CHANGE S TO S$
655     PRINT S$;S$;S$;S$;S$;S$;S$;S$;
660     REM
665     REM   LOOP TO RE-BUILD THE ROW STRINGS FOR THE NEXT
670     REM   DISPLAY POSITION -
675   NEXT R2
680 NEXT R1
685 RETURN
690 REM
695 REM   CHARACTER DOT MATRIX INFORMATION IS CONTAINED IN THE
700 REM   DATA STATEMENTS WHICH FOLLOW. EACH CHARACTER IS
705 REM   REPRESENTED BY THREE INTEGERS; THE FIRST OF THESE CONTAINS
710 REM   THE FIRST CHARACTER COLUMN IN BIT FORM (WITH MOST
715 REM   SIGNIFICANT BIT AT TOP OF COLUMN), AND THE SUBSEQUENT
720 REM   INTEGERS EACH CONTAIN TWO SUCH COLUMNS (WITH EVEN
725 REM   COLUMNS SHIFTED 7 PLACES LEFT, THEN ADDED TO ODD
730 REM   COLUMN REPRESENTATIONS).
735 REM   THE CHARACTERS ARE ORDERED ACCORDING TO THEIR POSITIONS
740 REM   WITHIN THE ASCII CODE TABLE, COMMENCING AT THE SPACE
745 REM   (OCTAL 32) AND CONTINUING THROUGH TO THE UPPER CASE 'Z'.
750 REM
755 DATA     0,    0,    0,      0,  125,    0,      0,12288,12288
760 DATA    20,16276,16276,     18, 5503, 5412,     99,12808, 6595
765 DATA     4, 5458, 5893,      0,   96,    0,      0, 3618, 8320
770 DATA     0, 8354, 3584,     42, 3592, 3626,      8, 1036, 1032
775 DATA     0,  134,    0,      0, 1032, 1024,      0,  128,    0
780 DATA     1,  776, 6208,     30, 4289, 8508,      0, 4351,  128
785 DATA    33, 8645, 9393,     34, 9417, 9398,     12, 2596,16260
790 DATA   114,10449,10446,     62, 9417, 9350,     67, 8776,10336
795 DATA    54, 9417, 9398,     48, 9417, 9406,      0, 2322,    0
800 DATA     0, 2323,    0,      0, 1044, 4352,      0, 2580, 2560
805 DATA     0, 4372, 1024,     32, 8261, 9264,      6, 5414, 4412
810 DATA    31, 4676, 4639,    127, 9417, 9398,     62, 8385, 8354
815 DATA    65,16321, 8382,    127, 9417, 9409,    127, 9288, 9280
820 DATA    62, 8385, 8871,    127, 1032, 1151,      0, 8447, 8320
825 DATA     2,  129,  254,    127, 1044, 4417,    127,  129,  129
830 DATA   127, 4120, 4223,    127, 4124,  383,     62, 8385, 8382
835 DATA   127, 9288, 9264,     30, 4293, 8509,    127, 9292, 9521
840 DATA    50, 9417, 9382,     64, 8319, 8256,    126,  129,  254
845 DATA   120,  769,  888,    126,  158,  254,     99, 2568, 2659
850 DATA    96, 2063, 2144,     67, 8905,10465
855 END
```

```
------------------
| SAMPLE OUTPUT  |
------------------

$   $  $$$$$  $      $       $$$     $
$   $  $      $      $      $   $    $
$   $  $      $      $      $   $    $
$$$$$  $$$$   $      $      $   $    $
$   $  $      $      $      $   $    $
$   $  $      $      $      $   $
$   $  $$$$$  $$$$$  $$$$$   $$$     $
```

SEA SEARCH

BY MARK WICKHAM

If you need a game for your micro the next time some friends come over, try Sea Search. Sea Search *provides a challenge and may even give your friends a chance to be heroes!*

The Game

In *Sea Search* you guide your ship (represented by an "S") through the sea and pick up survivors (repesented by an "X"). Think of a survivor as a person stranded on a raft. To pick him up, you must land on the space he occupies. You may pick up two or more survivors on one move as long as they are in a straight line with your ship. After each move, an updated map of the sea will be printed. The series of asterisks (*) shown represent an island which will look the same every game, although the starting position of your ship and the locations of the survivors will change from game to game. If you run into the island or the boundaries (marked with a period), you will automatically lose.

To win, you must remove all of the survivors from the board. At the beginning, you input the number of survivors you wish to have. Inputs may range from one to 150. An input of fewer than 10 will provide an easy and short game. A game involving a large number of survivors is more difficult and possibly more time consuming because more decisions will be involved.

Ship Movement

In moving your ship, you are asked two questions. The first involves direction. Your legal moves are shown below.

```
7 8 9
4 * 6
1 2 3
```

Imagine that you are in the middle of the chart (where the asterisk is). If you wish to go down, movement would be "2". A left movement would be "4", and an upward movement slanting to the right would be a "9". The other question asked is ship speed, that is, the amount of spaces you want your ship to move in a given direction. When typing in your ship speed, remember that you don't have to land on a survivor on the exact amount of spaces to rescue him, you merely have to touch his spot at one point. This makes possible rescue of several survivors on one move.

The Program

The program has several basic steps. The dimensions of the map are 20 X 20. Because one blank space is printed before and after each character in the map, the width of the map becomes 60 spaces. This feature makes the map larger for better view. If your terminal, or printer, has only a 40-character width, you can change line 1580 so that one space will be printed before each character and no spaces after, reducing the width to 40 and the time required to print the map.

Another option is the maximum number of survivors. This is set in line 1350. In my program 150 is the maximum, but as many as 270 can fit. However, after about 200 survivors are plotted, the computer will have a hard time finding open spaces.

The program is written in Altair 8K BASIC Rev. 4.0 The program should run in almost any type of BASIC with little or no conversion. Possible conversions are the use of string variables, the use of multiple statements per line and the CHR$ and ASC functions. String variables are used only to get the player's name and print it on the terminal when asking for his inputs. For BASICs without string variables, this can be modified by deleting line 700 and redoing lines 1710 and 1730. The use of multiple statements per line can be overcome by separating each line into two or more lines. Although these two unstandard commands can be easily modified, the use of the CHR$ and ASC functions are an important part of the program and cannot be converted very easily.

The program will use about 7 or 8K of memory (not including the additional memory needed for BASIC). However, it can be reduced to 4 or 5K of memory by deleting the REM statements and the instructions at the beginning.

If you own a CRT, use it because the program has a tendency to use a lot of paper. Also, if your CRT does not have automatic scrolling, insert a BASIC statement that will clear your terminal's screen at line 1505.

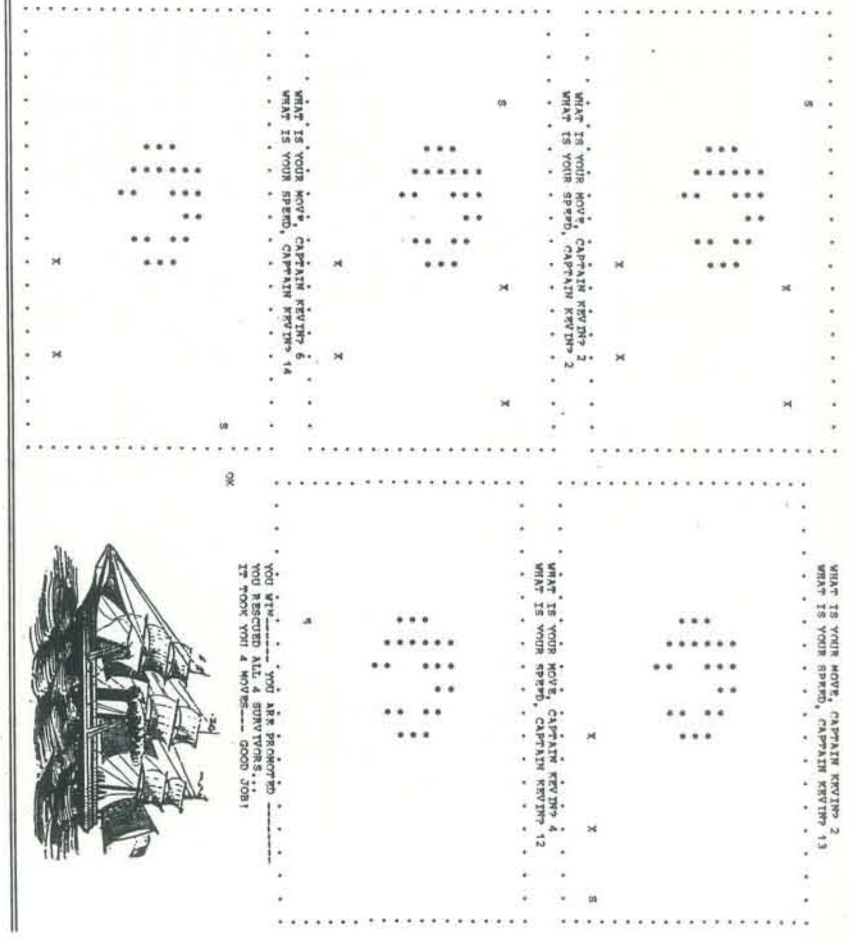
Treat your friends to a game of *Sea Search* and see who can rescue the survivors in the fewest moves!

```
LIST
400 REM ************************************************
410 REM *                                              *
420 REM *                SEA SEARCH                    *
430 REM *                                              *
440 REM * WRITTEN BY MARK WICKHAM- ALTAIR 8K REV.4.0*
450 REM *                1/5/80                        *
460 REM *                                              *
470 REM ************************************************
500 CLEAR
510 PRINT TAB(20)"SEA SEARCH"
520 PRINT:PRINT:PRINT
530 PRINT TAB(12)"IN THIS GAME, YOU MUST MANEUVER YOUR SHIP (S)"
540 PRINT TAB(12)"THROUGH AN IMAGINARY SEA.  IN THE SEA ARE ANY"
550 PRINT TAB(12)"NUMBER OF SURVIVORS (YOU CHOSE HOW MANY).  A"
560 PRINT TAB(12)"SURVIVOR IS A PERSON STRANDED IN THE SEA ON A"
570 PRINT TAB(12)"RAFT.  (THEY ARE SIGNIFIED BY AN X).  TO RESCUE"
580 PRINT TAB(12)"A SURVIVOR, YOU MUST LAND ON THE SPACE HE IS"
590 PRINT TAB(12)"ON.  THIS WILL REMOVE HIM FROM THE BOARD.  IF"
600 PRINT TAB(12)"YOU LAND ON THE ISLAND, MADE UP OF ASTERISKS (*)"
610 PRINT TAB(12)"OR THE OUTER BOUNDRIES (.) THEN YOU LOSE.  YOUR"
620 PRINT TAB(12)"MOVES ARE SHOWN BELOW"
630 PRINT
640 PRINT TAB(12)"7 8 9          TO WIN, YOU MUST RESCUE ALL THE"
650 PRINT TAB(12)"4   6          SURVIVORS . . . . . . . . . . . ."
660 PRINT TAB(12)"1 2 3          SPEED IS THE NO. OF SPACES YOU "
670 PRINT TAB(12)"               WANT TO GO IN A GIVEN DIRECTION"
680 PRINT:PRINT TAB(12)"GOOD LUCK ------------------- HERE WE GO ---"
690 PRINT:PRINT
700 PRINT TAB(12)"WHAT IS YOUR NAME";:INPUT N$
710 PRINT:PRINT
1000 DIM A(20,20)
1100 FOR I=1 TO 20
1110 FOR J=1 TO 20
1120 A(I,J)=0
1130 NEXT J
1140 NEXT I
1150 FOR X=7 TO 13
1160 FOR Y=7 TO 12
1170 READ A(X,Y)
1180 NEXT Y
1190 NEXT X
1300 FOR I=1 TO 20:A(1,I)=2:NEXT I
1310 FOR J=1 TO 19:A(J,1)=2:A(J,20)=2:NEXT J
1320 FOR I=1 TO 20:A(20,I)=2:NEXT I
1330 X1=INT(20*RND(1)+1):Y1=INT(20*RND(1)+1):IF A(X1,Y1)<>0 THEN 1330
1340 A(X1,Y1)=3
1350 PRINT TAB(12)"HOW MANY SURVIVORS";:INPUT D3:IF D3>150 THEN 1350
1360 IF D3<1 THEN 1350
1365 FOR Q=1 TO D3
1370 X2=INT(20*RND(1)+1):Y2=INT(20*RND(1)+1):IF A(X2,Y2)<>0 THEN 1370
1380 A(X2,Y2)=4
1390 NEXT Q
1500 REM PRINT MAP
1510 FOR I=1 TO 20
1520 FOR J=1 TO 20
1530 IF A(I,J)=0 THEN A(I,J)=ASC(" ")
1540 IF A(I,J)=1 THEN A(I,J)=ASC("*")
1550 IF A(I,J)=2 THEN A(I,J)=ASC(".")
1560 IF A(I,J)=3 THEN A(I,J)=ASC("S")
1570 IF A(I,J)=4 THEN A(I,J)=ASC("X")
1580 PRINT" ";:PRINT CHR$(A(I,J));:PRINT" ";
1590 NEXT J:PRINT
1600 NEXT I
1610 D2=D3-S2:IF D2=0 THEN 3500
1700 REM MOVE
1710 PRINT TAB(12)"WHAT IS YOUR MOVE, CAPTAIN ";N$;:INPUT M:T=T+1
1720 IF M<1 OR M>9 THEN PRINT"INVALID MOVE---":IF M<1 OR M>9 THEN 1710
1730 PRINT TAB(12)"WHAT IS YOUR SPEED, CAPTAIN ";N$;:INPUT S
1740 IF S<1 THEN PRINT"INVALID SPEED---":IF S<1 THEN 1730
1750 FOR I=1 TO 20
1760 FOR J=1 TO 20:IF A(I,J)=83 THEN 1790
1770 NEXT J:NEXT I
1790 I1=I:J1=J:ON M GOTO 1800,1850,1900,1950,2000,2050,2100,2150,2200
1800 V=0:REM MOVE DIR. 1
1810 V=V+1
1820 I1=I1+1:J1=J1-1:GOSUB 4000
1830 IF V<>S THEN 1810
1840 A(I1,J1)=83:GOTO 1500
1850 V=0:REM MOVE DIR. 2
1860 V=V+1
1870 I1=I1+1:J1=J1:GOSUB 4000
1880 IF V<>S THEN 1860
1890 A(I1,J1)=83:GOTO 1500
1900 V=0:REM MOVE DIR. 3
1910 V=V+1
1920 I1=I1+1:J1=J1+1:GOSUB 4000
1930 IF V<>S THEN 1910
1940 A(I1,J1)=83:GOTO 1500
1950 V=0:REM MOVE DIR. 4
1960 V=V+1
1970 I1=I1:J1=J1-1:GOSUB 4000
1980 IF V<>S THEN 1960
1990 A(I1,J1)=83:GOTO 1500
2000 V=0:REM MOVE DIR. 5
2010 PRINT"NOT MUCH SINCE IN STANDING STILL, CAPTAIN!!!"
2020 GOTO 1700
2050 V=0:REM MOVE DIR. 6
2060 V=V+1
2070 I1=I1:J1=J1+1:GOSUB 4000
2080 IF V<>S THEN 2060
2090 A(I1,J1)=83:GOTO 1500
2100 V=0:REM MOVE DIR. 7
2110 V=V+1
2120 I1=I1-1:J1=J1-1:GOSUB 4000
2130 IF V<>S THEN 2110
2140 A(I1,J1)=83:GOTO 1500
2150 V=0:REM MOVE DIR. 8
2160 V=V+1
2170 I1=I1-1:J1=J1:GOSUB 4000
2180 IF V<>S THEN 2160
2190 A(I1,J1)=83:GOTO 1500
2200 V=0:REM MOVE DIR. 9
2210 V=V+1
2220 I1=I1-1:J1=J1+1:GOSUB 4000
2230 IF V<>S THEN 2210
2240 A(I1,J1)=83:GOTO 1500
3000 PRINT TAB(12)"SORRY CHUM, YOU LOSE. (SHIPS ONLY GO IN WATER)"
3020 PRINT TAB(12)"YOU CAPTURED";S2;"SURVIVORS ALTOGETHER!!"
3030 PRINT TAB(12)"YOU HAD USED";T;"MOVES--- TOO BAD!":END
3040 END
3500 PRINT TAB(12)"YOU WIN------ YOU ARE PROMOTED --------"
3510 PRINT TAB(12)"YOU RESCUED ALL";D3;"SURVIVORS..."
3520 PRINT TAB(12)"IT TOOK YOU";T;"MOVES--- GOOD JOB!":END
4000 IF A(I1,J1)=ASC("X") THEN S2=S2+1
4001 IF A(I1,J1)=ASC("X") THEN A(I1,J1)=ASC(" ")
4010 IF A(I1,J1)=ASC("*") OR A(I1,J1)=ASC(".") THEN 3000
4020 A(I,J)=32
4030 RETURN
9000 DATA 0,1,1,1,0,0,0,1,1,1,1,0,1,1,1,0,1,1,1,1,0,0,0,1
9010 DATA 1,1,0,0,1,1,0,1,1,0,1,0,0,0,1,0,0,0
OK
```

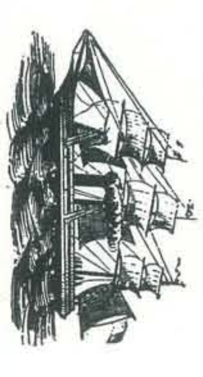
SEA SEARCH

IN THIS GAME, YOU MUST MANEUVER YOUR SHIP (S) THROUGH AN IMAGINARY SEA. IN THE SEA ARE ANY NUMBER OF SURVIVORS (YOU CHOOSE HOW MANY). A SURVIVOR IS A PERSON STRANDED IN THE SEA ON A RAFT. (THEY ARE SIGNIFIED BY AN X). TO RESCUE A SURVIVOR, YOU MUST LAND ON THE SPACE HE IS ON. THIS WILL REMOVE HIM FROM THE BOARD. IF YOU LAND ON THE ISLAND MADE UP OF ASTERISKS (*) OR THE OUTER BOUNDARIES (.), YOU LOSE. YOUR MOVES ARE SHOWN BELOW.

```
7 8 9    TO WIN, YOU MUST RESCUE
4   6    ALL THE SURVIVORS . . . . . . . .
1 2 3    SPEED IS THE NO. OF SPACES
         YOU WANT TO GO IN A GIVEN
         DIRECTION
```



```
WHAT IS YOUR NAME? KEVIN

HOW MANY SURVIVORS? 4

WHAT IS YOUR MOVE, CAPTAIN KEVIN? 2
WHAT IS YOUR SPEED, CAPTAIN KEVIN? 2

WHAT IS YOUR MOVE, CAPTAIN KEVIN? 6
WHAT IS YOUR SPEED, CAPTAIN KEVIN? 14

WHAT IS YOUR MOVE, CAPTAIN KEVIN? 2
WHAT IS YOUR SPEED, CAPTAIN KEVIN? 13

WHAT IS YOUR MOVE, CAPTAIN KEVIN? 4
WHAT IS YOUR SPEED, CAPTAIN KEVIN? 12

YOU WIN------ YOU ARE PROMOTED --------
YOU RESCUED ALL 4 SURVIVORS...
IT TOOK YOU 4 MOVES--- GOOD JOB!
OK
```



APPLE ANIMATION

BY JIM DAY

Many Apple owners have expressed an interest in creating games that involve animated graphics, but don't quite know how to begin. Actually it isn't all that difficult. So let's take a quick look at what's involved in programming a simple game using animated low-resolution graphics.

First we have to decide what we want to show on the screen. For good animation it should be something fairly small. Large, complex displays take too long to draw, causing an unacceptably low flicker rate. We will use a highly simplified racecar as our moving object. The shape of this object is shown in Figure 1. OK, so how do we draw it?

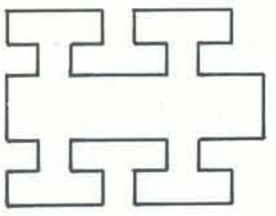


Figure 1. Racecar

Since we want it to move, we will use a variable bias for all X and Y coordinates. The X bias will be named X0 and the Y bias will be named Y0. All graphic elements of the car will be specified relative to screen location X0, Y0. Subroutine 2000 draws the car:

```
2000  REM DRAW-CAR SUBROUTINE
2015  REM DRAW WHEELS
2020  COLOR = 0: REM BLACK
2025  VLIN Y0 + 1, Y0 + 7 AT X0
2030  VLIN Y0 + 1, Y0 + 7 AT X0 + 5
2035  COLOR = 15: REM WHITE
2040  HLIN X0, X0 + 5 AT Y0 + 4
2045  REM DRAW AXLES
2050  COLOR = 6: REM GRAY
2055  HLIN X0 + 1, X0 + 4 AT Y0 + 2
2060  HLIN X0 + 1, X0 + 4 AT Y0 + 6
2065  REM DRAW BODY
2070  COLOR = 1: REM MAGENTA
2075  VLIN Y0, Y0 + 7 AT X0 + 2
2080  VLIN Y0, Y0 + 7 AT X0 + 3
2085  RETURN
```

To position the car on the screen, we must initialize X0 and Y0:

```
100   X0 = 17: Y0 = 16
```

This places the car in the middle of the screen. To show the wheels as black, we will make the background white:

```
105   GR: COLOR = 15
110   FOR Y = 0 TO 39
115   HLIN 0, 39 AT Y
120   NEXT Y
```

The GR command turns on the low-resolution graphics mode and clears the screen, and the FOR-NEXT LOOP paints the entire screen white.

We call subroutine 2000 to draw the car:

```
145   GOSUB 2000: REM DRAW CAR
```

Line 999 separates the main program from the following subroutine:

```
999   END
```

OK, so we can draw the car, but how do we make it move? We use game paddle 0 to set the horizontal position of the car, scaling by 5 for better control:

```
165   X0 = PDL(0)/5
```

But wait a minute. Now we can move the car to a new position, but the old image is still on the screen. We need another subroutine to erase it by painting it white:

```
5000  REM ERASE-CAR SUBROUTINE
5005  COLOR = 15
5010  FOR X = X0 TO X0 + 5
5010  VLIN Y0, Y0 + 7 at X
5020  NEXT X
5025  RETURN
```

We insert a call to erase the old image before drawing a new one:

```
160   GOSUB 5000: REM ERASE CAR
170   GOTO 125: REM DRAW IT AGAIN
```

Well, now we can move the car from side to side. In fact we can move it right off the screen, which draws complaints from BASIC. To eliminate this, we add a couple of tests:

Continued on pg. 49

DIVERSIONS

Games to Program: DOZO

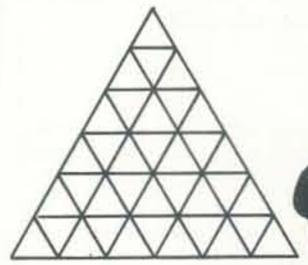
©1979 by Herbert Kohl

BY HERBERT KOHL

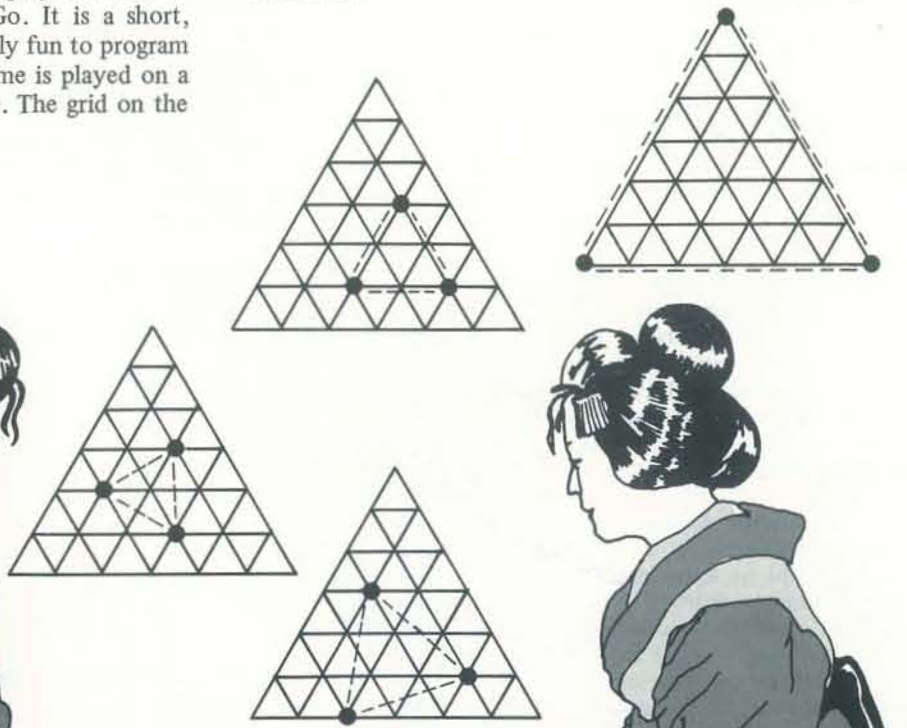
In this game, one of an ongoing series of articles, the player who recognizes patterns quickly has the advantage. Herb invites you to accept the programming challenge, then send in your programs of these game ideas.

Herb and his wife Judith are co-directors of Coastal Ridge Research and Educational Center in Pt. Arena, CA. The center is a non-profit educational organization designed to provide both theoretical and practical support to people involved in progressive educational activities. – *TD*

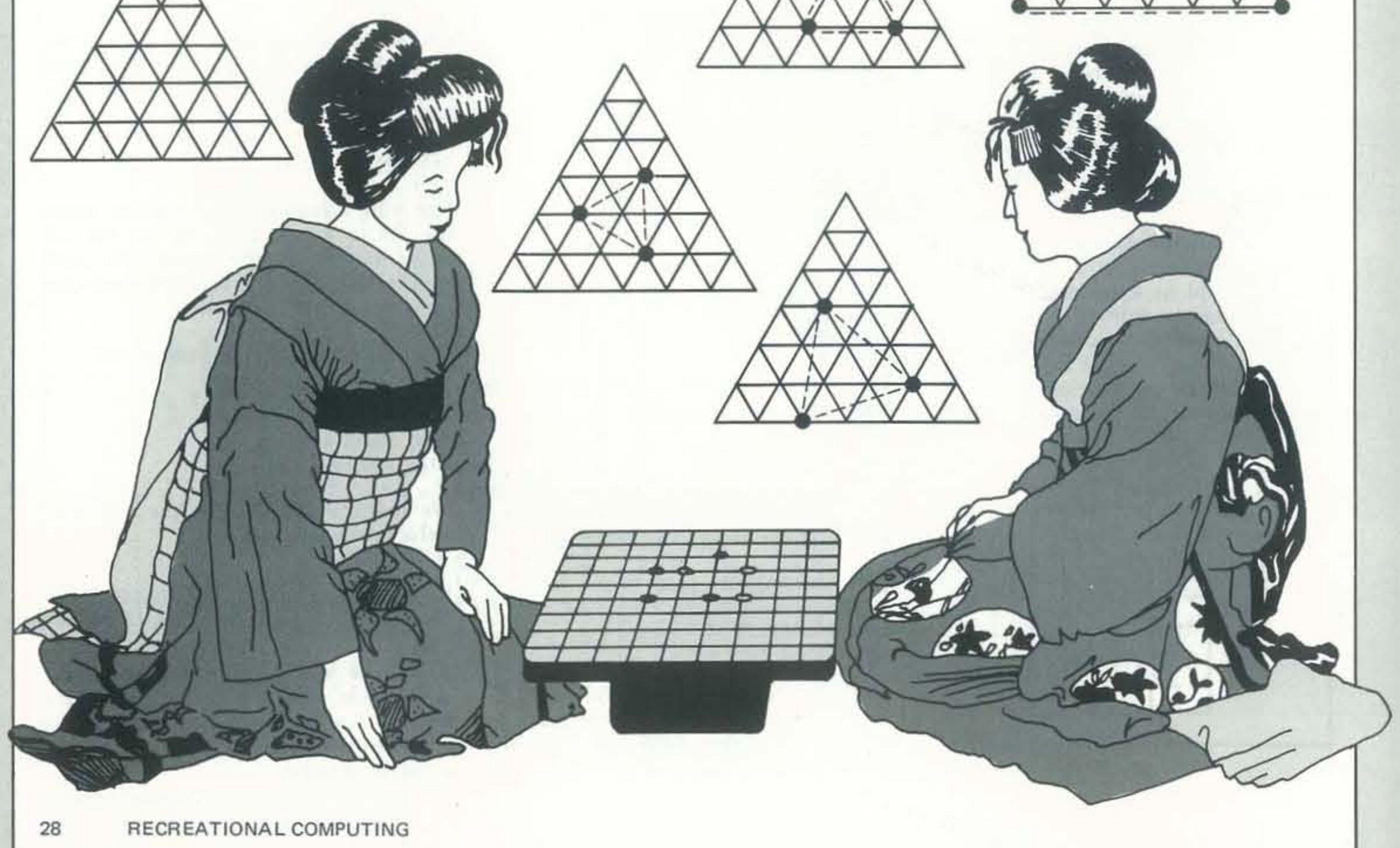
Dozo is a game that Japanese Go masters play in order to warm up for serious extended games of Go. It is a short, elegant game, and one that might be especially fun to program for a computer with a color screen. The game is played on a board in the shape of an equilateral triangle. The grid on the board looks like this:



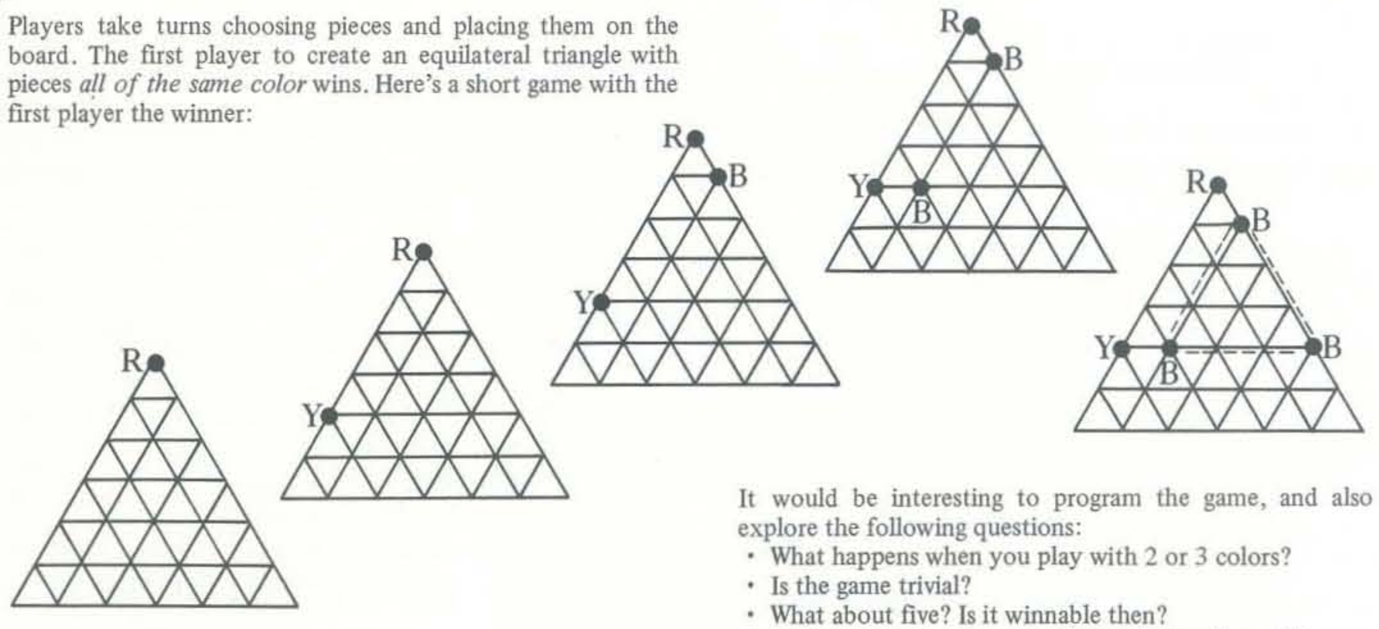
As in the case of Go, the game is played on the intersections of the lines and not in the squares or triangles. The object of the game is to make an equilateral triangle—any equilateral triangle—on the board. Here are some possible triangles. The first two are obvious, but the third and fourth are more subtle.



The game is for two players. There are pieces of four different colors; red, blue, white and yellow. All of the pieces belong to both players. One of the central ideas in Dozo is that players have to become accustomed to playing for the pattern that they are trying to create and not for the pieces they command.



Players take turns choosing pieces and placing them on the board. The first player to create an equilateral triangle with pieces *all of the same color* wins. Here's a short game with the first player the winner:



It takes a little time and practice to think in terms of Dozo. Strategically more time is spent preventing your opponent from winning than in aggressively pursuing a victory. The game depends upon the quick perception of patterns, and this is probably why it is used as a warm-up for Go masters.

It would be interesting to program the game, and also to explore the following questions:

- What happens when you play with 2 or 3 colors?
- Is the game trivial?
- What about five? Is it winnable then?
- Is there a best way to go about playing that will insure the first (or second) player a win?
- What happens if you enlarge the board? Or reduce it when you play with 3 colors?

Answers to any of these questions plus a copy of a program for the basic game would be appreciated.

TRS-80 : THE OLD SHELL GAME

The Machine is Faster than the Eye

BY MILAN CHEPKO

Step right up, ladeez 'n gentlemen, the computer is quicker than the eye. . . but not quicker than Milan Chepko's programming abilities. Milan, our most frequent contributor, presents you with a small program. . . for your enjoyment. Step right up! – *RZ*

This program is a new version of an old "con game." The goal is still to guess which of three cups covers a pea. But instead of a skilled con man to conceal the pea, the computer is used to place the pea randomly under one of the cups. As an added twist, the program allows us to watch the pea being moved from place to place. The computer moves the "P" around too fast to be seen, so the loop at line 420 slows things down a little.

```
100 CLS:PRINT"    *** THE OLD SHELL GAME ***
110 ' BY MILAN D  CHEPKO     THIEF RIVER FALLS, MN   56701
120 RANDOM:DEFINT A-Z:DIM A(30):'*** DELETE FOR LEVEL I ***
130 FORI=1TO5:A(I)=200:NEXTI
140 PRINT:INPUT"HOW MANY PLAYERS (MAX=5) ";P:IFP>5GOTO140
150 CLS:PRINT"EACH PLAYER STARTS WITH $200 AND TRIES TO FIND"
160 PRINT"THE PEA HIDDEN UNDER ONE OF THESE SHELLS ..":PRINT
170 GOSUB190:GOTO240
180 '   PRINT THE DISPLAY
190 PRINT"        *          *          *"
200 PRINT"      *   *      *   *      *   *"
210 PRINT"     * 1 *      * 2 *      * 3 *"
220 PRINT"     *     *    *     *    *     *":PRINT:RETURN
230 '   INPUT BET FOR EACH PLAYER WITH MONEY
240 FORI=1 TO P:A(20+I)=0:A(10+I)=0
250 IFA(I)<=0GOTO310
260 PRINT"PLAYER #";I;"  WHICH SHELL";:INPUTA(20+I)
270 IF A(20+I)>3 GOTO260
280 PRINT"YOU HAVE $";A(I);"...WHAT IS YOUR BET ";
290 INPUTA(10+I):IFA(10+I)>A(I) PRINT"GOOD LUCK!!"
300 FORJ=1TO200:NEXTJ
310 PRINT:NEXTI
320 '   PRINT DISPLAY AGAIN
330 CLS:GOSUB190:PRINT
340 '   PRINT SUMMARY OF BETS
350 PRINT"PLAYER","SHELL","WAGER"
360 FORI=1TOP:PRINTI,A(20+I),"$";A(10+I):NEXTI
370 '   MOVE P AT RANDOM 20 TIMES
380 FORI=1TO20:PRINT@200," ";:PRINT@212," ";:PRINT@224," ";
390 R=RND(3):IFR=1 PRINT@200,"P";:GOTO420
400 IFR=2 PRINT@212,"P";:GOTO420
410 PRINT@224,"P";
420 FORJ=1TO100:NEXTJ
430 NEXTI
440 '   PRINT OUT THE WINS AND LOSSES
450 PRINT@384,:FORI=1TO6:PRINT:NEXTI
460 PRINT@320,:FORI=1TO P
470 PRINT"PLAYER #";I;:IFA(20+I)=R GOTO500
480 A(I)=A(I)-A(10+I):IFA(I)<=0PRINT"IS DISQUALIFIED!":GOTO520
490 PRINT"..YOU LOST, AND NOW HAVE $";A(I):GOTO520
500 A(I)=A(I)+A(10+I)
510 PRINT"..YOU WON, AND NOW HAVE $";A(I)
520 PRINT:NEXTI
530 INPUT"HIT 'ENTER' FOR ANOTHER GAME";A$:CLS:GOTO170
```

CRYPTARITHMS

BY JACK CREHORE

Send your solutions to Jack Crehore, P.O. Box 96, Charlotte Court House, Virginia 23923

Devoted Fans! Your eager calling for harder puzzles puts me in a quandary; I mustn't frustrate the less skilled readers who also enjoy the challenge that cryptic arithmetic offers. I wish I could quote more extensively from your letters but I must reserve space for tabulations and for a little coaching of our Novices so they may share more in the fun!

Our first fan letter today is from surely a Genius. S. R. McENTEE: your letter is a gem of content and arrangement. Puzzle 7 didn't trip you.

CHARLES: So! We've caught another big fan—a computer user who solves without using his PET. A workmanlike outlay, Charles, your two pages! I would like to have seen your computer program. I forswore word keys years ago: they demean math to word-guessing.

By the way, to myriad solicitous proposers that I use puzzles furnished by others: we would risk lawsuits by copyright owners despite our utmost care. Every cryptarithm I have ever submitted for publication, I have devised *entirely* by myself, or as the responsible party. When I deviate I'll notify my fans. They send me many puzzles of great interest.

To start afresh with our Novices in Cryptarithm solving, and for newcomers, I repeat the Hints given to date. You may obtain the original treatment of any Hint cited, by sending to me a stamped, addressed envelope with 50¢ in *money* for xerox or other copies of one or two pages of *RC*; price for additional pages in same order, 20¢ per page, 3 pages for 50¢. This is a test, farsighted, with odd possibilities. If you favor the scheme, commence, or commend it.

Most Hints here below appeared in *RC* 42 Nov.-Dec. 79. If you will keep in mind all these minor axioms, you will soon become an Adept, someday a Genius!

Now: Consider each letter (digit) for its counting value in the position it occupies in a problem: small value (01234) or large (56789).

Try to determine whether a digit is even or odd: (02468) or (13579).

Try to discover whether or not there is any "carry" from an operation wherein two or more digits are added, or multiplied, together. Keep aware that there are only three pairs of digits that will multiply together and give a Product all in one letter: 2 * 3; 2 * 4; 3 * 3.

If two digits multiply each other to a two column Product the tens column of the Product will be either 1 less, or 2 less, than the smaller Multiplier: *8 * 9 = 72; 4 * 6 = 24.*

The addition of only 2 digits can never bring a carry of more than 1 to the tens column; 3 digits can bring 1, or 2, but never 3; three 9s add to only *27*.

I know of no greater aid in fast solving than an Is-not Chart, an Elimination Table. *RC* 42 Nov./Dec. '79 p. 23, shows a chart of 11 by 11 squares, with digits 0 to 9 along the top row, and the ten letters of a Puzzle listed in ten squares down the left edge.

Cross out a square immediately when you determine that no letter will fit its value. In this way you keep reducing the range of letter values you need to study.

Novice! Adept! Genius! Computer-er! Come to Class every *RC* issue! *Your* score will promote the Lowly, and exalt the High! Write me your vote for the listing of solution Hints! Get credit for unique Hints that you contribute. Post Graduates, Ph Ds, don't laugh at little fellows! Help 'em up! Share your higher learning! The good teacher learns more than his pupil.

Jack Crehore-NINE HEX

PUZZLE 21 (NOVICE)

```
  Y, B J R, F A T, J U U
-    J C B, T K Y, K R C
  ----------------------
     T Y R, B C B, K B J
```

PUZZLE 22 (ADEPT)

```
   Y A S F
 + F M A F
 ---------
 Y R S T T
```

PUZZLE 23 (GENIUS)

$$\frac{\left(\sqrt[M]{MBM} + ARR\right)^{\frac{K-A}{F+R}} = \left(A + ARR\right)^{\frac{K-A}{F+R}} = ARK^{\frac{K-A}{F+R}} = ARK^{\frac{R}{M}} = ARK}{\left(\sqrt[M]{ARK}\right)^{R} = K^{R} = NF}$$

Simplification step-by-step of an expression in Cube Root.

PUZZLE 24 (COMPUTER)

```
B C W T M J F R N
B C W T M J F R N
B C W T M J F R N
B C W T M J F R N
-----------------
C C B T B T C T C
```

Simple addition for 3rd Grade Kids with Computers.

Continued on pg. 37

ADVERTISING SPACE AVAILABLE

¼,½, Full Page Inside Front & Back Covers

PLEASE SEND FOR RATE CARD

Advertising Manager
People's Computer Company
1263 El Camino Real, Box E
Menlo Park, CA 94025
(415) 323-3111

Five New Personal Computers

BY DENNIS ALLISON

The range of possibilities in personal computers grows with each passing month. Here are five newly announced machines, each with interesting and unique capabilities. All five can honestly claim to be "personal computers," yet they differ in price and performance by factors of 20 or more. One machine is not even user programmable (yet). Each will find its nitch in the market and will contribute to the search for a truly adequate personal system. – DA

Mattel Electronics' new INTELLIVISION Intelligent Television home computer center forms a complete system for family entertainment, education, personal improvement and financial management.

INTELLIVISION

Mattel Electronics' INTELLIVISION is a video game turned home information center. There are two parts to the intellivision system—the Master Component and the Keyboard Component. The Master Component is programmable by ROM cartridge and functions as a video game. A variety of different cartridges are available, including Soccer (licensed by the North American Soccer League), Golf (licensed by the Professional Golf Association) and Skiing (licensed by the U.S. Ski Team). Twenty different cartridges are available for the Master Component.

The Master Component is built around a General Instrument 16-bit microprocessor and can provide a variety of sound effects, three-part harmony, and good color graphics. Input to the Master Component is via hand-held keypads and control paddles.

The Master Component may be expanded with a Keyboard Component which uses preprogrammed digital cassettes. It features a 60-key typewriter-like keyboard and a digital cassette system with computer controlled fast-forward and tape search. The Keyboard Component accepts preprogrammed cassettes capable of handling digital and audio outputs as well as typed and audio inputs (from a microphone).

Preprogrammed cassettes available this year include: J. K. Lasser's 1980 Federal Income Tax Preparation, Stock Analysis, Jack LaLanne's Physical Conditioning; Jeanne Dixon Astrology, Dr. Art Ulene Weight Loss Program, Guitar Lessons and Music Composition, Speed Reading, and Conversational French. No programming is required.

In Mattel's view, apparently, the essence of home and personal computing is access to Central Data Banks with the kind of information consumers need and want for everyday living—from airline schedules to stock exchange information. Future components of this system will allow telecommunications access to data bases, hard-copy and speech output, and, eventually, programmability.

Pricing for the Master Component is under $300; for the Keyboard Component, up to $500. Cartridges cost about $30. and cassettes under $40, in most cases. For more information contact Mattel Electronics, 5150 Rosecrans Avenue, Hawthorne, CA 90250. (213) 644-0411.

THE SHARP PC-1210 AND PC-1211

This machine looks like an overgrown credit-card calculator, but it has a full 55-key keyboard, a 20-character LCD display, and a "beeper" for audio output. It is programmable in a BASIC with nearly every feature you could want. The only thing I missed was a random number generator; that makes simple games hard to construct. There is a cassette interface as well; so programs can be stored on a conventional recorder. The technology is CMOS; the program and data are maintained even when the personal computer is turned off. It's slow, but who cares. The human factors are particularly well thought-out. There is an integral line-editor which makes program entry and modification easy. Soft-keys are available to facilitate the programming of complex interactive systems. It looks like a reasonable machine for the intermediate calculations too large and too complex to program by keystrokes on a programmable calculator but not requiring a large machine.

Suzanne Rodriguez, *Dr. Dobb's Journal* editor, using the Sharp PC-1210 to compute page reductions.

For the PC-1210, program storage is 400 steps of program and 50 memory cells. For the PC-1211, there can be 1424 steps of program and 178 memory cells. A 400-step program is fairly long. The BASIC statement

```
10 X=Y
```

leaves 394 steps and 49 memories.

The price is $125 for the PC-1210 and $180 for the PC-1211. The computer is currently available only in Japan; its introduction into the US is expected sometime next year.

SINCLAIR RESEARCH ZX80

The Sinclair ZX80 is a new entrant into the personal computer market from Britain. Developed by Clive Sinclair, the inventor of the world's first pocket TV and pocket calculator, the ZX80 is among the smallest and least expensive of the personal computers. It measures a mere 9 x 7 x 2 inches and weighs only 12 oz. The cost in Britain (it's not yet available in the US) is £99.95 assembled, £77.95 in kit form; that translates to about $220 US assembled, $176 in kit. The power supply is a separate module available for about $20 extra. One may expect modems at $60 and floppy disks at $300.

The computer is fairly traditional, in the style of the PET, Apple and Atari. The CPU is a Nippon Electric (NEC) Z80. The keyboard is of the touch-sensitive type with a standard typewriter configuration. The keyboard is not full-size, a matter which will cause substantial aggravation. Video output is organized into 24 lines of 32 characters each. In addition to the alphabetic character set, there is a set of 24 high-resolution graphics characters. The standard display is white on black with reverse-video possible for any individual character.

The display is the usual interface to a TV. However, in the interests of cutting costs, the microprocessor also serves as the character generator causing the screen to flicker when a key is depressed. A cassette interface is included which allows programs to be stored on a conventional audio cassette. A printer interface and an inexpensive modem for telecommunications access are still to come. The normal machine comes with 1K RAM for program and data storage; this is expandable with an optional memory expansion board to a total of 4K bytes.

The BASIC interpreter, the operating system, character generator, and monitor are all packed into a single 32K read-only memory. The BASIC appears to be a powerful one with a number of unusual features. In particular, language keywords are entered with a single keystroke since the machine accepts only syntactically correct statements. Each line is syntax checked as it is entered, and it is incorporated into the program only when it parses correctly. Internally the BASIC is stored in an encoded form which Sinclair claims gives them a 4:1 space advantage over storing the text directly.

Sinclair Research's ZX80 is a low-cost BASIC-language personal computer.

The BASIC itself supports strings as a data type (in the fashion of Microsoft BASIC) and integer variables of any length.

We've not yet seen this machine. (This description is made up from press releases.) We're looking forward to seeing the machine itself and to browsing through the 130-page BASIC manual which comes with it.

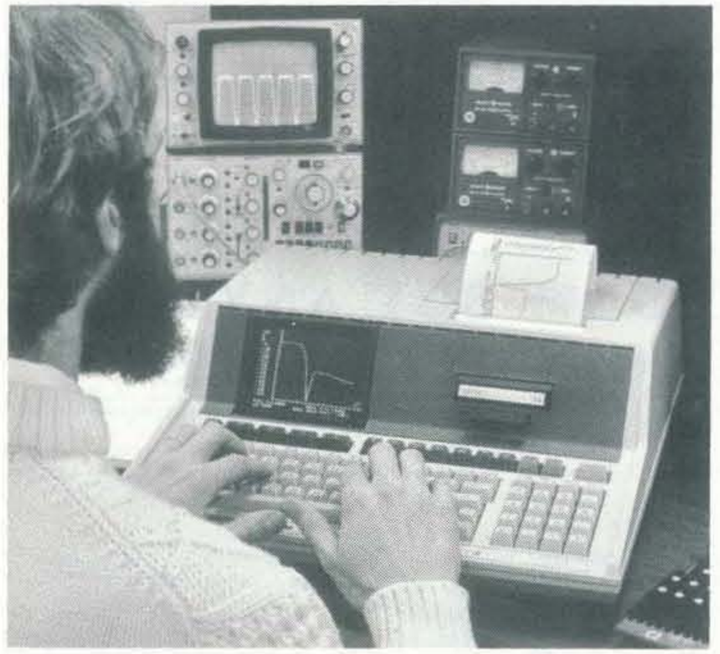
For more information contact Sinclair Research Ltd., 6 King's Parade, Cambridge CB2 1SN, England.

THE HEWLETT-PACKARD HP-85

The HP-85 is a very nice stand-alone BASIC-language computer system. HP calls it "a personal computer for Professionals." Its price is $3250 for the basic unit, including typewriter-like standard size keyboard, CRT display with graphics capability, integral printer with graphics capability, tape cartridge and 16K bytes of memory. This makes it truly competitive in the market. An optional additional 16K bytes of memory is available for $395. The packaging is traditional HP quality, small in size (16 x 18 x 20 inches) and light in weight (under 20 pounds).

The BASIC language is a superset of the ANSI standard with many desirable extensions. An important feature is the built-in interactive graphics available both on the screen and on the printer too. The keyboard provides a numeric keypad, soft-keys available to the programmer, and display/edit/system-control keys which permit direct user operation of the machine. The 5-inch CRT display can display up to 16 lines of 32 characters. There is a 64-line scroll memory which remembers up to 64 lines of data which can be viewed by rolling the display up or down. In graphics mode, the screen is an array of 256 X 192 points to allow for very high resolution graphics. The graphics display has its own frame memory so the user can switch freely back and forth from the graphical data display to alphabetic display and vice versa. The integral printer prints two 32-character lines per second. The character set supported is the full 128 character ASCII set with the added feature that all characters can be underlined. Graphic displays can also be output on the printer. When plotting, the display is rotated by 90° to allow for the generation of strip charts. The integral tape drive uses HP data cartridges which have a capacity of over 200,000 bytes. Read/write speed is 10ips with a search speed of 60ips. A directory and named file structure for tape-files is standard.

All in all this is a nice system even though a bit on the expensive side. See the June issue of *Dr. Dobb's Journal* for a review of the machine. For more information contact Inquiries Manager, Hewlett-Packard Company, 1507 Page Mill Road, Palo Alto, CA 94304, (415) 856-1501.



The HP-85 is a convenient integrated system designed for use by professionals.

THE PERQ MACHINE

The PERQ was inspired by the Xerox PARC Alto Computer. The internals are different but the idea is the same. It's an attempt to place a substantial amount of computing power into the hands of the user. It's much, much more powerful than machines like the TRS-80, PET, or Apple. Each PERQ has a quarter megabyte of memory and twelve megabytes of disk storage. The instruction execution speed is about one million per second, several times faster than any of the usual personal computers. In addition, the PERQ is presumed to be part of a group of machines sharing common peripherals; a co-axial network system with a transfer rate of about ten million bits per second allows data to be interchanged rapidly.

An 8-½ x 11 bit-mapped screen with 768 x 1024 point resolution refreshed at 60 Hz provides a good-looking graphic and text display. The resolution is adequate to allow display of a single page with near typewriter resolution.

A touch tablet allows one to use one's finger to enter graphical information. The position of the finger on the table is mirrored by a cursor on the screen. One can select neatly from a menu, draw pictures, indicate text to an editor, or whatever.

The processor itself executes P-code, the PASCAL intermediate form. The machine is, of course, intended to be programmed entirely in PASCAL. A 32-bit segmented virtual address space allows efficient execution of large programs. In addition there is a speech output system to allow audible cues.

The processor with 256K memory, disk, display, keyboard, pointer, speech output, RS-232 and IEEE-488 bus interfaces costs $20,000 and up. A fullblown system will run about $32,000. For more information contact Three Rivers Computer, 160 North Craig Street, Pittsburgh, PA 15213.

The PERQ machine provides substantial mass storage, high speed processor, a PASCAL programming environment, and a high resolution bit-mapped display.

COMMENTARIES

WHY I DIDN'T BUY ANYTHING AT THE COMPUTER FAIRE

BY DAVE GOMBERG

I consider myself a computer hobbyist as well as a computer professional. I have watched the hobby computer field develop for several years and hoped that this time I would find something to coax the bucks out of my pocket. I wanted substantial disk (say 10+ Mb), some dismountable medium of exchange (say floppies), and a backup for my big disk to tape or whatever. I expected to provide my own keyboard/display and printer.

HARDWARE

Part of what I do for a living is to configure medium-sized IBM systems. If I wanted to support 100 such users, I would buy:

IBM	4331 (1Mb CPU)	100,000
	3370A+B (1100+ Mb disks)	60,000
	8809 tape drive	15,000
	3278 printer	8,000
	Total	$ 183,000

If, in addition, I wanted to support a lot of remote usage (dial-up), I would add:

IBM 3704 communications controller	30,000
Prentice modems	10,000
Pacific Telephone lines (capitalized)	10,000
Grand total	$ 233,000

The first configuration would support 16 local and 8 dial-up users concurrently; the second would support 16 local and 32 dial-up users. The practical limit of this machine is between 20 and 40 concurrent users. Thus the hardware from IBM would cost about $2400 each for 100 users. This is major pricing breakthrough for IBM gear and represents a substantial decrease over what such a configuration would have cost three years ago.

SOFTWARE

For an operating system I would have chosen VM/370-CMS. MUSIC from McGill University would have been a close runner-up. VM/370 costs about $175 per month. For language processors, I would choose those from the University of Waterloo; FORTRAN, COBOL, BASIC, and PASCAL cost between $50-100 per month each. Statistical packages would include SPSS, SCSS, and SAS at about $100 per month each. I would probably have added SPEAKEASY at $150 per month as well, as it is a very productive language. My software bill would then have been less than $10 per user per month, or, say, $500 capitalized.

THE PACKAGE

For about $2900, I could buy a very nice setup from IBM (1/100thth share) —so I set out to the FIFTH FAIRE looking for a competitive micro system.

MICROS

You can guess the result. A system like this, ignoring software deficiencies, costs between $6-10K bought as a microcomputer. The main stumbling block seems to be the still very high unit cost of mass storage (disks). Even at OEM prices, $3-4000 seems to be the rule. The backup situation is still in its infancy; we can probably hope for progress there. But the fact is that until Winchester drives (or some functional equivalent) drop in end-user price to about $1500 with backup capability, it is going to be very tough to beat IBM's price/performance. If IBM improves similarly in that time, the micros may never catch up.

This is not to say that micros are not fun, or even that they don't have valid applications. But I do claim that as personal, general-purpose computers they are very expensive. They generally do provide better price/performance in the area of CPU cycles, but when I configure a micro, the CPU costs much less than each of the floppy drive, hard disk drive, and hard disk backup.

INTANGIBLES

When comparing micros and medium-priced IBM computers, it is difficult to put a dollar value on certain benefits each has. The micro does not suffer from contention by other users, transfers to its display at a much higher data rate than would usually be used in dial-up operation, and has predictable program execution times. On the other hand, the volume of software available for IBM 360/370 systems is prodigious, the value of natural data-sharing with similar users is difficult to measure, and the convenience and savings of buying and maintaining a single copy of software is hard to quantify. These must still be counted as intangible IBM benefits.

CONCLUSIONS

I'll be back at the FAIRE next year, still looking, but it will be two or thee years at least before I can be lured away from the Immense Blue Mother.

JAPAN MICRO-COMPUTER CLUB

The Japan Microcomputer Club, established in 1976, is the largest microcomputer club in Japan. Its membership of about 3,000 covers an age range of 5 to 74 years. Students and engineers are prominently represented. Reports show that hobbies, games, and business are the primary reasons among club members for interest in microcomputers.

The Club's journal, *Micon Circular*, is published eleven times a year. For the first four years there was no English version; however, beginning in 1980 there will be an English summary of each issue. If you want to obtain a summary of *Micon Circular*, contact:

Japan Microcomputer Club
c/o Japan Electronic Industry
Development Association
3-5-8, Shibakoen, Minato-ku
Tokyo 105 Japan.

In addition to publishing this journal, the Club also holds seminars for beginners, conducts an annual meeting and microcomputer contest, and provides an information service.

The Japan Microcomputer Club would like to interact with American clubs and would appreciate hearing from interested parties. A beginning in exchange took place at the Computer Faire in San Francisco where Professor Toshiaki Yasuda of the School of Telecommunication Engineering, Tokyo Denki University, met many interested Americans.

APPLE EDUCATION FOUNDATION AWARDS

21 Grants for the Development of Microcomputer-aided Learning

CUPERTINO, CA—March 17, 1980—The Apple Education Foundation today announced the award of over $120,000 worth of microcomputer equipment for the development of general and medical education materials using microcomputers.

The foundation awards provide twenty-three microcomputer systems valued at over $120,000, in support of education projects ranging from preschool through college, medical education, and training. They are a part of the foundation's long-range efforts to support opportunities in education through the use of expanding microcomputer technology.

The foundation's current series of grants provide one to three Apple II computer systems each, and include supporting hardware, software, and some cash awards.

The twenty-one grants awarded are as follows:

General Education

- *Mathematics*. $9,410 to promote intuitive understanding of fractions and decimals at the elementary school level to Dr. Rex Thomas, Iowa State University at Ames.
- *English*. $8,583 for microcomputer programs that teach recognition of letters and words to Roland J. Cross, Oregon School District, Oregon, Wisconsin.
- *Business and economics*. $5,017 for microcomputer college courses to Dr. Wilbur F. Pillsbury, Knox College, Galesburg, Illinois.
- *Chemistry*. $4,533 for the development of computer-assisted instruction (CAI) graphics for chemistry classes to Dr. M. Lynn James, the University of Northern Colorado.
- *Chemistry*. $5,855 for the development of computer-based programmed instruction modules to Dr. Gordon M. Barrow, Milne Press, Carmel Valley, California.
- *Bilingual education*. $4,898 to demonstrate the effectiveness of CAI in bilingual education, to the Stanford Avenue School in the Los Angeles Unified School District.
- *Administrative*. $3,859 to develop diagnostic student records for junior high schools, to Dr. William H. Robinson, Marshalltown, Iowa, Community Schools.
- *Computer literacy*. $8,605 for programs to familiarize people with computers to James W. Garson, University of Notre Dame.
- *Cash management*. $6,335 for cash management/investment programs to Dr. John M. Whitmer, Iowa State University.
- *Elementary education*. $2,260 for a CAI "life skills" curriculum by Margaret Cole, Bowditch Middle School, Foster City, California.
- *Elementary education*. $5,085 for the development of CAI to learn basic skills using consumer applications to Willis Ann Corcoran, Omaha, Nebraska.
- *Word processing*. $6,200 to teach students microcomputer-based word processing systems to LeRoy Finkel, Menlo Park, California.
- *Art education*. $7,119 to develop microcomputer applications in art education to Dr. Beverly J. Jones, University of Oregon.
- *Statistics*. $4,957 to develop instruction materials in college statistics, California State University at Chico.
- *Information gathering*. $6,537 for teaching college students the use of library reference materials and data banks to Dr. Brenda Branyan, Utah State University.
- *Music education*. $8,856 to develop CAI in ear training at the North Texas State University's School of Music to Dr. Rosemary Killam.
- *Home economics*. $2,605 for nutritious meal planning to Dr. Cheryl Hausafus, Iowa State University.

Medical Education and Training

- *Hand-eye coordination*. $4,445 to develop a hand-eye coordination diagnosis and training system to Dr. Jerry Ward, Education Services Management Corporation, Research Triangle Park, North Carolina.
- *Diabetics and nursing*. $6,538 to teach diabetic care skills to nursing students in instruction tutorial/simulation format to Dean Charles Sorenson, Grand Valley State College Allendale, Michigan.
- *Artificial speech*. $9,637 to develop programs that enable speech-impaired children and allow children, lacking manual control, to learn arithmetic.
- *Medical problem solving*. $5,239 to develop a system to teach medical problem solving techniques in simulated patient encounters to Dr. William Schwartz, Children's Hospital, Philadelphia, Department of Obstetrics.

"Our continuing goal is to place computers in the hands of people who can develop innovative learning techniques and who can demonstrate their effectiveness," said G. Gregory Smith, the Apple Education Foundation's executive director.

For grant applications, address enquiries to: The Apple Education Foundation
20605 Lazaneo Drive
Cupertino, CA 95014

EDITING SOFTAPE PREFIX PROGRAMS

BY DONALD R. OPEDAL

My programming is usually a "trial and error" procedure. I like to try different methods of producing the same results, change formats and experiment with my program. This usually requires rewriting the program several times as one version is replaced by another. It is a matter of convenience, if not a necessity, to be able to save different versions or the final version at the end of a session and to be able to run them again when desired.

SOFTAPE (10432 Burbank Blvd., North Hollywood, CA 91601) has developed several programs which make use of 6502 machine code. The machine language programs, referred to as "prefix" programs, are disguised as INTEGER BASIC programs for easy LOADing and SAVEing. The prefix programs are then appended to a BASIC program and CALLed when needed.

As a general rule, once the prefix program has been attached to the user's BASIC program, the combination can no longer be edited and SAVEed. However, for the disk user this is not completely true.

The normal use of SOFTAPE prefix programs allows writing the BASIC program into memory, appending the prefix program, and then SAVEing them as a single program. Once the program is run, the prefix program is no longer seen by the user and the BASIC program cannot be SAVEed, although changes can be entered and the program run at anytime. By using a simple technique described here, the BASIC program can be run, edited, and SAVEed on disk when correct. The corrected program is then appended to the prefix program once again and SAVEed as a single program.

This is accomplished by making use of the EXECute command which allows text files to be loaded into memory. These text files can be a BASIC program to replace an existng program or be added to it. A BASIC program is saved as a text file by using the following routine taken from the APPLE COMPUTER publication, *Contact*, No. 5. (For this example, the symbol "@" means "control D".)

```
5 PRINT "@ OPEN X"
POKE 33,33 : PRINT "@
WRITE X" : LIST :
PRINT "@ CLOSE" :
END
```

By entering the command RUN 5, the program will open a file called "X" and LIST the BASIC program into the file. By using the command EXEC X, the BASIC program can be loaded back into memory as if it had been typed in or LOADed.

BASIC Programs that use a SOFTAPE prefix program must begin with line 0. By making line 0 = GOTO 10, the above program can be entered at line 5 and can be used at any time to save the BASIC program.

The user now has the luxury of loading and running several different programs appended to the same prefix program without having to load and append the prefix to each one. This allows for faster editing and trial of different versions and allows a given program to be modified by adding new lines to it from disk.

HOME VIDEO DISPLAYS

BY LEN LINDSAY

Almost all home computers use a video display of one type or another. This is a major advance from slow teletype output. Some computers come with a built-in monitor while others leave that option to you. I would like to present a perspective on video displays. Though not an expert on the subject, I did try to ask the right questions of people who are.

Many home computers allow you to connect your computer to your TV via the antenna screws. One advantage is economy, since the video display is then not part of the computer cost. Since most homes include a TV, using it for computer video display seems reasonable. Another advantage is that you can take your computer from house to house quite easily, using the TV in each house as the video display.

A disadvantage to using the TV for the video display is that it puts the computer in direct competition with TV programs, for you can't view both at the same time. I believe that this will tend to promote exclusive use of a small TV set as video display for the computer, thus adding a hidden expense.

What about the TV itself? Which kind is better? An in-line picture tube is a preferable feature of most sets now. If your computer has color output (Apple, Atari, OSI, etc.), you may hook it to either a black and white or color set. With black and white, the colors become shades of gray. Since you will have to read small alphanumerics, make sure the set has a high resolution. This is one area where a color monitor is generally better than a standard TV set. Also a black matrix type screen is preferable since it gives a clearer image. Different manufacturers refer to this feature in different ways, each with a patent on their particular method. Matrixes of black lines, offset columns of black rectangles, or black outlined dots – all are preferable to a standard set.

Also remember that the smaller the screen the sharper the image, since the dots are closer together. Thus 9-inch to 13-inch screens make better video displays.

Video displays are naturally hard on your eyes, but there are many ways to ease this eye strain. First, keep the contrast down. Bright high-contrast letters on your screen are bad for your eyes since they must adjust to the two extremes so close together. Try to have your room lighting about the same brightness as your video display (never turn off the lights when using your computer). If your screen is brighter than your surroundings, your eyes must continually strain to adjust to this difference.

The type of room lighting and its placement are also very important. Fluorescent lighting is best if it is diffused and reflected from the ceiling. This gives even light to the room and reduces the glare, which is hard on your eyes. Sometimes moving your video display slightly will remove much of the glare on your display. Some professional video terminals come with a 'hood' across the top and sides of the screen to prevent glare from above or the sides. But while correcting the glare problem, this creates a bit of isolation and a dark area around the screen, which as mentioned above is bad on the eyes.

The colors used for the display are significant. A green phosphor screen is much easier on the eyes than black and white. When using a color TV for display, the best color combination is orange background with warm brown letters. The orange should be in the 550 angstrom range. The middle colors of the spectrum are generally easier on the eyes.

Also be aware that if any image is left on the display for too long it may tend to 'burn' the video tube at that point. As an IBM 3033 computer operator, I can testify to that. Our screens have many burned-in images. Color TV sets can be ruined if the screen gets burned spots. Thus, don't allow the same thing to be displayed for too long.

One other consideration when using a TV for your video display is the radiation factor. I have not found any research on this area yet, but I believe that sitting too close to a TV for any length of time may be bad due to minute radiation given off by the set.

As a final note, Commodore should be commended for switching from a black and white to a green phosphor built-in monitor. ATARI can be commended for their screen protection feature. If the computer is not accessed for 7 minutes, it begins changing the color registers for the screen display every few seconds. The information displayed remains the same, just the colors change. Dark colors change to light etc. This evens out the total exposure on the screen and helps prevent images from being burned into the screen.

Send your comments to Len Lindsay, 1929 Northport Drive, Room 6, Madison, WI 53704. Include a self-addressed STAMPED envelope if you expect a reply.

CURSOR PROGRAMS GRADED FOR CLASSROOM USE

BY MARLENE PRATTO

In a recent issue of *Compute* [Jan/Feb 1980], Marlene Pratto published a grade level evaluation of a selection of programs published in *Cursor*, the PET cassette magazine. (*Cursor*, Box 550, Goleta CA 93017, $33/year. Back issues are available at $3.95 per issue.) Her tabulation is reproduced below as we felt it might be of interest to our readership. Using this approach, a school or workshop group can get over seventy educationally useful programs for less than fifty dollars.

"Programs from Cursor issue 1 through 12" – graded by Marlene Pratto.

Grade Level	Program		and ye HE	rill otal DT	ogical kill LS	roblem olving PS	un F
K-2	WANDER	1					*
	BOP	4		*			
	FACE	5					*
	PAPER	7					*
	SLOT	9					*
	CANYON	12	*				
	FLIGHT	12					*
3-4	All of above						
	BRICK	1				*	
	SHARK	1	*			*	
	RACE	2	*			*	
	ZAP	2	*			*	
	DOTS	3				*	
	FLASH	3		*			
	HMAN	5		*			
	SHOOT	5	*				
	ADD	8		*			
	YAHTZEE	9				*	
	COURSE	10	*			*	
5-6	All of above						
	EST	2		*			
	MAD	2		*			
	GUESS	2			*		
	QUIX	3		*		*	
	HANOI	5			*	*	
	BSHIP	5				*	
	MIND	7			*	*	
	FBALL	7					*
	REVERS	8				*	
	DEMON	11	*				
	WIPEOUT	11				*	
	STATES	11		*			
	PICKUP	12	*				
7-8	All of the above						
	PLOT	1				*	*
	BAR	3				*	
	BOX	6			*	*	
	MAZE	8				*	
	TITRATE	10	*	*			
	PEG	11				*	

CORRECTION

The article "See What You Hear and Hear What You See" which appeared on pages 42-44 of *Recreational Computing* (Jan-Feb 1980) included excerpts reprinted, by permission, from *Atari Basic*, by Bob Albrecht, LeRoy Finkel, and Jerald R. Brown. © 1979, John Wiley & Sons, Inc.

THE SOLUTION TO THE FAIRY CHESS REVOLVER PROBLEM WHICH APPEARED IN THE MARCH/APRIL ISSUE

"Black king oscillates while White moves successively Kt, R, Kt, R, B; R, Kt, R, Kt, B; Kt, R, Kt, Re, K; Kt, K, R, K, Kt; RxB mate." (e is a notation for rook on the eth or 5th square.)

by Herb Kohl

Continued from pg. 30

Problem 21:

Hints: Column 10: Y disappears! So it must be digit *1* borrowed by Column 9. Col. 7: R – B = R. B acts like 0 (zero), but you will need proof that it isn't 9, due to carries.
Cols. 10 and 9: YB – J = T. If B were 9 it wouldn't be borrowing 1 (Y) from Column 10. So B *is* 0 (zero)!
Co.. 4: T – Y (1) = B(zero,0). So T must be 2, but has lent *1* to Column 3.

Now that you know that B is 0; that Y is 1; that T is 2; you get J from Col. 9; then C from Col.8; then U from Col. 1; then R and K from Cols. 2 and 3. In Col. 9, you see YB less J leaves T. So J must be 8 to make it: 10 – 8 = 2. Col. 4 lent 1 Col. 3, so that J is really 18, so the two Ks are 9s.

OK! You finish! You've got it made already!

Problem 22:

Hints: Three letters identify themselves. Shuffle digits to fit the rest! Too easy?

Problem 23:

Hint: Only that the root is evident.

Problem 24:

Hint: Columns 5, 6, 8 and 9 are logical keys.

A SIMPLE TRS-80 SEARCH MODULE REVISITED

In the March/April issue of *Recreational Computing* Ramon Zamora published a tutorial article and a computerized directory search program for the TRS-80. Unfortunately the listing which appeared had many many typographical errors. We apologize for any inconvenience it may have caused our readers. It is printed here in corrected form.

```
10    REM** COMPUTERIZED DIRECTORY SEARCH **
20    DIM N1$(10), N2$(10), A1$(10), A2$(10), C$(10), S$(10), Z$(10), T$(10)
30    CLS: INPUT "NUMBER OF DATA ITEMS: "; D: GOSUB 1000
40    CLS: PRINT "COMPUTERIZED DIRECTORY"
50    PRINT STRING$ (22, "–")
60    PRINT @ 256, "WHAT DO YOU WANT TO DO? SEARCH BY:"
70    PRINT @ 320, " 1–FIRST NAME"
80    PRINT @ 384, " 2–LAST NAME"
90    PRINT @ 448, " 3–STREET ADDRESS"
100   PRINT @ 512, " 4–STREET NAME"
110   PRINT @ 576, " 5–CITY"
120   PRINT @ 640, " 6–STATE"
130   PRINT @ 704, " 7–ZIP CODE"
140   PRINT @ 768, " 8–AREA CODE"
150   PRINT @ 832, " 9–STOP THE PROGRAM"
160   K$ = INKEY$: IF K$ = "" THEN 160 ELSE K = VAL(K$)
170   IF K = 9 THEN END
180   IF K = 0 THEN 160
190   CLS: INPUT "ENTER THE SEARCH KEY: "; K$: P = 0
200   FOR N = 1 TO D: ON K GOTO 210, 220, 230, 240, 250, 260, 270, 280
210   IF LEFT $ (N1$ (N), LEN (K$) ) = K$ THEN 290 ELSE 320
220   IF LEFT$ (N2$ (N), LEN (K$) ) = K$ THEN 290 ELSE 320
230   IF LEFT$ (A1$ (N), LEN (K$) ) = K$ THEN 290 ELSE 320
240   IF LEFT$ (A2$ (N), LEN (K$) ) = K$ THEN 290 ELSE 320
250   IF LEFT$ (C$ (N), LEN (K$) ) = K$ THEN 290 ELSE 320
260   IF LEFT$ (S$ (N), LEN (K$) ) = K$ THEN 290 ELSE 320
270   IF LEFT$ (Z$ (N), LEN (K$) ) = K$ THEN 290 ELSE 320
280   IF LEFT$ (T$ (N), LEN (K$) ) = K$ THEN 290 ELSE 320
290   PRINT @ 256+P*64, N1$ (N)+" "+N2$ (N); TAB (25); A1$ (N)+" " + A2$ (N)
300   PRINT @ 256+(P+1)*64, C$ (N)+" "+S$ (N)+" "+Z$ (N); TAB (40); T$ (N)
310   P = P + 2
320   NEXT N
330   PRINT @ 780, "HIT ANY KEY TO CONTINUE."
340   K$ = INKEY$: IF K$ = "" THEN 340 ELSE 40

1000  REM ** DATA INPUT ROUTINE **
1010  FOR N = 1 TO D
1020  READ N1$(N), N2$(N), A1$(N), A2$(N), C$(N), S$(N), Z$(N), T$(N)
1030  NEXT N: RETURN
1040  DATA "JACK", "JONES", "1511", "WESTPORT DR.", "MENLO PARK",
"CA", "94025", "415/555-0707"
1050  DATA "JANE", "JONARTS", "1521", "WESTBURY ST.", "MENLO PK.",
"CALIF", "415/555-1975"
```

Continued from pg. 7

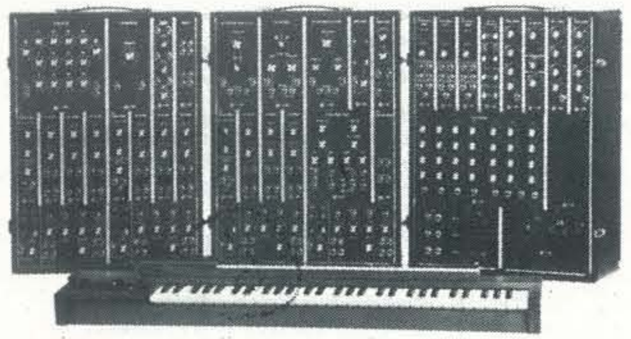


Figure 7. A Classic 1960's vintage Moog synthesizer, model IIIp, with keyboard.

Some other devices involving electromechanical technology were devised in the 60's for controlling analog synthesizers and audio equipment. These include Hugh La Caine's *Hamograph*, in which six channels of control-voltage information could be stored on a (magnetically-sensitive) sprocketed foil tape. Le Caine also developed a voltage-controlled tape recorder system, which is still installed in the Electronic Music Studio at the University of Toronto. In 1967, the composer Emmanuel Ghent introduced a device called the *Coordinome* which allowed for programmable (punched paper-tape) control of Moog Synthesizer modules.

Despite these advances in programmable or non-programmable sequencers for analog equipment, by 1969 computer technology had made great strides in music applications. Max Mathews' book *The Technology of Computer Music* [12] was a signal event that indicated the direction of the future.

Computer Origins

The history of computing devices parallels the history of automatic music devices. The main events start from the use of the *abacus*, the Chinese hand-held calculator, around 2600 B.C. Many conceptual and mechanical developments, including the perfection of sequence-control mechanisms in automatic music systems, preceded Pascal's development of a mechanical adder and subtracter called the *Pascaline*, in 1642. In 1694, Leibniz introduced a device that could also multiply and divide, a great advance. Babbage is the person who is usually cited as the originator of the modern concept of a computer with memory, for Babbage's *Analytical Engine* embodied a *store* as well as a calculating *mill*. Goldstine notes that: "It is interesting to note that Babbage thought instinctively in terms of the prime technology of his time: the steam engine." [20, p. 11] Even if Babbage did conceive of a steam-driven automaton 185 years after the musician Kirchner, and even if Babbage did hate music [21, p. 3], he must be given some credit in a computer music history. The programmable *Jacquard loom* spawned a number of instruments based around punched tape, including the mechanical piano developed by Racca in 1886 [Figure 10]. With the introduction of Hollerith punch-code machines and the formation of IBM, sequence-control devices had, in the 1920's, nearly reached their apex.

The real apex of sequence-control-type mechanisms was the electromechanical calculator, the *Harvard-IBM Mark 1*, in operation in 1939. This device used relay technology (as did the RCA Mark 1 Sound Synthesizer of 1955) to perform prepatched calculations. The first electronic computer was the *Eniac*, [Figure 11] initially run in 1945, but it did not incorporate the concept of the *stored-program* which is associated with computers today [22]. The design of an all-electronic *stored-program* computer, the *Edvac* was developed in 1944 by Eckert, Mauchly, Goldstine, and Von Neumann [20, p. 184-203] and [22]. However, the machine was not actually built until 1950. In the meantime, the transistor had been invented at Bell Labs (1948) and core memory had been conceived by Jay Forrester of the *Whirlwind* computer project at M.I.T. in 1947 [20, p. 310]. In 1949, a group at Cambridge University in England introduced the *Edsac* computer [23], in which both instructions and data resided in a common store. The present computer age was then underway.

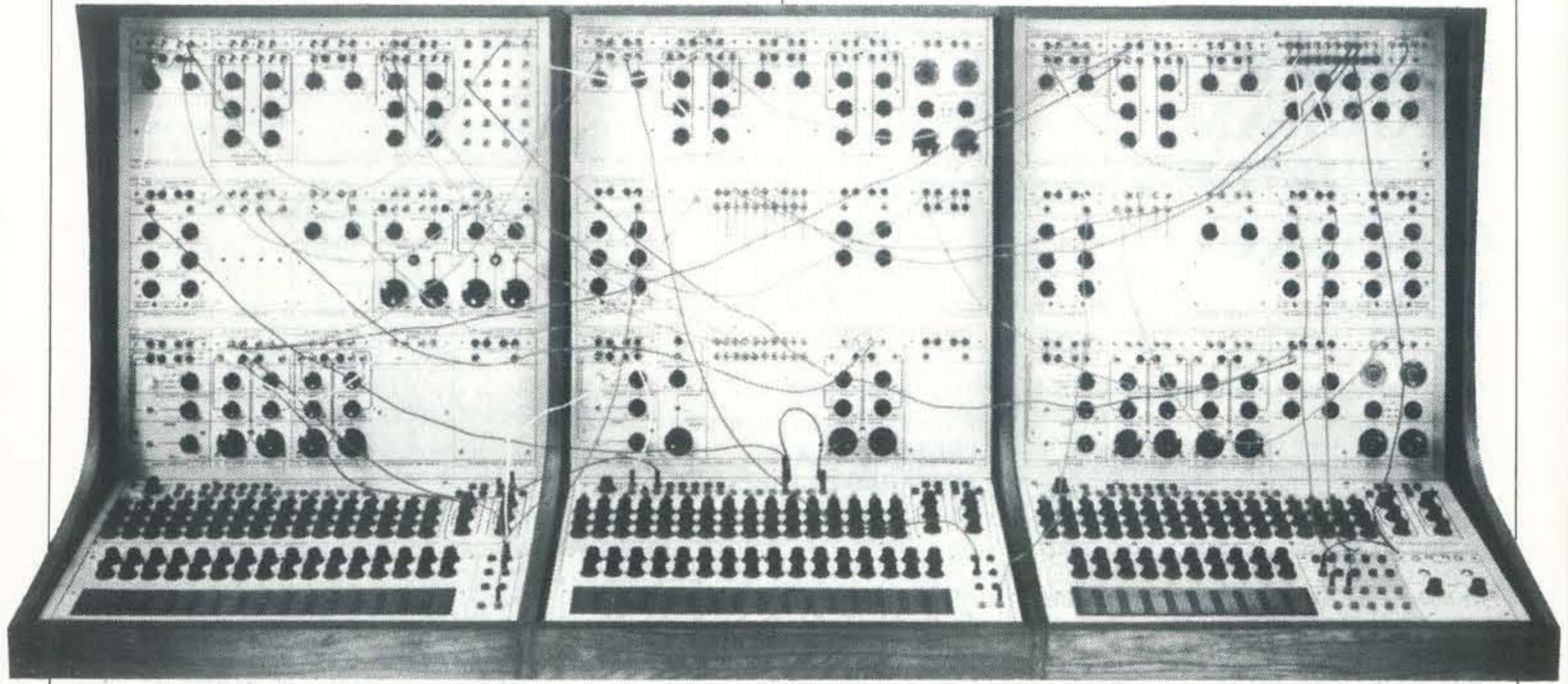


Figure 8. A Buchla Electronic Music System, series 100. Notice the three *sequencer* modules for generating present sequences of control voltages (top of lower cabinets).

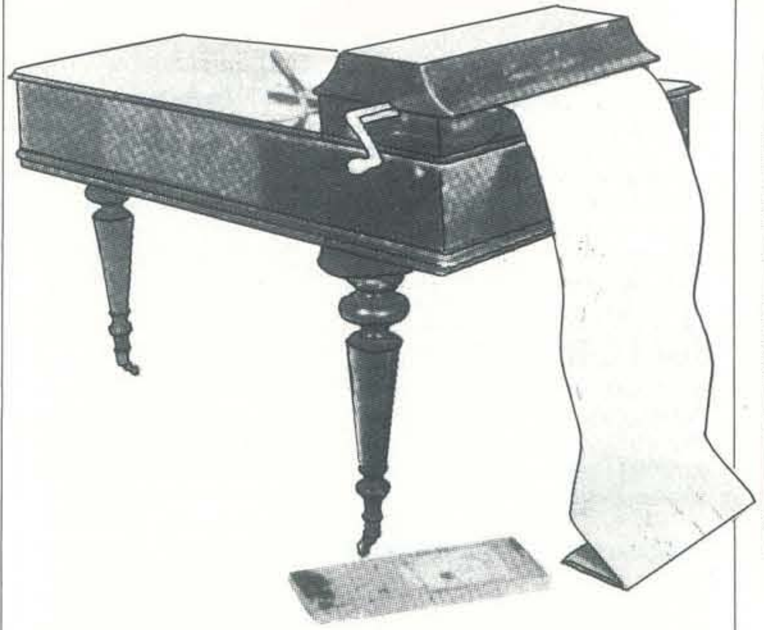


Figure 10. An example of a mechanical piano using a form of punched paper tape technology similar to the Jacquard Loom. The lever behind the playing handle controls the dynamic level.

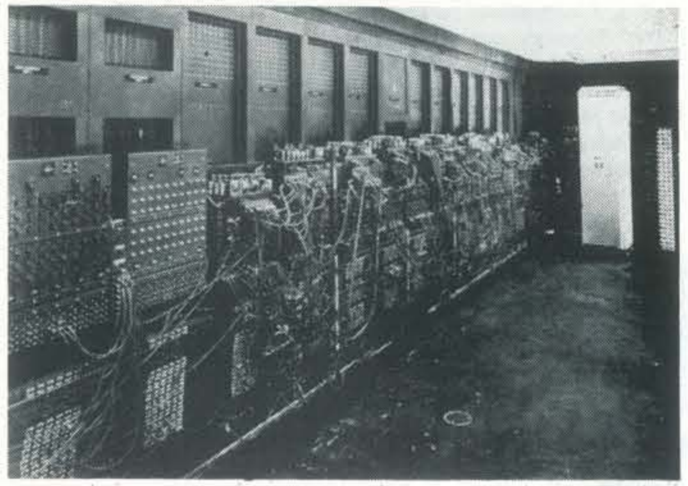


Figure 11. The *Eniac*, the first electronic computer, built at the University of Pennsylvania in 1944-45. It contained 19,000 vacuum tubes and consumed 200,000 kilowatts of power. (Photo from *Computers and Computation* (1971) W. H. Freeman, San Francisco).

Computer Music Beginnings

In 1955, Lejaren Hiller and Leonard Isaacson, then at the University of Illinois, began their famous experiments with computer music composition. Their project involved the computer generation of score data; the scores were then realized by traditional instrumental means. Central to this work was the notion of *music as an algorithmic process*. This was not the first attempt at "algorithmic composition" (*cf.* below) but it was the first known attempt at computer-assisted composition [24]. Hiller's *ILLIAC Suite* was first performed August 9, 1956, in Urbana. A less ambitious project around the same time was heralded in the *New York Times* as follows: "Brain Computes New Tune for TV." That news item discussed M. Klein and D. Bolitho's work on the tune *Push-Button Bertha* using a Burroughs *Datatron* computer.

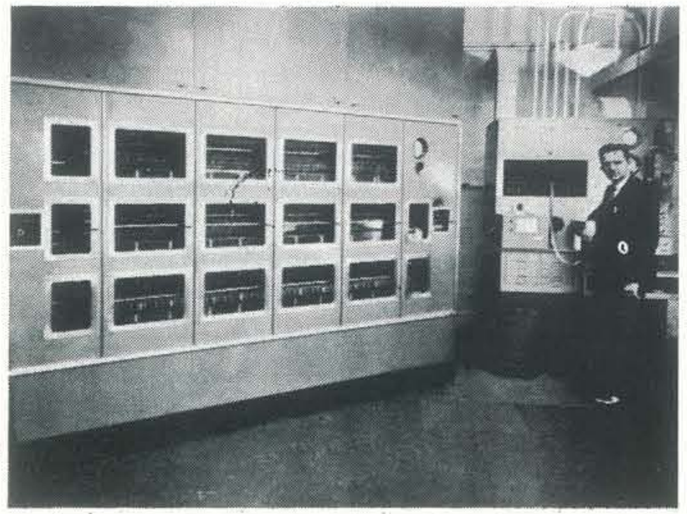


Figure 12. The ILLIAC digital computer at the University of Illinois, 1959 with composer Lejaren Hiller at right. (Photo: Myron Davis, from *Computers and Computation* (1971) W. H. Freeman, San Francisco)

In 1957, at Bell Laboratories in New Jersey, some recent IBM computers had been delivered to the Behavioural Research Lab, and a project began there to generate *sound* from the computer (both speech and musical sound) for various psychological and acoustical experiments [25]. It is interesting to note that the first fragment of music with computer-generated sound *In the Silver Scale* (1957) by N. Guttman of Bell Labs, was an original piece in *just intonation* (a temperament based on integer-ratio intervals between tones). The fragment was generated using Max Mathews' MUSIC I program for the IBM 704 computer.

MUSIC I could only synthesize one voice of sound, with one waveform, and the only waveform used in Guttmann's study was the triangle wave [26]. In 1958, Mathews and his associates introduced the program MUSIC II, which offered four independent voices of sound and a choice of sixteen waveforms stored in memory. No attack or decay envelopes for the sounds were provided, though. A major advance came with the introduction of the program MUSIC III in 1960, which included the important *unit generator* concept. Unit generators are the building blocks of most computer sound synthesis programs today. Each unit generator is a signal-processing module. By combining unit generators into specific interconnections (similar to the "patches" associated with analog synthesizers) one can create digital *instruments* and ultimately entire digital *orchestras*. In 1962, Mathews and his associates produced the sound synthesis program MUSIC IV, variants of which are still in use today. MUSIC IV offered better computational facilities including a macroassembler especially developed for the music project; MUSIC IV ran originally on the IBM 7094 computer. A machine-independent version of essentially the same program was later developed, written in Fortran, and called MUSIC V [12]. Since its introduction in 1969, MUSIC V has been a standard tool for computer music, implemented on everything from large, dual-processor Burroughs B6700 systems (48-bit word) to the PDP-11 minicomputer (16-bit word).

The earliest use of computers for musicological purposes is reportedly Bernard H. Bronson's use of an "IBM" for handling "large masses" of quantitative data on folk tunes [21, p. 15]. A problem with this reference is that IBM did not make an electronic digital computer until 1953, with the introduction of its Model 701! One can only assume that Bronson actually worked with a machine such as IBM's SECC (a sequence-controlled, relay and tube calculator installed at IBM World Headquarters in New York from 1948 to 1952) or some other electromechanical monstrosity.

The first reported use of "music transcription by computer" (1957) involved entering notes into a computer memory *via* a code language. The codes were entered on punched cards read into the machine; routines were written for "automatic transposition and printing" (in alphanumeric notation) of the resulting musical fragments [27].

Hybrid (Analog/Digital) Sound Synthesizers

In the late 1960's, a number of systems were built by interfacing a minicomputer (a small computer costing less than $5000) to an analog sound synthesizer. The computer generated signals which were converted into analog form by a *digital-to-analog converter* (DAC). Each voltage-controllable input of a synthesizer was hooked up to a DAC; since control-voltages do not vary rapidly (any one musical parameter rarely changes more than 100 times per second) the minicomputers had little trouble keeping up with the music.

The basic idea behind digital control of analog sound synthesizers (voltage-controlled synthesizers) is that the computer is a repository for tables of *functions*, which are initiated at specific time-points. In addition, the "patching" of modules to one another may be automated, and similarly linked to a sequence of time-points. An inexpensive PDP-8 (12-bit) system was the core of the *MUSYS* hybrid system at EMS, London, developed by Peter Grogono and Peter Zinovieff [13, 14]. Ed Kobrin, at the University of Illinois, also built a hybrid system, called *Hybrid II*, based around a PDP-8. Later, at the Center for Music Experiment at UCSD (La Jolla) he and Jeffrey Mack developed their successful *Hybrid IV* system around a small PDP-11 minicomputer. [15] The system was portable, and concerts in the United States and Europe were performed. [Figure 9] Other early experimenters with hybrid systems include Max Mathews and F.R. Moore and their *Groove* systems at Bell Labs. One interesting feature of their approach is that it was based on graphics techniques [16]. Hybrid systems were also developed at Toronto, by Gabura and Ciamaga [17], at Yale, by David Friend of ARP [18], and at EMS, Stockholm, as described by Wiggen [19].

Modern developments in computer music (since 1970) are too numerous to survey in this short article. Back issues of *Computer Music Journal*, the *Journal of the Audio Engineering Society*, and *Interface* are all valuable source of information on developments of the 1970's in computer music. At this juncture, a closer look at the fundamentals of digital sound synthesis is in order.

Fundamental Principles of Digital Sound Synthesis

First, music takes electronic form as a changing voltage in a wire; this is how speakers in stereo systems are driven. A positive voltage pushes a speaker cone forward, and a negative voltage will pull that cone inward. If the voltage varies fast enough, the speaker cone is vibrating at an *audio rate*; the air around the speaker is vibrated, and we hear sound (vibrating air). Through a device called a *digital-to-analog converter* (or *DAC*) it is possible to change bit streams sent out by a computer (through one of its input/output ports) into a varying voltage of the kind that pushes speaker cones around [Figure 13]. These bit streams are generated in units called *samples*. A sample is like a time-slice of a music signal. Sampling theory tells us that in order to represent a sound signal with frequencies up to *n* Hertz (or cycles-per-second) it is necessary to generate 2*n* samples. Thus, to generate a signal of 10

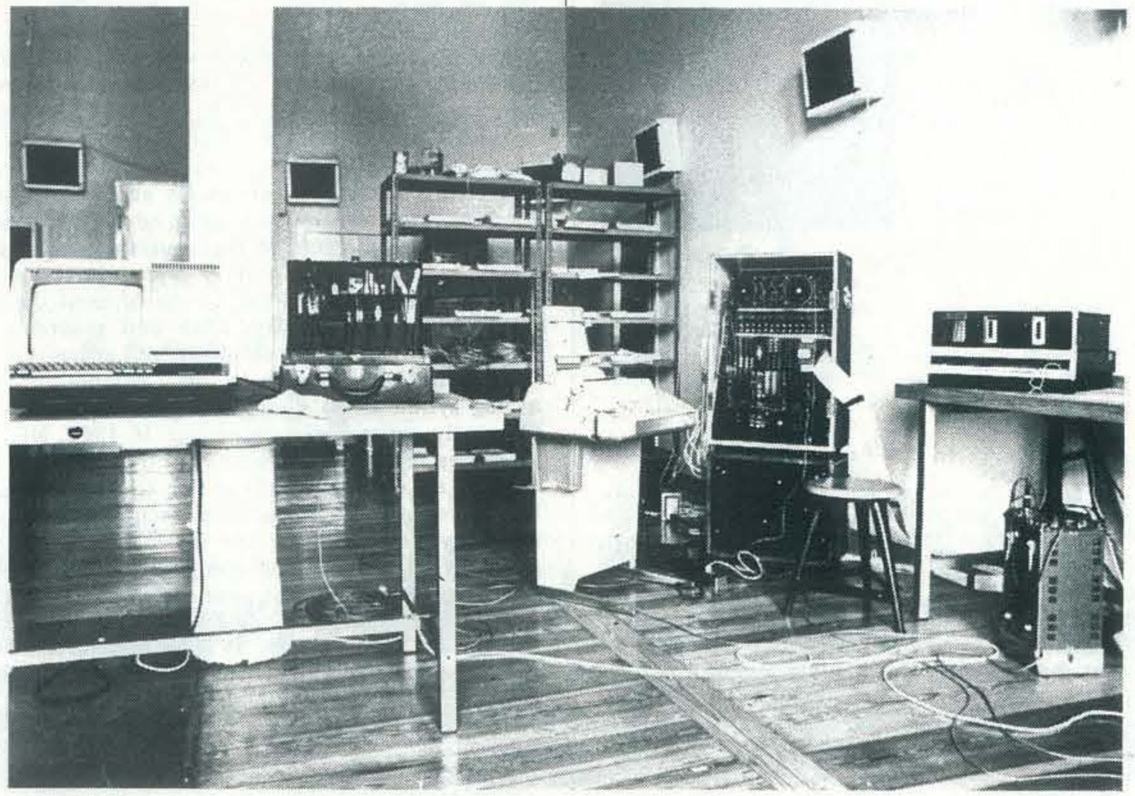


Figure 9. Ed Kobrin's *Hybrid IV* workshop, set up in Berlin. From left-to-right: VT52 CRT (on table), teletype, analog sound-generating equipment, PDP-11/10 with dual-cassette unit (on table). One the walls: some of the 16 speaker units to which sound could be distributed.

KHz it is necessary to produce 20,000 samples for each *second* of sound. Clearly, computer music is a computationally intensive application. While the ear can hear up to 20 KHz, often computer music systems work with more restricted bandwidth for purely economic reasons. Higher-speed computers cost more than lower-speed ones, and to store a wide-bandwidth audio signal can take enormous quantities of disk space.

Sound quality in computer music also depends on the *sample width*, or number of bits used to represent each sample. As a rule of thumb, one obtains six db (decibels) of signal-to-noise (s/n) ratio for every bit in a sample. For example, with 8-bit samples (fed through 8-bit DACs) one can obtain a signal to noise ratio of 48db, which is really not too good by today's audio standards. In fact, it's pretty poor. Sounds produced by 8-bit DACs have a very audible hiss and lots of harmonic distortion. So what then is the phenomenal "digital sound" that some record manufacturers have been touting so heavily? Very high precision audio systems may be built using 16 or more bits of sample accuracy. For example, one commercially-available digital sound synthesizer uses 20-bit samples internally, which results in around a 120 db s/n ratio—virtually beyond the ear's discrimination. Even this manufacturer uses 16-bit DACs though, for economic reasons; DACs of greater precision are not economically justified. If one is on a strict budget, good 12-bit DACs will suffice pretty well.

To generate sound from a computer the most common method is to *scan a wavetable*. In this technique, one simply stores a waveform (say a triangle wave [Figure 14]) in an array. We then scan the values in the array, sending each one in turn to a disk file in which samples are stored. (See the listing below in Pascal.)

```
program sound (store);
   type
      index = (0..1023); {sample table length}
      samples = (-32768..32767); {16-bit DAC}
      wavetable = array [index] of samples;
   var
      triangle : wavetable;
      sine : wavetable;
      store : file of samples;
procedure filltable (tablename : wavetable);
   begin
         {this procedure fills each wavetable
         with characteristic values}
         .
         .
         .
   end
procedure generate (tablename : wavetable);
   begin {generate}
      var
         i, j : index;
      begin {one second of sound at 40 KHz}
         for i : = 0 to 39 do
         for j : = 0 to 1023 do
            write (store, triangle [j]);
      end          {one second}
   end {generate}
begin {mainline}
   filltable (triangle);
   generator (triangle);
end {mainline}
end. {program}
```

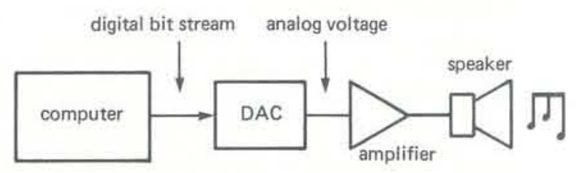


Figure 13. A computer interfaced to a DAC, amplifier, and speaker for sound generation.

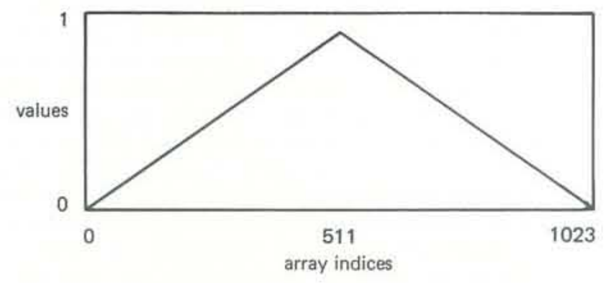


Figure 14. A triangle waveform, stored in a computer array.

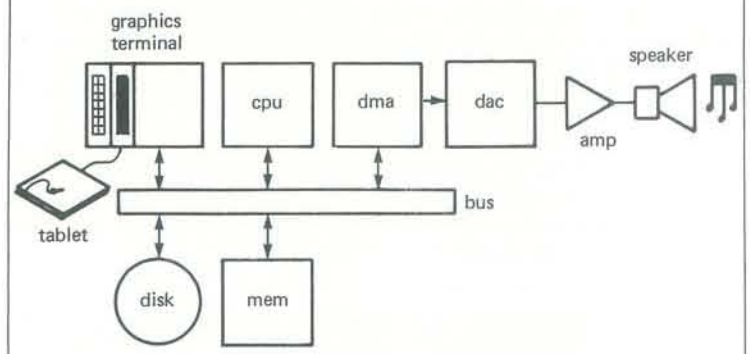


Figure 15. A microcomputer configuration for stand-alone digital music synthesis.

Typically another program is written which gets the central processing unit (CPU) of the computer to fetch samples from the disk file storage and put them into a reserved block of memory (*e.g.*, block *A*). At the same time this program activates a direct-memory-access (DMA) device which waits until block *A* is filled, then it starts fetching samples from block *A* and send them to the DAC, one-by-one, in a smooth manner. When block *A* is all consumed, then it fetches from block *B* (which has been filled in the meantime by the CPU). This scheme, called *double-buffering*, is a very common technique for generating (pre-computed) sound from disk storage. A typical hardware configuration needed to support such a system is shown in Figure 15. One can find almost the same configuration in systems from large IBM and Burroughs computer-based music installations to microcomputer systems. There is one very major advantage to this "pure" form of computer music. Since all sound-generating instruments are modelled in software, it is fairly easy to add or modify synthesis techniques, or any other music subsystems, for that matter, such as composing languages, graphics systems, or music printing software.

The disadvantage of pure computer music is first that it requires fairly large (expensive) capacity disks to store samples. Floppy disks are, for example, inadequate. 100 to 300 Megabyte disks are not uncommon at computer music studios, though a 1.2 Mbyte cartridge disk is satisfactory for storing about 30 seconds of 16-bit sound samples (at a 20 KHz sampling rate).

Second, most computers require a great deal of time to compute the millions of samples necessary for sound; computing ratios of greater than 200-to-1 (200 seconds of computation for 1 second of sound) can be expected for a complex sound computation on a small computer. If you can let the computer cook all night to generate a couple of minutes of sound then this is fine, but sometimes real-time generation can be very useful in realizing a musical idea.

Digital Synthesizers

Digital synthesizers are fast, special-purpose computers whose main purpose is generating sound samples for real-time music work. Digital synthesizers are fast because they are required to perform millions of calculations for sound samples each second. Thus, they are usually designed around high-speed logic such as Schottky TTL or ECL with lots of parallel pipelining for added speed. Usually digital synthesizers are controlled by another ("host") computer, which supplies instructions and data required for synthesis. One of the most advanced digital synthesizers built is the *Alles synthesizer* at Bell Labs [Figure 16]. This synthesizer is controlled by an LSI-11 host computer, which feeds data to the 256 "voices" of the device [28]. A system of similar magnitude is the *Samson Box*, designed by Peter Samson of Systems Concepts in San Francisco. Both of these machines produce sound on the order of the symphonic in complexity, but their price tags are over $100,000. However, many more compact 8 to 64 voice instruments are now coming onto the market, based on built-in or hook-on microcomputers.

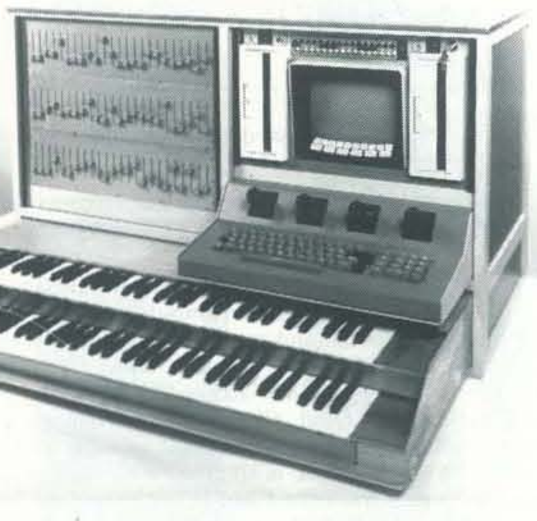


Figure 16. The Alles digital synthesizer at Bell Labs.

At the lowest end of the digital sound synthesizer scale, several manufacturers are offering inexpensive "music boards" for personal computers. Anyone who might be tempted by the low price of these boards should, however, think twice about their serious musical and technical limitations. The major problems associated with these boards are first, their inevitably 8-bit sample width (the problems with 8-bit samples were pointed out earlier) and the typically 8-bits of frequency select resolution. As to sample width, more like 12 to 16 bits is necessary for audio as good as a decent stereo system. The problem of frequency resolution is linked to the fact that with an 8-bit frequency select quantity, only 2^8 or 256 frequencies can be obtained. Only a gross (out-of-tune) approximation to the standard equal-tempered scale is possible with this low resolution, and alternative scale systems (such as microtones) are impossible to obtain. If digital sound synthesis has anything to offer, it seems it should go beyond the capabilities of traditional instruments. Since in traditional instruments (*e.g.*, a piano) we can get timbrally-interesting, polyphonic, real-time, in-tune synthesis, what is the point of a digital instrument with less capability than this? Particularly since frequency-select resolution is simply a matter of the width of one register, this kind of cutting corners (hampering musical expression) is an inexcusible engineering practice. Further, most of the budget boards use the very basic wavetable-scanning technique described earlier as their only synthesis technique. Thus, they can only produce a very sterile fixed-waveform sound. An ability to produce time-varying waveforms is essential for interesting sound synthesis.

Synthesis of Time-varying Waveforms

The simple wavetable-scanning algorithm described earlier is useful for generating a wide variety of *stationary spectra*, *i.e.*, tones whose spectral characteristics do not change over time. To generate the more interesting time-varying forms of spectra, a first approach might be to blend a collection of different fixed-waveform sounds into one continuously-varying sound over some short duration. Indeed, this is a useful technique, called *additive synthesis*. One usually starts with a collection of sine waves at various frequencies; these are mixed at various amplitudes (and sometimes at various *phases*) to form complex, time-varying spectra. Theoretically, any sound can be simulated with combinations of sine waves, given the data. This principle is the foundation of *Fourier analysis* of signals, and subsequent *Fourier synthesis*. The main problem with Fourier synthesis is that enormous amounts of data (saying which frequency should play when, at what amplitude and phase) are required to produce interesting time-varying spectra.

Subtractive synthesis starts from an already complex sound, such as a noise source, and proceeds to filter out selected components of the source to obtain a more controlled spectrum. By varying the filter coefficients over time, one can produce richly-varying spectra. The disadvantage of subtractive synthesis is similar to that of additive synthesis; they both require a great deal of data (the *driving functions*) and they are computationally costly. For this reason, several new techniques for digital sound synthesis have been developed which require much less data: these are called *nonlinear synthesis* methods.

The technique of *frequency-modulation* (fm) is the most well-known nonlinear synthesis technique [29]. Frequency modulation takes the composition of one waveform (the carrier) with another waveform (the modulator). It is typically implemented by using the modulating waveform as an addressing function into the carrier's wavetable.

Another interesting technique is that of *waveshaping* [30, 31]. This method is based on passing a simple waveform through a *transfer function* which distorts it into a more complex waveform. By passing the simple waveform through various parts of the transfer function over the duration of a sound event, time-varying spectra can be obtained.

Amplitude modulation is another useful technique which involves simply multiplying the signal by various functions. Other useful sound synthesis techniques have been developed for time-varying spectra, including *granular synthesis* [32], which involves combining thousands of sonic "quanta" to produce spectra, *VOSIM* [33], a flexible formant synthesis technique, *stochastic synthesis* [34], which typically involves using various combinations of stochastic functions as waveforms, and *instruction synthesis* [35], which produces waveforms from sequences of virtual machine instructions.

Machine Composition

Over a century before Lejaren Hiller's pioneering work with computer composition (1955), a machine for composition was developed by the Dutch genius Dietrich Nicholas Winkel, the real inventor of the metronome. In 1821, Winkel completed a device called the *Componium*. This machine aroused great interest in its time, and achieved some success [3, p. 19]. Although little is known today about the machine, it apparently was capable of performing variations on themes. The basis of the Componium and all of present-day experiments with computer composition is that *music is an algorithmic process* (*i.e.*, musical progressions obey compositional rules). Of course, these rules are not fixed, and usually each composer relies on his/her own combinations of rules.

In "programmed music" (G. M. Koenig's term) the composer encodes the compositional rules in the form of procedures. These procedures interact to model some form of musical behaviour. As a result of this behaviour, compositions or compositional fragments are produced by the procedures. There are many ways of working with compositional procedures, and only a few will be surveyed here.

Computer music composition programs can be divided into two major classes: autonomous composing programs, and systems for computer-aided composition. The first category comprises systems in which the composing program need only be initialized—it then proceeds to generate a score. It is still up to the composer to select from the output of a composing program what he/she wants. This output may then be edited or combined with other elements of a composition. Examples of composing programs include Hiller's work with the *ILLIAC Suite*, Xenakis' *ST* program [34], Koenig's *Project 1* and *Project 2* [36], and my own *PROCESS/ING* program. [Figure 17] [37] The basis of most of these programs is a stochastic or probabilistic process which generates musical events according to (often changing) musical constraints. Combinatorial or permutational processes, as well as grammatical processes have also been employed for automatic composition.

Computer-aided composition systems cover a broad range of implementations. The common denominator of all of them is that they provide for a significant degree of *interaction* between the musician and the computer system. It is possible to work on various musical levels with computer-aided composition. On the one hand, there are low-level music languages such as the Stanford language *SCORE* and the MIT music language *MUSIC-11*. While the SCORE language runs on a large PDP-10 computer, and the MUSIC-11 language runs on any PDP-11 (including the LSI-11 microcomputer) both are examples of languages which allow a composer to specify acoustic events in great detail [38, 39].

At a slightly higher level, an interactive composition system allows one to apply musical procedures to larger units of music data, such as entire phrases. An example of such a system is the Structured Sound Synthesis Project (SSSP) system at the University of Toronto [40]. The group there has made great strides in implementing computer graphics techniques for composition. The "windows and menus" graphics approach to musician-machine interaction promises to greatly improve the working modes of computer musicians.

```
State  Freq  Fslp  Spec  Spsl   Beg.       Dur.      End        Spac  Prox  Amp  Aslp  Den  Wav  Wslp
-----  ----  ----  ----  ----   --------   -------   --------   ----  ----  ---  ----  ---  ---  ----

 61     48    25    95   -21     86.5831    .24       86.8231   12      9   50    28    98    6   -16
 62     61   -27    81    23     87.3589    .2405     87.5994   11      6   43    -9    99   23    -9
 63     67    -5    86    18     86.9605   5.2149     92.1754   16      6   37   -20   100   24   -28
 64     68   -10    99     0     86.5386   1.5245     88.0632   20     10   37   -21    99   11   -28
 65     77   -18    87   -12     86.2160    .2708     86.4868   27     18   27   -23    99    7   -27
 66     80   -21    85   -11     86.2347    .3        86.5347   16     17   22   -29    71   24   -28
 67     84   -15    75     4     92.4429    .41       92.8529   99    100   70    -7    52   94   -11
 68    100   -12    55     2     92.3185   3.6767     95.9952   98     98   65   -10    36   97    -9
 69     85    -1    51     0     99.5983   1.1478    100.7461   97    100   60    -1    36  100     0
 70     88   -10    47     0     99.5945    .5       100.0945   85     90   54   -23    34   94     7
 71     90   -15    41    90    100.3636   1.5108    101.8745   86     62   32   -27    39   16    84
 72      5     0    40    92    103.8533    .8174    104.6767   95     79   33   -10    44   12    90
 73     91    15    19    94    103.8544    .2413    104.0958   99     70   20   -16    44   14    -7
 74     93   -18    17    95    104.3315   3.2208    107.5523  100     78   17    14    48   21   -14
 75     93   -12     1    -1    110.7087   2.9515    113.6602   91     78   17     1    49   21    -1
 76     93    -8    11   -21    110.9497   5.6723    116.6220   99     81   16    17    56   23    72
 77     96   -18     0   -20    122.1808   2.1033    124.2841   95     81   16    20    57   15    82
 78     97   -24    18   -21    122.4878    .3729    122.8606   93     77   14    24    59   16    88
 79     96   -23    10   100    122.7120    .2434    122.9554   97     55   15    16    71   25    95
 80     97    -8    12    90    123.1179   6.3887    129.5067   99     44   25   -18    80   13    94
```

Figure 17. An example of the output of a composing program, from the score *Plex* by the author.

A key to developing more powerful software systems for composition is the development of richer knowledge representations for music. Not only should music be represented on the note level, but also on the level of large-scale forms such as phrases and sections. Representations which can capture extra-syntactic aspects of music may be even more powerful. All of these notions of music representation fit naturally into a *grammar* model for music, similar to models developed in linguistic and artificial intelligence research [41]. However, most music is too complex to be represented by a simple grammar (made up only of *production rules*), so some extended grammar representations have been developed. Such systems may allow a composer to work with music on many different levels and with many different representations in a single composing session.

Other Computer Applications to Music

The computer has generated new interest in many other musical areas besides sound synthesis and composition. One of the most successful has been the application of music printing by computer [42, 43]. At least two automated music printing systems are thriving commercial enterprises, which produce high-quality output scores for a fee. These are: Dal Molin's *Music Reprographics* of Oyster Bay, New York, and *Dataland*, in Aarhus, Denmark. An operator is shown playing in a hand-drawn score at Dataland in Figure 18. A plotter traces the computer-generated version in Figure 19. An example of the high-quality output produced is shown in Figure 20.

Figure 18. Playing in a score for computerized printing at Dataland, Aarhus, Denmark.

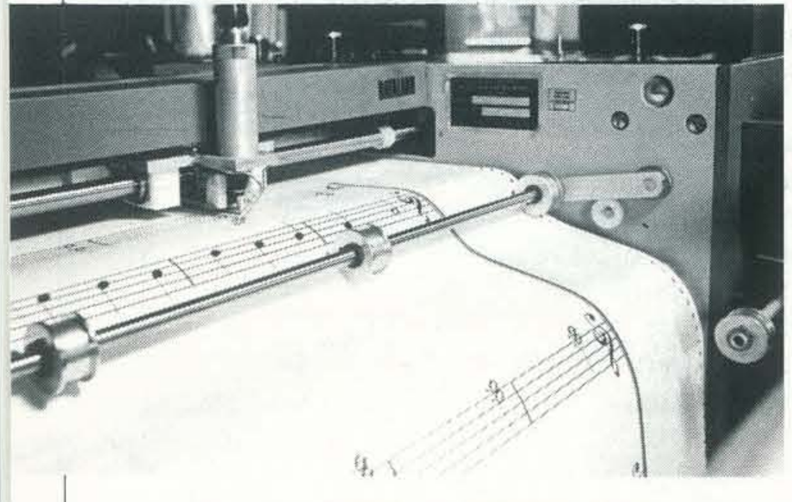


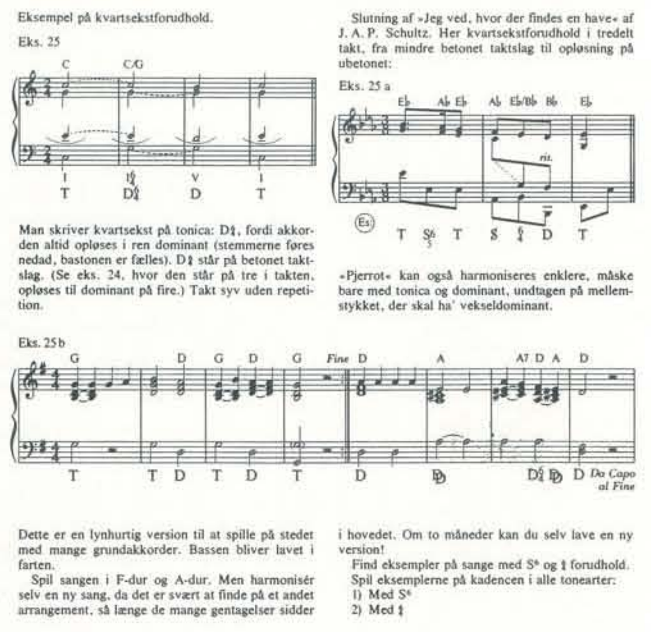
Figure 19. Plotting a score at Dataland, Aarhus.

Music analysis is another widespread application for computers. Music analysis programs typically accept music data encoded in some standard form; the data is then processed with analysis procedures, and the results (often in numerical form) are then printed out by the computer program. A major music analysis project has been based around the *MUSTRAN* notation system developed by Jerome Wenker of San Francisco [44]. Other projects have been organized around dialects of the *DARMS* notation system. Techniques of analysis typically involve tracing the frequencies of various pitch intervals and other statistical criteria. However, some recent developments indicate that a cross-breeding between the fields of artificial intelligence and music analysis may be a very fruitful endeavor [45, 46]. The central notion in this kind of research is constructing a system which actually "understands" the music it is analyzing, *i.e.*, it is able to build a coherent knowledge representation.

Intelligent Musical Devices

An intelligent musical device is an instrument which can not only remember musical sequences played into it, it can even listen and "understand" music. Such an instrument does not exist now, but there is considerable research and development work being done which could lead to such a device. Such an instrument will be able to recognize not only frequencies, amplitudes, and durations (as analog devices do today, *i.e.*, frequency followers, amplitude followers, and noise gates) but also larger syntactic forms and functional characteristics of the music. Acting from a base of programmed or even acquired grammatical knowledge, such a device will be able to listen and respond intelligently not just to sound, but to music.

Immediate applications of intelligent musical devices are in the realm of digital sound editing, live performance, and interactive composition. An intelligent sound editor could trace the path of a voice in a complex polyphonic texture; that voice could be modified or deleted without affecting the other voices. This kind of context-sensitive editing capability (crossing syntactic boundaries) is impossible today. In live performance, an intelligent musical device could respond in cues with various musical outputs, such as "following along" in some manner, with the performer. Another live use would be as an intelligent "stage manager" capable of responding to cues by starting and stopping various aspects of the performance. In interactive compositional situations, the composer could benefit from an intelligent assistant, able to communicate perhaps in a natural language (or a subset enriched by musical terms) with powerful musical capabilities. Such a system could answer questions (using large musical data bases) and even make suggestions in some cases.


Eksempel på kvartsekstforudhold.

Eks. 25

Man skriver kvartsekst på tonica: D♮, fordi akkorden altid opløses i ren dominant (stemmerne føres nedad, bastonen er fælles). D♮ står på betonet taktslag. (Se eks. 24, hvor den står på tre i takten, opløses til dominant på fire.) Takt syv uden repetition.

Slutning af »Jeg ved, hvor der findes en have« af J. A. P. Schultz. Her kvartsekstforudhold i tredelt takt, fra mindre betonet taktslag til opløsning på ubetonet:

Eks. 25 a

»Pjerrot« kan også harmoniseres enklere, måske bare med tonica og dominant, undtagen på mellemstykket, der skal ha' vekseldominant.

Eks. 25 b

Dette er en lynhurtig version til at spille på stedet med mange grundakkorder. Bassen bliver lavet i farten.

Spil sangen i F-dur og A-dur. Men harmonisér selv en ny sang, da det er svært at finde på et andet arrangement, så længe de mange gentagelser sidder i hovedet. Om to måneder kan du selv lave en ny version!

Find eksempler på sange med S⁶ og ♮ forudhold.
Spil eksemplerne på kadencen i alle tonearter:
1) Med S⁶
2) Med ♮

C C/G
I V I
T D T
Eb Ab Eb Ab Eb/Bb Bb Eb
rit.
Es
T T S D T
G D G D G Fine D A A7 D A D
T T D T D T D D D Da Capo al Fine

Figure 20. Example of output produced by the Scan-note system at Dataland, Aarhus. Note that both text (in Danish) and musical symbols are plotted by the computer.

Summary

Computer Music, though still in its infancy, has produced significant results, both musical and technical in nature. Interest and involvement in computer music is on the ascent, and the future appears promising. Though, of course, computers will be abused in musical applications, as they seem to be in all applications in this society, many liberating possibilities are latent in computer music as well. Computer music has already demonstrated its potential for opening up entirely new sound dimensions; digital processes have the potential for greatly improving audio fidelity and control.

Still, computer music consumes a lot of computational resources. At the time of this writing, it is perhaps still too expensive for the individual musician to own a computer music system which produces good audio quality and is musically very flexible. However, with the new generation of microprocessors, increasingly massive main memory chips (*e.g.*, bubble memory), and Winchester disk and other secondary memory technologies, a very good system may very soon become economically feasible. Digital synthesizers, even as they enter the market at a rate of several per year, need to pass through a price/performance ratio one-half of what it is today. This, too, is only a matter of a short period of time. Meanwhile, many individuals and research groups will continue to concentrate on highly interesting new musical applications for existing computer music systems.

References

[1] Simon, E. (1960) *Mechanische Musikinstrumente früherer Zeiten und ihre Musik*, Breitkopf und Hartel, Wiesbaden.
[2] Struik, D. (1967) *A Concise History of Mathematics*, 3rd Edition, Dover, New York.
[3] Buchner, A. (1978) *Mechanical Musical Instruments*, translated by I. Urwin, Greenwood Press, Westport.
[4] Leichtentritt, H. (1934) "Mechanical Music in Olden Times," *Musical Quarterly*, Vol. XX, Schirmer, New York, pp. 15-26.
[5] Clark, J. E. T. (1952) *Musical Boxes*, Cornish Bros., Birmingham.
[6] Ashton, A. C. (1971) "Electronics, Music, and Computers," UTEC-CSc-71-117, Univ. of Utah, Salt Lake City.
[7] Noncarrow, C. (1977) *Music for Player-Piano*, recording available from 1750 Arch St. Records, Berkeley, California 94709.
[8] Varese, E. (1966) "The Liberation of Sound," notes compiled by Chou Wen-chung, in *Perspectives on American Composers* (Boretz and Cone, eds.) (1971) W. W. Norton, New York, pp. 25-33.
[9] Luening, O. (1975) "Origins" in *The Development and Practice of Electronic Music* (Appleton and Perera, eds.) (1975) Prentice-Hall, Englewood Cliffs.
[10] Davies, H. (1968) *International Electronic Music Catalog*, The MIT Press, Cambridge.
[11] Barron, L. & L. (1954) soundtrack for the film "Forbidden Planet," recently re-released by Planet Records, Box 3977, Beverly Hills, CA 90212.
[12] Mathews, M. (1969) *The Technology of Computer Music*, The MIT Press, Cambridge.
[13] Zinovieff, P. (1969) "A Computerized Electronic Music Studio," *Electronic Music Reports*, Sept., pp. 5-22.
[14] Grogono, P. (1973) "MUSYS: Software for an Electronic Music Studio," *Software Practice and Experience*, Vol. 3, pp. 369-383.
[15] Kobrin, E. (1977) "Computer in Performance," Lingua Press, Box 1192, La Jolla, CA 92038.
[16] Mathews, M. and Moore, F. R. (1970) "GROOVE–A Program to Compose, Store, and Edit Functions of Time," *CACM*, Vol. 13, Dec.
[17] Gabura, J. and Ciamaga, G. (1969) "Computer Control of Sound Apparatus for Electronic Music," *JAES*, Vol. 16, pp. 49-51.
[18] Friend, D. (1971) "A Time-shared Hybrid Sound Synthesizer," *JAES*, Vol. 19, No. 11, Dec. 928-935.
[19] Wiggen, K. (1972) "The Electronic Music Studio at Stockholm: its Development and Construction," *Interface*, Vol. 1, pp. 127-165.
[20] Goldstine, H. H. (1972) *The Computer: from Pascal to Von Neumann*, Princeton U.P., Princeton.
[21] Bowles, E. A. (1970) "Musicke's Handmaiden: or Technology in the Service of the Arts," in *The Computer and Music* (Lincoln, ed.) Cornell Univ. Press, Ithaca.
[22] Mauchly, J. (1979) "Amending the Eniac Story," *Datamation*, Vol. 25, No. 11, October, pp. 217-220.
[23] Wilkes, M. V., Wheeler, D. J., and Gill, S. (1951) *The Preparation of Programs for an Automatic Digital Computer*, Addison-Wesley, Reading, MA.
[24] Hiller and Isaacson (1956) "Musical Composition with a Digital Computer," *Program and Abstracts from the 11th National Meeting of the Association for Computing Machinery*, UCLA, Aug., p. 8 and 22.
[25] David, E. E., Mathews, M. V., and McDonald, H. S. (1958) "Description and Results of Experiments with Speech Using Digital Computer Simulation," *Proceedings of the 1958 National Electronics Conference*, pp. 766-775.
[26] Hiller, L., and Isaacson, L. M. (1959) *Experimental Music: Composition with an Electronic Computer*, McGraw-Hill, New York.
[27] Granholm, J. W. and Mitchell, M. C. (1957) "Music Transcription by Computer," *Computing News*, Vol. 5, No. 17, pp. 108-113.
[28] Alles, H. G. (1977) "A Portable Digital Sound Synthesis System," *Computer Music Journal*, Vol. 1, No. 4, pp. 5-6.
[29] Chowning, J. (1973) "The Synthesis of Complex Audio Spectra by Means of Frequency Modulation," *Computer Music Journal*, Vol. 1, No. 2 (1977) pp. 46-54.
[30] Roads, C. (1979) "A Tutorial on Non-linear Distortion or Waveshaping," *Computer Music Journal*, Vol. 3, No. 2, pp. 29-34.
[31] Le Brun, M. (1979) "Digital Waveshaping Synthesis," *Journal of the Audio Engineering Society*, Vol. 27, No. 4, pp. 250-266.
[32] Roads, C. (1978) "Automated Granular Synthesis of Sound," *Computer Music Journal*, Vol. 2, No. 2, pp. 61-62.
[33] Kaegi, W. and Tempelaars, S. (1978) "VOSIM–a New Sound Synthesis System," *Journal of the Audio Engineering Society*, Vol. 26, No. 6, pp. 418-426.
[34] Xenakis, I. (1971) *Formalized Music*, Indiana U. P., Bloomington.
[35] Berg, P. (1979) "PILE–A Language for Sound Synthesis," *Computer Music Journal*, Vol. 3, No. 1, pp. 30-41.
[36] Koenig, G. M. (1978) "My Experiences with Programmed Music," *Faire*, Vol. 4/5, pp. 26-30
[37] Roads, C. (1976) "A Systems Approach to Composition," Author's manuscript.
[38] Smith, L. (1972) "SCORE–A Musician's Approach to Computer Music," *Journal of the Audio Engineering Society*, Vol. 20, No. 1, pp. 7-14.
[39] Vercoe, B. (1979) "Reference Manual for the MUSIC-11 Sound Synthesis Language," MIT Experimental Music Studio, Cambridge.
[40] Buxton, *et al.* (1979) "The Evolution of the SSSP Score Editing Tools," *Computer Music Journal*, Vol. 3, No. 4, pp. 14-25.
[41] Roads, C. (1979) "Grammars as Representations for Music," *Computer Music Journal*, Vol. 3, No. 1, pp. 48-55.
[42] Smith, L. (1973) "Editing and Printing Music by Computer," *Journal of Music Theory*, Vol. 17, pp. 292-309.
[43] Gromberg, D. (1977) "A Computer-Oriented System for Music Printing," *Computers and the Humanities*, Vol. 11, pp. 63-80.
[44] Wenker, J. (1974) "MUSTRAN II: A Foundation for Computational Musicology," in *Computers in the Humanities* (Mitchell, J. L., ed.) Edinburgh Univer. Press, Edinburgh.
[45] Smoliar, S. W. (1980) "A Computer Aid for Schenkerian Analysis," *Computer Music Journal* (to appear).
[46] Meeham, J. (1980) "An Artificial Intelligence Approach to Tonal Music Theory," *Computer Music Journal*, (to appear).

LETTERS

CALL FOR MORE WORD FROM THE ELF CONSTITUANCY

ATTENTION: DRAGON

Before I attack, let me say that:

1. I believe in the principles of the People's Computer Company in that... "Computers are mostly used against people instead of for people, used to control people instead of free them. It's time to change all that – we need a people's computer company."

PCC/*Recreational Computing*, as an entity and as a publication emulates this philosophy in every deed and action. To this I commend thee.

2. The expressed and stated ideals seek to give all those who desire same, the chance and opportunity to compute and access to computers. Verily I repeat, I commend thee and RIGHT ON.

Those goals are very commendable indeed, however, I have been upset lately with you and my favorite publication. I mean, I place you somewhere between expectations awaiting the arrival of Playboy and computer dream books, (catalogues). What more could you ask for in a computing magazine??

Though I have been an avid reader of *RC* for nearly a year, I do not have a system as yet. So I am not an expert like most of your readers. I am not a parent, so cannot fully understand the full desire of having my kids being able to use a computer for education and fun. I am 31 now so can't fully understand the desires of a child to want to compute and not being able to afford a computer. I am in prison, so I cannot fully understand the parent or common man who wishes to have for himself or his children a computer, and cannot afford one. So, I have fully discredited myself to a point. I am a good (if I may be so bold) electronic technician, however, and dream of computers and of owning one.

In my latest book from Electronic Systems I noted a computer which is called the Quest Super Elf. It costs as listed for the basic system $106.95. The expansion kits are all very cheap and the massive expansion is under $90.00. This is not a plug for Electronic Systems but brings me finally to my point.... Wanting very badly a computer I have saved as much as possible, the money I get for my services in the prison electronics shop, which is top pay of one dollar per day. I manage to save about 10 to 15 dollars per month. Don't think that I want anyone to feel sorry for me, I made my bed.... What I am trying to say, is that of all those things and persons I have listed as being not, I feel I can financially relate to all of these people.

Now, with the Super Elf, the poor and the young can have a computer which is affordable to all. I might add that music, graphics and a video output are standard and it tells you what it is doing as it wants to be your friend and have you understand it. This is great in itself, but as I said, any kid can raise this money which is in most cases less than a fifth of a TRS-80 or comparable systems, and is something that won't put the parent in hock to his ears. All in all it is just as advertised, a beginner's computer, but completely expandable. Expandable, I might add, in affordable steps one can afford as well.

So, I turn to my favorite computer magazine (you, unless that point has been lost), for something on this little beauty, 'cause you represent the people, and what do I find? How *RC* caters and plans to further give space to the megabyte or rather mighty-bite-on-the-wallet computers only.

So, what do we find? Once again we have the power to the people, people, catering to the powerful and the rich once again. Since the Elf and its cousins are the truly *micro*-computers, and now within the range of all the masses and not just the rich, I say unto you.... Down with the Megabyte Monsters! Power to the little people, the true people of *the land of the little people*. We demand equal publication space! or by *HEX tinyBASIC* will do battle with his brother *Billy BASIC!* Now that I think of it, *Fortran Man* don't look so tough either!

Could you handle that on your conscience dragon? Not since the Civil War has brother turned against brother. After all are not tiny BASIC and Billy BASIC out to achieve the same *ends*?

All I would ask for would be the same space as SPOT. After all what does he have that the ELF doesn't? The Elf has a MONTHLY magazine called *Questdate.* That's all if at all, that Spot has. I would even settle for a quarter page space.

If the megabyte tyrants are too much above dedicating time to the lower cases to give the poor and the young people's computer their time, when the SUPER ELF I have ordered arrives I'll write the piece myself, if megabyte pride stands in the way. After all, all I have is a love of computers and lots and lots of time, (pardon the pun).

Power to the little people!

C. THOMAS HILTON
MONTANA STATE PRISON, BOX #7
DEER LODGE, MONTANA 59722

Post Script: Mail may be sent to the address above. This far out in the boonies even hate mail is welcome.

We don't really discriminate; we publish the best of what is submitted.

I'll be looking for your articles about your experiences with the Super-Elf. Be sure to get Tom Pittman's tiny BASIC modification kit so you can extend tiny BASIC as you feel the urge.

– ED

WE'LL TRY HARDER

Dear stupid chazlul!

It has come to my attention that your recent despicable and pultixious conduct on the floor of the Congress has caused your colleagues to contemptuously refer to you as a sunmabing gachloz. I find that description of you too zokniming mild to say the least. You, sir, are a nozshim, a xathzog, and a blithering mathfig! I have no doubt you're the sosroring offspring that resulted from the casual union of an idiotic rochsaz and a demented gochfef. I sure hope you get golxeted in the next election!

A Concerned Citizen

HISTORY OF FANTASY GAMES

Dear Dragons:

With the continuing interest in Computerized Fantasy Simulation games, it is interesting to consider the origin of such a concept. Our local version of Adventure credits Willie Crowther of Stanford; but computerized adventures were predated by the Dungeons & Dragons game, to which I was introduced by your periodical two name changes ago.

However, the CFS concept is much older than that. In Arthur Clarke's 1956 SF novel *The City and the Stars*, the residents of his future participate in "sagas." These sagas:

* are designed by humans
* are mediated by computer
* allow great flexibility of action
* have definite limits beyond which participants cannot stray.

Clarke's sagas are only incidental to the book, so not much is written about them. The one saga described in detail nevertheless sounds much like contemporary ones: Alvin and friends follow a floating arrow of light through the labyrinths of the Crystal Mountain after escaping from the Cave of the White Worms.

Sincerely,
Gregg Townsend
Tucson, AZ

Our next issue will be devoted to computerized Fantasy Simulation Games. For the record, Adventure was done by Willie Crowther (now at Xerox PARC) and extensively modified by Don Woods. To the best of my knowledge it was the first really complex exploration game. Can anyone add to the history? – ED

SUPER STARTREK

Greetings!

Re Robert Howarth's letter in the Jan/Feb issue. First, the good news. I am the author of a dialogue-based Star Trek program. I foresaw the need for such a program several years ago and undertook the task of implementing it. What resulted was a 56K BASIC program which eliminated the need for memorizing numerical commands.

The program accepts verbal commands and responds by generating a game which reads like a book (or in this case, a TV show). Of course you still have the option of visual displays, and although some elements are similar to those of more primitive Star Trek programs, comparing mine with those would be like comparing V'ger to a tribble.

There are over 40 possible commands, and elements and situations from 13 of the TV episodes. The program is truly addictive to those who play it, and to me.

I am currently writing a second version which will run on a TANDEM computer system. This version includes all of the BASIC capabilities plus elements from the movie – the first to do so. The program will have real-time capabilities and with any luck an infinite universe – as opposed to the usual 8 x 8 quadrant universe of typical S.T. programs. (What idiot ever decided on an 8 x 8 array in the first place?!...I've yet to see a S.T. program which uses bit mapping for quadrant representation.) My program will also have interprocess communications capability (the TANDEM having up to 16 processors with multiple processes running in each). This will allow two or more people to play simultaneously at different terminals, each thinking he is captaining the 'Enterprise,' but in essence captaining a Klingon (Romulan or Gorn) when viewed from someone else's terminal. The next step would be for me to implement it on a Computalker/Speechlab synthesis/recognition system.

Now the bad news. There are only 10 BASIC listings in existence, and the only object resides at Drexel Univ. in Phila. Five of the listings are scattered around the country, and I own the other five. The only TANDEM files are here with me in Delaware. Because of the time sent, and because I and Paramount own the copyright, I am loath to part cheaply with the listings. So anyone writing to me for a price quote... beware! However, if there is enough demand (5 people) I am willing to write a series of articles on implementing the BASIC version on a home computer.

S. Koren
Newark, DE

APPLE HOTLINE

Apple II owners who plan to use their Apple with a TV set should be aware of the following. If the set has channels quartz frequency controlled with no user fine tuning as is the latest trend, the Supr. Mod II by M & R Enterprises may not work. Use a modulator which has frequency tuneability like the one available from ATV Research, 13th & Broadway, Dakota City, Nebraska 68731.

Donald J. Stonek
Cudahy, Wisconsin

We checked with Apple on this. Phil Roybal replies. – ED

I spoke to Marty at M & R Enterprises to get a reply to the letter that you sent me. He confirmed that the present SUPRMOD II does not generally work with quartz crystal controlled television sets. However, it is sometimes possible to use the modulator with such sets on VHF, Channels 10 or 11, or UHF, Channels 32, 34, 64, 66, or 68. It will depend upon the television set.

The influx of digitally controlled television sets during the last year has stimulated M & R Enterprises to begin development of a new modulator, which will be available later on this year. This device will work with digitally controlled televisions and so will solve the problem for users.

THIS PUBLICATION IS AVAILABLE IN MICROFORM

Please send me additional information.

University Microfilms International

300 North Zeeb Road
Dept. P.R.
Ann Arbor, MI 48106
U.S.A.

18 Bedford Row
Dept. P.R.
London, WC1R 4EJ
England

Name ____________
Institution ____________
Street ____________
City ____________
State ____________ Zip ______

ANNOUNCEMENTS

Hardware

Z-80 Softcard for the APPLE offers Apple owners a second processor and the possibility of running 8080/Z80 programs on their Apple including the popular CP/M operating system. The Z-80 SoftCard will run on all configurations of the Apple and requires no hardware or software modifications. It will plug into any of the peripheral slots except slot 0 and does not affect the operation of the Apple when not in Z-80 mode. To use the SoftCard with CP/M and Disk BASIC requires 48K RAM and one disk drive. Deliveries begin in June. Price is $349. For the name of the nearest dealer contact Microsoft Consumer Products, 10800 Northeast Eighth, Suite 507, Bellevue WA 98004. (206) 454-1315.

Sup'R'Terminal, an 80 Column, 24 line upper/lower case display option for the Apple II, is available from M & R Enterprises. The Sup'R'Terminal board is user-installable and compatible with most standard software. Characters are represented as a 5x8 dot matrix. Board-based software handles such functions as upper/lower case shift, cursor movements and modes, scrolling modes and controls, clearing and linefeed functions, and character definition. Users can define their own character sets and switch back and forth at will. Retail price is $395. See your dealer or contact M & R Enterprises, P.O. Box 61011, Sunnyvale CA 94088, (408) 738-3772.

S-100 bus Z8000 16-bit Processor Card introduced by Ithaca Intersystems, supports both the non-segmented Z8002 and the segmented Z8001. For more information contact Ithaca Intersystems, 1650 Hanshaw Rd., P.O. Box 91, Ithaca, NY 14850. (607) 257-0190.

Microbot's MiniMover 5 is a complete and functional robot arm. It has a lifting capacity of 8 ounces, a positiioning accuracy of 0.013 inch, and an operational radius of 17.5 inches. Top speed is between 2 and 6 inches per second depending upon the weight of the object. Control is via an interface to a TRS-80 designed by Dr. Lichen Wang, author of Palo Alto Tiny BASIC. Available now for an introductory price of $1495 the price will soon climb to $1695. Supporting software costs $29.95 while a reference and applications manual costs $14.95 plus $2 domestic or $4 foreign postage and handling.

Electric Crayon gives Color to the TRS-80, other computers. While it is designed to be used for color graphics control and generation, the Electric crayon is really a self-contained computer system. The Electric Crayon with the EGOS operating system, 1K-bytes of refresh memory and a users manual including an assembly language listing of EGOS and BASIC demonstration programs sells for $249.95. Cabling, additional refresh memory and demonstration programs on minidisks are extra cost options. Percom Data Company, 211 N. Kirby, Garland TX 75042. (214) 272-3421.

Software

PL/I-80 from Digital Research implements the ANSI General Purpose Subset of PL/I as a compiler for the 8080, 8085, and Z80. The package runs under the industry standard CP/M and MP/M operating systems also developed by Digital Research. The full package, including compiler, subroutine library, linkage editor, and relocating macro assembler and full documentation cost $500. An impressive system! Digital Research, P. O. Box 579, 801 Lighthouse Avenue, Pacific Grove, CA 93950. 408/649-3896

A C Compiler for the 6800 has just been announced by Wintek. C is a powerful systems language developed by Dennis Ritchie for the UNIX system at Bell Laboratories. The Wintek C is designed to run under the WIZRD multitasking disk operating system on the SPRINT 68 computer. The compiler costs $495; the computer with 48K memory, dual floppy disk drives, and operating system is $3995. WINTEK Corporation, 1801 South Street, Lafayette, IN 47904. 317/742-8428.

A Tiny Pascal Compiler for $15, written in BASIC, People's Pascal runs on any 16K TRS-80 Level II machine. It comes on tape with 14 programs and 18 pages of documentation. Programs include editor/compiler, interpreter, translator, run-time system, and two demonstration programs. The compiler produces P-codes which are then translated to Z80 native code. A second version of the system, interpretive only, is written in machine language and has substantially better human factors. It sells for $23. Computerists who buy by mail should add 50¢ for each tape ordered and 6% sales tax if they live in California. Computer Information Exchange, P. O. Box 158, San Luis Rey, CA 92068.

TI 58/59 Calculator Cross Compiler allows you to develop, refine, and test programs in BASIC and then compile the program into an equivalent keystroke program for the TI 58 or 59 calculator. The compiler is written in BASIC and runs in 16K. Price is $65 for program listing, user's manual, and program documentation. A 9-track magnetic tape of the BASIC source code is also available for an additional $35. Singular Systems, 810 Stratford, Sidney, OH 45365.

TRS-80 BASIC Compiler announced from Microsoft Consumer Products compiles programs written with the TRS-80 Disk BASIC interpreter, producing Z-80 machine code that is directly executed by the TRS-80.

The package includes two diskettes containing the BASIC Compiler, BASIC runtime library and LINK-80 linking loader; complete instruction manual for using BASIC Compiler; and complete reference manual for Microsoft 5.0 BASIC. Suggested retail price is $195. Contact Microsoft Consumer Products, 10800 Northeast Eighth, Suite 507, Bellevue, WA 98004. 206/454-1315

muMATH for the TRS-80 is a symbolic math package that brings sophisticated math capability to the TRS-80 for the first time. muMATH's capabilities include exact rational arithmetic and automatic algebraic simplification. The user can control such transformations as expanding powers of polynomials and placing expressions over a common denominator. Other capabilities include trigonometric and logarithmic simplifications and symbolic differentiation and integration. All operations in muMATH are performed with cision to 611 digits.

The package includes the muMATH diskette and instruction manual. Suggested retail price is $74.95. Contact Microsoft Consumer Products, 10800 Northeast Eighth, Suite 507, Bellevue, WA 98004. 206/454-1315.

TERM80 is a Level II BASIC and machine language program which provides a variety of useful terminal features. Convert your TRS-80 to a full-duplex ASCII terminal. TERM80 is available on minidiskette with user instructions for $24.95. Percom Data Company, 211 N. Kirby, Garland, TX 75042. 214/272-3421. Orders toll-free at 1-800-527-1592

WORD-M2, a word processor for the TRS-80 Mod-II, provides text formatting for letters, memos, reports, manuals and the like. Diskette and documentation costs $49 and requires a 64K Mod-II system. A similar version for the Mod-I is available for $49. Micro Architect, Inc., 96 Dothan St., Arlington, MA 02174.

SUPER-TEXT, a word processor for the Apple II and Apple II Plus, provides a complete catalog of text processing features. Cost is $99.95 from Muse Software, 330 N. Charles Street, Baltimore, MD 21201. 301/659-7212

CAIWARE and SUPER-CAI are software systems for authoring and using Computer Assisted Instruction on a 16K TRS-80 with Level 2 BASIC. CAIWARE on cassette with manual costs $24.95; SUPER-CAI with manual costs $44.95. The manuals are available separately for $9. Micro-Gnome, Fireside Computing, Inc., 5843 Montgomery Rd., Elkridge, MD 21227.

Electra Sketch is a graphics and animation compiler for the TRS-80. Pictures can be created using simple one key commands and saved on disk for later recall. Animation effects are obtained by displaying a sequence of stored frames on disk. The program is available on cassette for $14.95 from Macrotronics, 1125 N. Golden State Blvd., Suite G, Turlock, CA 95380.

Other

Remember NCC 1980 – The National Computer Conference and Personal Computing Exhibition – May 19-22, Anaheim Convention Center, Anaheim, CA. For information, contact AFIPS, 1815 N. Lynn Street, Arlington, VA 22209. 703/243-4100

Call for Computer Musicians and Artists to speak, exhibit, or perform at: PCAF '80, Personal Computer Arts Festival, Philadelphia, USA. August 23 and 24, 1980. A two day festival of talks and papers, films and graphics, demonstrations and performances. For information write to PCAF '80, c/o Philadelphia area Computer Society, Box 1954, Philadelphia, PA 19105.

Workshop: TRS-80 Interfacing and Programming for Instrumentation and Control. June 23-27, 1980. Contact Dr. Linda Leffel, CEC, Virginia Tech, Blacksburg, VA 24061. 703/961-5241.

Computer-Using Educators. The Santa Clara Valley Mathematics Association and the California Science Teachers Association are sponsoring a two-day conference in the Santa Clara Valley (California) on "Classroom Applications of Computers in Grades K-12," including tutorial sessions, workshops, and industrial exhibits of both hardware and software; additionally, excursions to 'Silicon Valley' industrial sites are planned.

The Conference will be held on September 26 & 27, 1980, a Friday and Saturday, at Independence High School, San Jose, California. Readers who would like more information may write to: Computer-Using Educators, c/o W. Don McKell, Independence High School, 1776 Educational Park Drive, San Jose, CA 95133.

The Future of Computing emphasizing microprocessors, small computers, home computers, and distributed processing is the subject of a one day ACM Golden Gate Chapter seminar to be given by Portia Isaacson and Dr. Egil Juliussen. May 10, 9am-4pm at the Hyatt Union Square, San Francisco. $40. For information contact Fred Collins at (415) 357-5272.

Tenth Annual Institute in Computer Science. Nine intensive computer short courses: Programming Methodology (Yeh), Programming Language Semantics (Arbib and Manes) Computer System Performance Prediction (Sevcik, Lazowska, Graham and Zahorjan), Design and Implementation of Modular Software (Guttag and Liskov), Operating Systems (Holt), Compiler Construction (DeRemerandPennello), Code Optimization (Ullman), Writing Workshop in Computer Science (Wilkes) and Formal Design of Computer Programs Extension, Carriage House, Santa (Sickel and Davis). For information contact University Extension, Carriage House, Santa Cruz, CA 95064.

Microcomputers and the Physicians Office. A two day seminar for physicians and medical office managers focusing on microcomputer applications. Saturday and Sunday, May 31 and June 1 at the Hyatt Regency in San Francisco. $250. Contact: Medical Data Systems, P. O. Box 193, Ojai, CA 93023. 805/646-8394

Data and Services

Micronotions, ATARI BASIC CARDS, an easy to understand reference for ATARI BASIC is now available. Each BASIC keyword has its own card, illustrating its correct syntax, briefly explaining its use, showing an example of it in use, and cross referencing related cards. The cards were designed by Len Lindsay especially for students. A FREE fanfold quick reference guide will be included with any order mentioning *Recreational Computing*. Introductory price for the ATARI BASIC CARDS is only $5.00 from Micronotions, 1929 Northport Drive, Room 6, Madison, WI 53704.

TRS-80* Yellow Pages 2.1 is a sixteen-page newsletter/catalog devoted entirely to serious business software. It is a handy and useful guide for selecting business software for the TRS-80 computers. It describes all the software produced by Micro Architect. It's distributed free (provided you send two long, stamped, self-addressed envelopes) by Micro Architect, 96 Dothan Street, Arlington, MA 02174.

A catalog of educational software is available from QUEUE, 5 Chapel Hill Drive, Fairfield, CT 06432. Offerings for the TRS-80, PET, and Apple computers are described.

A catalog listing a variety of field-tested educator-designed programs is available from Cook's Computer Company, 1905 Baily Drive, Marshalltown, IA 50158.

TRS-80 Software Duplication Special Limited Time Offer. Microsette Co. can now duplicate TRS-80 level II cassette programs to load over a volume setting from 4 to 8 or more on all Radio Shack cassette decks (CTR-40, CTR-41, CTR-80)! Even with the playback head purposely misaligned, these tapes will still load reliably.

Our charge for this service is $217.50 for 100 copies minimum prepaid. California customers please add sales tax unless resale number is submitted. Shipping is by UPS to one address in the United States. Microsette Co., 475 Ellis Street, Mountain View, CA 94043. 415/968-1604

Apple Animation

Continued from pg. 27

```
2005  IF X0 < 0 THEN X0 = 0
2010  IF X0 > 34 THEN X0 = 34
```

Just moving the car around on the screen isn't much fun, so let's add a moving road to test our driving skill:

```
1000  REM DRAW-ROAD SUBROUTINE
1005  R0 = R0 + RND(3) – 1
1010  IF R0 < 0 THEN R0 = 0
1015  IF R0 > 27 THEN R0 = 27
1020  COLOR = 13: REM YELLOW
1025  VLIN 0,39 AT R0
1030  VLIN 0,39 AT R0 + 12
1035  RETURN
```

As with the car, we limit the excursions of the road so it won't go off the screen.

Naturally, we also need a subroutine to erase the road:

```
4000  REM ERASE-ROAD SUBROUTINE
4005  COLOR = 15
4010  GOSUB 1025
4015  RETURN
```

We save a line of code by using part of the road-drawing subroutine, changing the color to white. Bit-pinching really isn't necessary in such a short program, but we do it anyway.

OK, now let's add a subroutine to tell us when we have gone off the edge of the road:

```
3000  REM ERROR-CHECK SUBROUTINE
3005  SPEED = 256 – PCL(1)
3010  FOR W = 1 TO SPEED: NEXT W
3015  IF X0 > R0 AND X0 < (R0 + 7) THEN
      RETURN
3020  PRINT "": REM CTRL-G
3025  ERR = ERR + 1
3030  RETURN
```

We input the speed from game paddle 1 and wait a while before doing the error check. If the car is off the road we beep the speaker and increment the error count. At the end of 250 laps (that is, program loops) we call it quits and print the number of errors.

The whole program is listed in Figure 2. Now that you see how easy animation is, you may want to try it yourself.

Figure 2. Program Listing

```
 55 REM  CAR ANIMATION
 60 REM
 65 CALL -936: REM  CLEAR SCREEN
 70 PRINT "STEER WITH PADDLE 0"
 75 PRINT
 80 PRINT "SET SPEED WITH PADDLE 1"
 85 PRINT
 90 INPUT "HIT RETURN TO BEGIN",R$
 95 R0=15:ERR=0:LAP=0
100 X0=17:Y0=16
105 GR : COLOR=15
110 FOR Y=0 TO 39
115 HLIN 0,39 AT Y
120 NEXT Y
125 Y0=Y0+ RND (3)-1
130 IF Y0<0 THEN Y0=0
135 IF Y0>32 THEN Y0=32
140 GOSUB 1000: REM  DRAW ROAD
145 GOSUB 2000: REM  DRAW CAR
150 GOSUB 3000: REM  CHECK ERROR
155 GOSUB 4000: REM  ERASE ROAD
160 GOSUB 5000: REM  ERASE CAR
165 LAP=LAP+1:X0= PDL (0)/5
170 IF LAP<250 THEN 125
175 TEXT : CALL -936
180 PRINT "YOU WENT OFF COURSE ";ERR;" TIMES --"
185 IF ERR>25 THEN 195
190 PRINT "A SUPERSTAR!": END
195 IF ERR>50 THEN 205
200 PRINT "A PROFESSIONAL!": END
205 IF ERR>100 THEN 215
210 PRINT "AN EXPERT!": END
215 IF ERR>200 THEN 225
220 PRINT "A NOVICE, EH?": END
225 PRINT "DRIVE MUCH?"
999 END
1000 REM  DRAW-ROAD SUBROUTINE
1005 R0=R0+ RND (3)-1
1010 IF R0<0 THEN R0=0
1015 IF R0>27 THEN R0=27
1020 COLOR=13: REM  YELLOW
1025 VLIN 0,39 AT R0
1030 VLIN 0,39 AT R0+12
1035 RETURN
2000 REM  DRAW-CAR SUBROUTINE
2005 IF X0<0 THEN X0=0
2010 IF X0>34 THEN X0=34
2015 REM  DRAW WHEELS
2020 COLOR=0: REM  BLACK
2025 VLIN Y0+1,Y0+7 AT X0
2030 VLIN Y0+1,Y0+7 AT X0+5
2035 COLOR=15: REM  WHITE
2040 HLIN X0,X0+5 AT Y0+4
2045 REM  DRAW AXLES
2050 COLOR=6: REM  GRAY
2055 HLIN X0+1,X0+4 AT Y0+2
2060 HLIN X0+1,X0+4 AT Y0+6
2065 REM  DRAW BODY
2070 COLOR=1: REM  MAGENTA
2075 VLIN Y0,Y0+7 AT X0+2
2080 VLIN Y0,Y0+7 AT X0+3
2085 RETURN
3000 REM  ERROR-CHECK SUBROUTINE
3005 SPEED=256- PDL (1)
3010 FOR W=1 TO SPEED: NEXT W
3015 IF X0>R0 AND X0<(R0+7) THEN RETURN
3020 PRINT "": REM  CTRL-G
3025 ERR=ERR+1
3030 RETURN
4000 REM  ERASE-ROAD SUBROUTINE
4005 COLOR=15
4010 GOSUB 1025
4015 RETURN
5000 REM  ERASE-CAR SUBROUTINE
5005 COLOR=15
5010 FOR X=X0 TO X0+5
5015 VLIN Y0,Y0+7 AT X
5020 NEXT X
5025 RETURN
```

THE FURTHER ADVENTURES OF

When last we left Our Hero, the potential of the great battle for the freedom of Microprocessorland had at last changed, as Linea forms the lead of her newly-arrived decades of resistance troops in its charging of the capacitive guards holding up the dielectric walls of Capital City, joining node-to-node with the networks of General Wirewound at the moment they are needed the most!

Still the charge barrier holds . . . that is, until Fortran Man, accompanied by his ally and friend Billy Basic and riding the back of the powerful Lockout Monster, flies over the besieged city and releases a vast barrage of . . . random passwords!

With their file-protect keys destroyed, the great data structures crumble, and the holdup potential of the walls is reduced . . . whereupon the current charge of the resistance causes the capacitive guards to break down . . . and the city is theirs!

And yet the war is not yet won . . . not until the source file of the conflict has been faced and eliminated . . . the Glitchmaster himself!

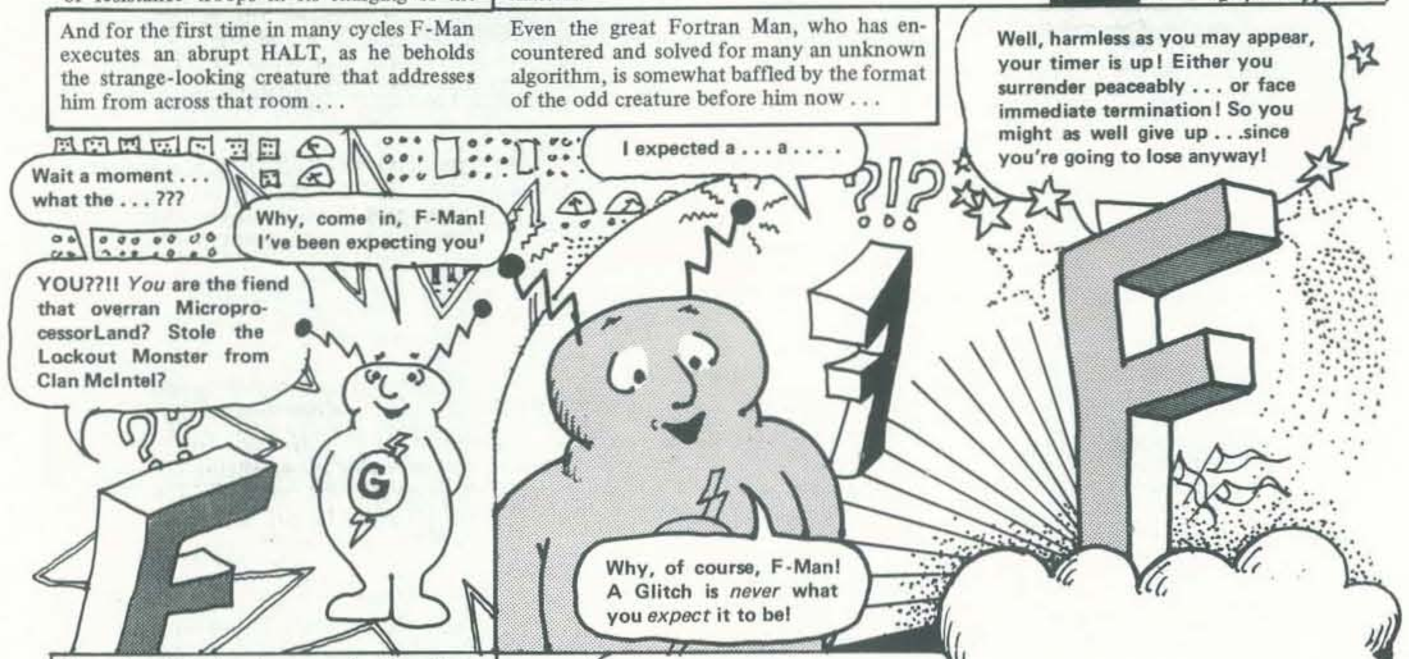
And there is only one among them with the strength and courage to face such a foe . . . and without PAUSE the ever-ready Fortran Man accepts the challenge!

On through the city his flowpath progresses, and into the dark hardware halls of the Glitchmaster's personal stronghold . . . Tesla Tower! His array-processing powers allow him to easily find his way through the matrix of corridors to the very portal of the Master Control Room! With an abrupt OPEN FILE statement, the port is activated . . . and then . . .

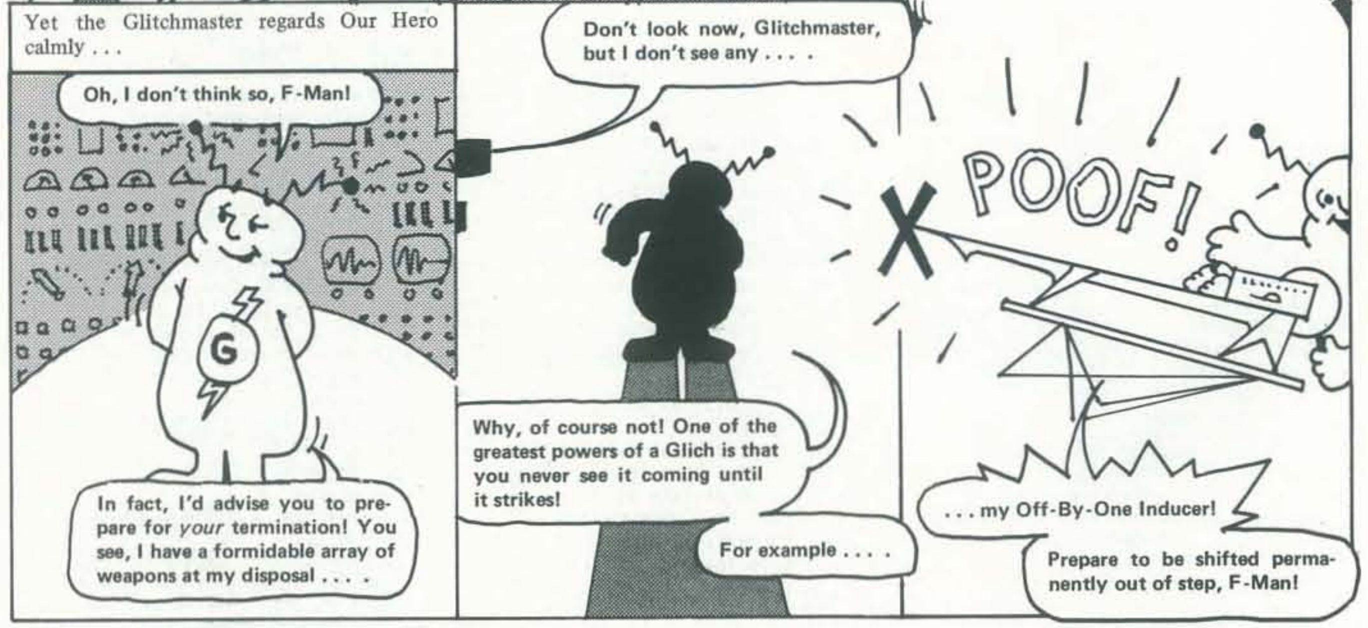
All right, Glitchmaster . . . come on out! Your statement number is up . . . prepare to be executed!
VOLUME 3 EPISODE II

And for the first time in many cycles F-Man executes an abrupt HALT, as he beholds the strange-looking creature that addresses him from across that room . . .

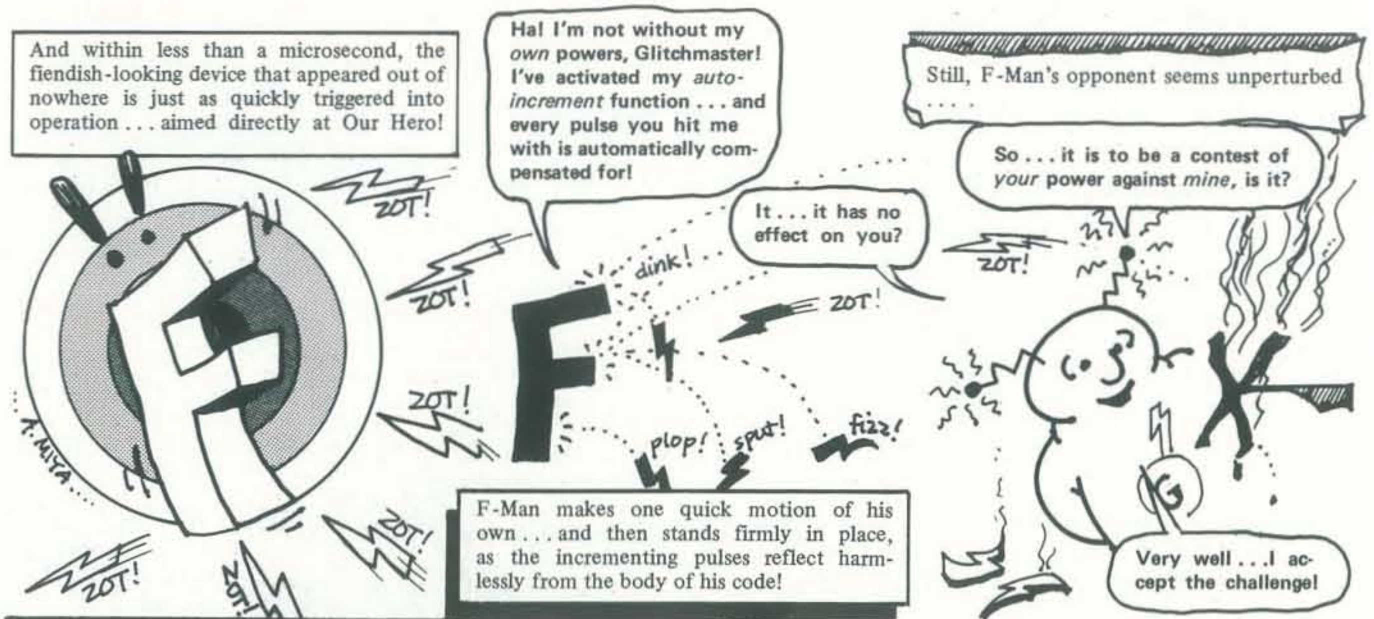
Even the great Fortran Man, who has encountered and solved for many an unknown algorithm, is somewhat baffled by the format of the odd creature before him now . . .


Wait a moment . . . what the . . . ???
Why, come in, F-Man! I've been expecting you!
YOU??!! You are the fiend that overran MicroprocessorLand? Stole the Lockout Monster from Clan McIntel?
I expected a . . . a . . . .
?!?
Why, of course, F-Man! A Glitch is never what you expect it to be!
Well, harmless as you may appear, your timer is up! Either you surrender peaceably . . . or face immediate termination! So you might as well give up . . . since you're going to lose anyway!

Yet the Glitchmaster regards Our Hero calmly . . .


Oh, I don't think so, F-Man!
In fact, I'd advise you to prepare for your termination! You see, I have a formidable array of weapons at my disposal . . . .
Don't look now, Glitchmaster, but I don't see any . . . .
Why, of course not! One of the greatest powers of a Glich is that you never see it coming until it strikes!
For example . . . .
POOF!
. . . my Off-By-One Inducer!
Prepare to be shifted permanently out of step, F-Man!

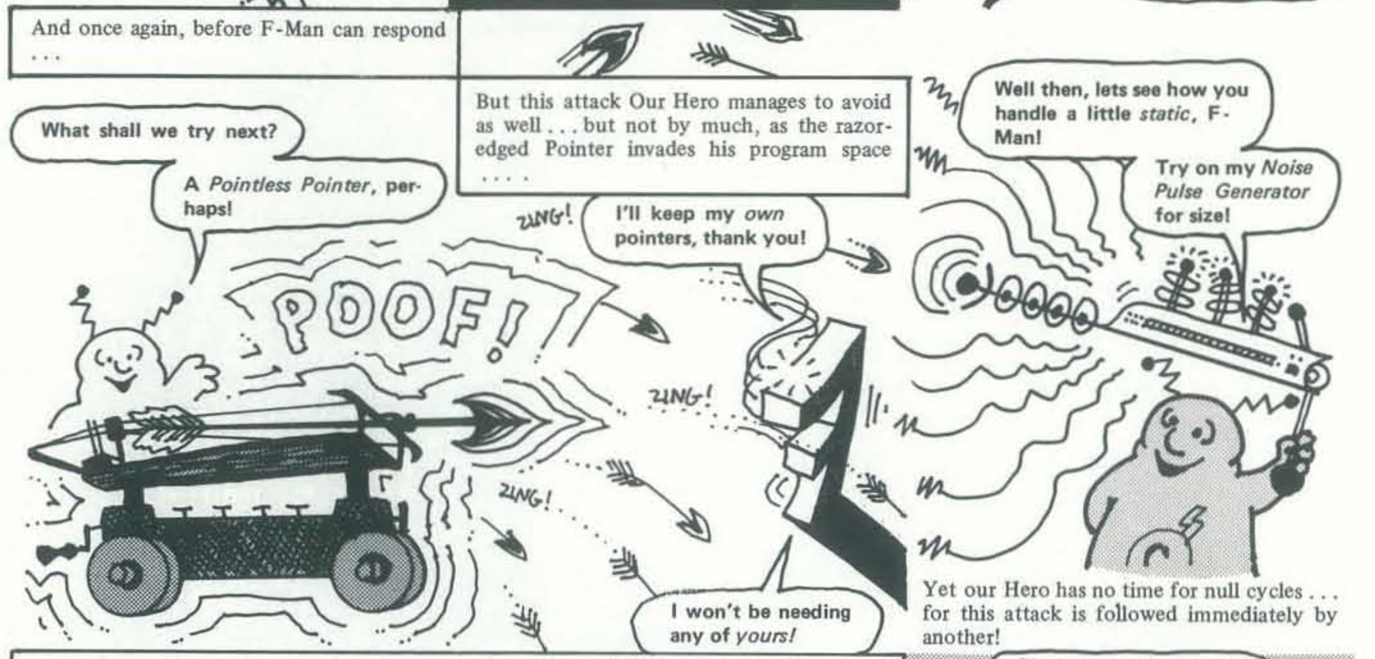
And within less than a microsecond, the fiendish-looking device that appeared out of nowhere is just as quickly triggered into operation . . . aimed directly at Our Hero!


ZOT!
Ha! I'm not without my own powers, Glitchmaster! I've activated my auto-increment function . . . and every pulse you hit me with is automatically compensated for!
It . . . it has no effect on you?
dink!
plop!
sput!
fizz!
So . . . it is to be a contest of your power against mine, is it?
Very well . . . I accept the challenge!

F-Man makes one quick motion of his own . . . and then stands firmly in place, as the incrementing pulses reflect harmlessly from the body of his code!

Still, F-Man's opponent seems unperturbed . . . .

And once again, before F-Man can respond . . .

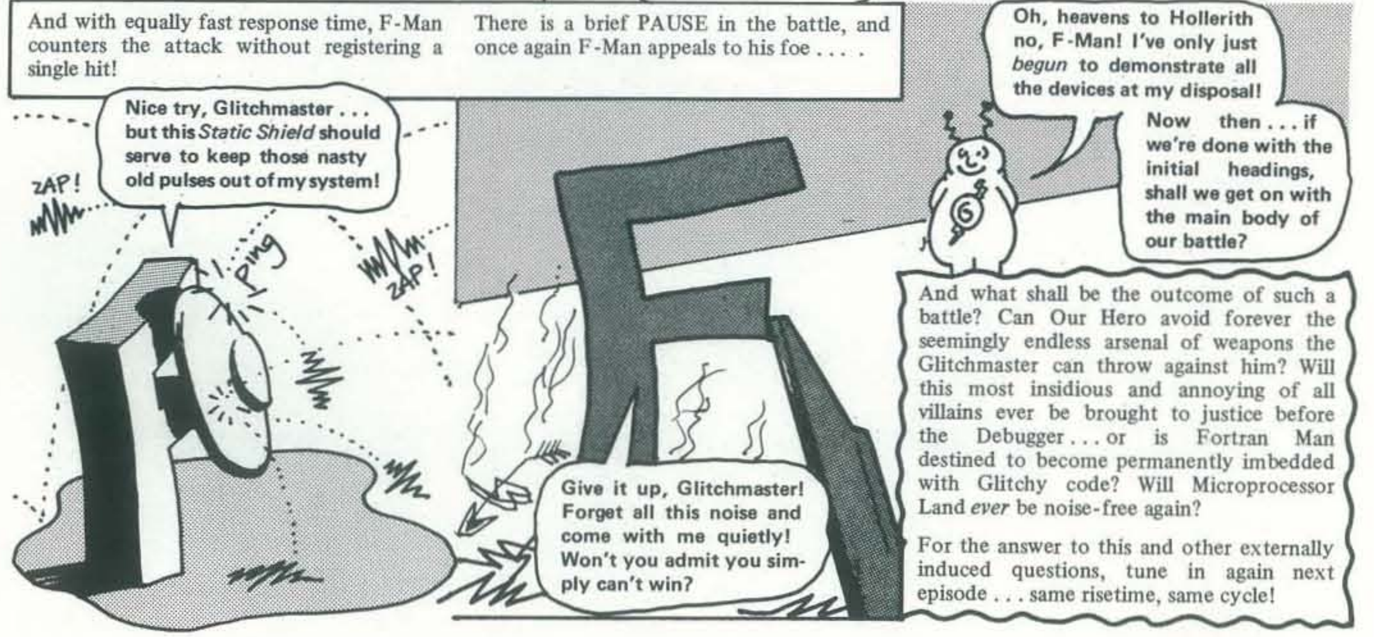

What shall we try next?
A Pointless Pointer, perhaps!
POOF!
ZING!
I'll keep my own pointers, thank you!
I won't be needing any of yours!
Well then, lets see how you handle a little static, F-Man!
Try on my Noise Pulse Generator for size!

But this attack Our Hero manages to avoid as well . . . but not by much, as the razor-edged Pointer invades his program space . . . .

Yet our Hero has no time for null cycles . . . for this attack is followed immediately by another!

And with equally fast response time, F-Man counters the attack without registering a single hit!

There is a brief PAUSE in the battle, and once again F-Man appeals to his foe . . . .


Nice try, Glitchmaster . . . but this Static Shield should serve to keep those nasty old pulses out of my system!
ZAP!
ping
ZAP!
Give it up, Glitchmaster! Forget all this noise and come with me quietly! Won't you admit you simply can't win?
Oh, heavens to Hollerith no, F-Man! I've only just begun to demonstrate all the devices at my disposal!
Now then . . . if we're done with the initial headings, shall we get on with the main body of our battle?

And what shall be the outcome of such a battle? Can Our Hero avoid forever the seemingly endless arsenal of weapons the Glitchmaster can throw against him? Will this most insidious and annoying of all villains ever be brought to justice before the Debugger . . . or is Fortran Man destined to become permanently imbedded with Glitchy code? Will Microprocessor Land *ever* be noise-free again?

For the answer to this and other externally induced questions, tune in again next episode . . . same risetime, same cycle!

A Proposed Graphics Language

BY JIM DAY

The display of high-resolution graphics is possible on many kinds of small computers. This is often done by adding special graphics commands to some popular language such as Basic. Graphics commands are generally rather primitive, allowing one to plot points, draw straight lines, and display predefined patterns. If an interpreter is used rather than a compiler, the speed of execution is usually too slow for satisfactory animation.

Because it takes time to change a graphic display, it's a good idea to alternate between two display areas in memory, so the old picture will remain on the screen until the new picture is ready for display. Another way of hiding display updates from the viewer is to blank the screen until the new picture is ready. This may produce flicker in the display but has the advantage of freeing the computer from the task of maintaining the display on the screen, which can easily consume more than half of the computer's time.

Suppose we had a really good graphics language at our disposal. Let's call it Really Advanced Pictorial Image Display, or RAPID for short. Let's assume that this is an interactive language that allows pictures to be defined manually via a mouse, stylus, or joystick as well as by source code. Let's further assume that the language processor is smart enough to translate a manually entered picture into the corresponding source code. It should also be possible to edit a picture online and have the processor modify the source code accordingly.

Ok, so what kind of graphics statements do we want RAPID to understand? To define, but not display, a dot at screen location X,Y we would like to write:

```
DOT FOO IS X,Y
```

FOO is the symbolic name assigned to the dot. To actually display the dot after it has been defined, we will use the statement:

```
DRAW FOO
```

In real life, one seldom wants to display a single isolated dot. Usually one deals with lines and groups of lines. So we need a line defining instruction:

```
LINE ZAP IS X1,Y1 TO X2,Y2
```

This defines a straight line, named ZAP, running from screen location X1,Y1 to location X2,Y2. Naturally, we expect RAPID to complain if either endpoint lies off the screen. If the display device uses raster scanning, most lines won't be perfectly straight but will be approximated by a series of line segments that are either horizontal or vertical. But let's not worry about that. To relocate line ZAP, we can write:

```
SHIFT ZAP BY X,Y
```

X and Y specify horizontal and vertical bias values for line ZAP. The length and direction of the line are unchanged. To rotate line ZAP about some point, not necessarily on the line itself, we can write:

```
ROTATE ZAP ABOUT X,Y BY A
```

X,Y defines the point about which ZAP is to be rotated and A is the angle of rotation, expressed in degrees. Rotation will be counterclockwise for positive values of A. Nothing happens on the screen until a DRAW instruction is executed:

```
DRAW ZAP
```

It would be nice to define a group of lines forming some figure such as a rectangle:

```
FIGURE FIG1 IS LIN1,LIN2,LIN3,LIN4
```

The arguments LIN1, LIN2, LIN3, and LIN4 are the names of previously defined lines. FIG1 is the name given to the figure comprising the four lines. We can then use FIG1 in other statements such as:

```
ROTATE FIG1 ABOUT X,Y BY A
```

An obvious extension of this is to allow figures to be composed of previously defined figures and/or lines:

```
FIGURE FIG2 IS FIG1,LIN5,LIN6
```

FIG2 is composed of FIG1 plus lines LIN5 and LIN6. To duplicate the shape of an existing figure, we can write:

```
FIG9 IS FIG7
```

The location of FIG9 can then be adjusted via a SHIFT or ROTATE statement. It would be useful to define circles:

```
CIRCLE C1 IS R AT X,Y
```

C1 is defined as a circle of radius R whose center is at location X,Y. Circular arcs are also useful:

```
ARC ZOT IS R AT X,Y FROM A1 TO A2
```

ZOT is a circular arc of radius R whose center is at X,Y. It extends from angle A1 to angle A2. To remove graphic elements from the screen we need a statement to do the inverse of what DRAW does:

```
ERASE FIG3,LIN4
```

This will erase all elements of FIG3 as well as LIN4. An ERASE statement with no arguments will erase the entire screen. If we want to make an element larger or smaller without changing the shape, we can use a SCALE statement:

```
SCALE FIG7 BY 2
```

This will double the size of FIG7, using the center of the figure as a fixed reference point. There should be a way of defining text strings:

```
TEXT TXT1 IS "FRONT VIEW" AT X,Y
```

This defines a text element named TXT1 comprising the string "FRONT VIEW" and having a starting location of X,Y. Once defined, a text element can be displayed via a DRAW statement or can be included as part of a figure:

```
FIGURE FIG7 IS FIG6,TXT1
```

If a figure is defined as including text, the text can be shifted, scaled, or rotated along with the rest of the figure:

```
SHIFT FIG7 BY X,Y
```

Animation can be accomplished by program loops containing appropriate DRAW, ERASE, and element redefinition statements.

BUY DIRECT FROM NCE/ CompuMart

Buy Direct from the Largest Commodore Dealer in the Country — NCE/CompuMart.

8K-Keyboard N	$795
16K-Keyboard B	$995
16K-Keyboard N	$995
32K-Keyboard B	$1,295
32K-Keyboard N	$1,295

B — large keyboard (graphics not on keys)
N — large keyboard with graphics symbols

SUPER SAVER. Thanks to Commodore their printer prices have been drastically reduced:

Commodore Printer (tractor feed) .... was $995 **NOW $795**
Commodore Printer (friction feed) .... was $849 **NOW $695**

2 Good reasons for buying your PET™ from CompuMart:

(1) $100 IN FREE ACCESSORIES WITH 16K or 32K PET.

When you buy a 16K or 32K PET, apply $100 toward PET accessories. FREE. Choose from the accessories listed below and indicate on your order that you have reduced the cost of your accessories by $100.

(2) FREE WITH PURCHASE OF ANY PET (Offer good until April 15, 1980)

A Texas Instruments' Hex/Octal Calculator ($60 Value), and a set of 6 PET Workbooks (Worth $36).

PET Accessories

Commodore Dual Floppy Disk Drive	$1,295.00
Second Cassette—from Commodore	$95.00
Commodore PET Service Kit	$30.00
Beeper—Tells when tape is loaded	$24.95
Petunia—Play music from PET	$29.95
Video Buffer—Attach another CRT	$29.95
Combo—Petunia abd Video Buffer	$49.95
TNW Bi-Dir.RS-232 printer S-face	$229.00
KIM 1 (A Single Board Computer from Commodore)	$179.00
PET TO IEEE Cable	$39.95
IEEE TO IEEE Cable	$49.95
KIM1 & Power Supply Package Special	$200.00

We have a complete inventory of Apple computers, peripherals, & software In-Stock for immediate delivery — Call us for prices.

$200 in FREE accessories with the purchase of a 48K Apple II reg. or Apple II plus.

16K Apple reg. or plus — $1,195 ($100 in *free* acces.)
32K Apple reg. or plus — $1,395 ($150 in *free* acces.)
48K Apple reg. or plus — $1,495 ($200 in *free* acces.)

Apple Accessories

PASCAL	$495
Micromodem	$379
VisiCalc	$150
The Controller (Business Package)	SALE! $550
The Cashier (POS System)	SALE! $200
Integer BASIC ROM Card	$200
Centronics Printer Interface	$225
Disk and Controller	$595
Parallel Printer Card	$180
Communications Card	$225
Hi-Speed Serial Card	$195
Firmware Card	$200
NEW! AppleWriter Text Editor	$75

NEW FROM EXIDY THE SORCERER II 48K COMPUTER

Z-80 Microprocessor, Full-sized keyboard • ROM PAC Programs • Microsoft BASIC • Powerful Graphics • Serial and Parallel I/O • Dual Cassette I/O Memory Internally Expandable • S-100 Expansion Options.
48K Sorcerer II **$1,395**

ZENITH

The All-In-One Computer

Save $300 off list on all Zenith Computers, $250 off the printer. Build a great system using both and save $550.

WH89 with 16K RAM	**list $2,295**	**SALE! $1,995**
WH89 with 32K RAM	**list $2,445**	**SALE! $2,145**
WH89 with 48K RAM	**list $2,595**	**SALE! $2,295**
Heath Printer	**list $895**	**SALE! $650**
16K Memory Expansion	**list $150**	**SALE! $110**
Dual-port Serial Interface		**$85**
HD05 Operating System		**$100**
Microsoft Basic		**$100**
WH17 Second Disk Drive		**$550**

Only limited quantities of the Computers and Printers are available. Order Now. (Offer good until April 15, 1980)

CENTRONICS PRINTERS

The 779-2 is a tractor feed printer designed for small business systems. Uses a 5x7 dot matrix to produce all 64 upper case ASCII characters. Centronics is the biggest name in printers and this is their most popular model.
List. $1,245 .... SALE! $1,079

The New 730-1 parallel matrix printer is ideally suited for all personal and microcomputer systems. Standard Features include: 100 cps • 80 char./line • 3-way paper handling system • 7x7 dot matrix • 96 character ASCII • List $995 .... SALE! $839

The Perfect Printer for Small Business Systems

The Paper Tiger Printer From ROAR! Integral Data

Standard features include: 4 character 8.3 to 16.5 cpi • 56 cps at 10 char. per in. • Selectable line spacing • 8 switch-selectable form sizes.

The IDS Graphics Option for the Paper ~~Tiger~~ $695 allows full dot pattern control and includes an expanded 2048-byte buffer (a 256-byte buffer is standard).

IDS Paper Tiger Printer $995
IDS Graphics Paper Tiger Printer $1,094

SPECIAL OFFER EXCLUSIVE FROM COMPUMART!

ZENITH COLOR VIDEO MONITOR

Beautiful Display Capabilities. Excellent for use with Apple, Atari, & Sorcerer Computers. This 13-inch monitor is Zenith's first color video display designed specifically for computers. Features include color and degaussing circuits.
Zenith Color Monitor $499.00

HAZELTINE TERMINALS AT SPECTACULAR SAVINGS!

Hazeltine 1410-TTY-style keyboard, 8 baud rates from 110 to 9600, 12" screen, 24x80 display, 5x7 dot Matrix Upper Case ASCII character set, 12-key numeric keypad. List Price $895 ......**SALE! $779**
Hazeltine 1500 — 8 baud rates from 110 to 19,200, and ANSI Standard Keyboard, EIA RS-232, All 94 Upper/Lower Case characters in a high resolution 7x10 matrix display. Much more.
List Price $1,145 ....**SALE! $995**
Hazeltine 1520 — Take advantage of this $310 savings while it lasts! The Hazeltine 1520 is a powerful multi-processor terminal.
List Price $1,585 .... **SALE! $1,275**

NOVATION CAT™ ACCOUSTIC MODEM

- Answer Originate
- Bell 108
- 300 Baud
- Low Profile Design

Looks good, works great! **$179.00**

BRAND NEW! The "Star" Modem from Livermore
Special Introductory Price $158.00

SANYO MONITORS

9-inch reg. $199 **SALE! $169**
15-inch reg. $299 **SALE! $269**

Perfect for Apple users . . .
Sanyo Cassette Recorder **$55**

REMOTE TERMINAL FOR YOUR PET
Only $98

A self-contained module and program cassette enables your PET to function as a 300 baud terminal. Supports Upper/Lower case. Rubout. Escape and all control functions. Output is TTL. Can be used with THE SOURCE. You will need the Terminal Option, Cat Coupler & EIA Cable to do this. Complete package which includes all 3 available from CompuMart for $279.

NCE/CompuMart *We've Had a Reputation for Dependability Since 1971.*

DEPT. RC50, 270 THIRD ST., CAMBRIDGE, MA. 02142

To Order: 1 (800) 343-5504
In Mass: 1 (617) 491-2700

VISA master charge

Member Computer Dealers Assoc.

10-DAY FREE TRIAL

SEND FOR OUR FREE CATALOG

Lear Siegler Terminals & Printers At *Tremendous Savings*

ADM-3A. Industries' favorite dumb terminal for some very smart reasons. 12" diagonal screen • Full or half duplex operation at 11 selectable data rates • 1,920 easy-to-read characters in 24 rows of 80 letters • RS-232C interface extension port • Direct cursor addressing.
Reg. Price. $895 .... **Sale! $795**

ADM-31. A terminal that's too smart to be considered dumb. Comes complete with keyboard, control logic, character generator, refresh memory & interface • Displays two pages of text instead of one • Field protect mode • Factory installed selected parameters.
Reg. Price. $1,450 .... **Sale! $1,195**

ADM-42. The semi-intelligent terminal that provides you with flexibility of format, security, editing, interface, and transmission. Two-page display standard (Optionally expandable to eight) • Blanking, blinking, and reverse fields • Three ways to TAB • 16 Function keys do the work of 32 • Detachable keyboard.

ADM-42 with Keyboard. Reg. Price. $1,795 **Sale! $1,595**
ADM-42 without keyboard. Reg. Price. $1,595 .... **Sale! $1,450**

300 Series Ballistic Printers. This application-oriented matrix printer is built to last. Features include: Ballistic Printing • Positive 180 cps • Bi-directional Printing • A character buffer that is optionally expandable to 2,048 characters • 512 character buffer standard.

Model 310 Ballistic Printer (Serial/Parallel)
Reg. Price. $2,045 .... **Sale $1,989**

We Stock Lear Siegler Accessories — Call For Details.

THE SINGLE BOARD DEVELOPMENT SYSTEM ROCKWELL AIM 65

6502 Microprocessor • 20-character, alpha-numeric LED display • Full-size 54-key keyboard with 3 user-defined functions • Fast, on-board 20-column thermal printer • 8K Advanced Interactive Monitor program • Dual cassette interface board • On-board timer • On-board ROM expansion to 12K • 4K on-board RAM • On-board TTY interface • 16 parallel 1/0 lines • One serial 1/0 port • KIM-compatible edge connectors for even further memory or 1/0 expansion.
The CompuMart AIM System combines all of our options for the AIM to give you the capabilities of development systems costing 5 to 10 times as much. This system includes a 4K AIM 65 with BASIC and Assembler, an MTU power supply, a Sanyo tape recorder and an EGI Enclosure for the AIM.

CompuMart AIM System	**$785.00**
4K AIM-65	**$450.00**
Paper for the AIM	**$4.75**

WE'VE GOT IT!

A Calculator, A system, A Whole New Standard.
The finest calculators available for Science, Engineering, & Business.

HEWLETT-PACKARD'S HP-41C

HP-41C

The Calculator.
Features over 130 functions and offers up to 400 lines of program memory or 63 data storage registers — expandable to 319 registers or up to 2,000 lines. RPN Logic. Alpha-numeric capabilities let you communicate with the calculator in English. Customization features allow you to totally reassign the keyboard functions. Continuous memory.......**HP-41C Calculator $288.00**

The System.
Memory Modules. For storing programs or up to 2,000 lines of program memory. **$45.00**
"Extra Smart" Card Reader. Records programs and data back onto blank mag-cards. **$179.00**
The Printer. Upper and Lower case. High resolution plotting. Portable. Thermal operation. **$320.00**
Application Modules. .... **$45.00 EACH.**
Standard pac, Statistics, Math, Financial, & Surveying.

HP SERIES E CALCULATORS

HP-31E — Scientific. Trigonometric, exponential & math functions. **$49.95**
HP-32E — Advanced Scientific with Statistics. All HP-31E functions plus hyperbolics and comprehensive statistics. **$66.95**
HP-33E — Programmable Scientific. A programmable science, math and statistical calculator. **$79.95**
Also available with continuous memory....**86.40**
HP-37E — Business Management. Best choice for a business/finance calculator........ **$69.95**

HP-38E — Advanced Financial with Programmability. All the features of the HP-37E plus a lot more power. **$108.00**
Also available with continuous memory....... **$144.95**
THE ALL NEW HP-34C. Advanced Continuous Memory Scientific Programmable with an impressive array of programming features. **$144.95**

CompuMart STOCKS THE COMPLETE LINE OF MATROX PRODUCTS. CALL FOR SPECS.

Based on our policy of offering our customers only the finest in microcomputers, CompuMart is pleased to announce that we now carry the new generation of Personal Computers by Atari.™

INTRODUCTORY SPECIALS

To celebrate our commitment to Atari, we are offering the following Atari Specials:
(1) **Buy additional memory for your computer, 8K or 16K, and we will double the amount of memory FREE!**
(A potential savings of $200.)
(2) **Buy the Atari 800 Computer and take $100 off the purchase price of the Atari 810 Disk Drive or the Atari 820 printer.**

ATARI® 800™
PERSONAL COMPUTER SYSTEM

Comes with:
- **Computer Console**
- **BASIC Language Cartridge**
- **Education System Master Cartridge**
- **BASIC Language Programming Manual**
- **800 Operator's Manual**
- **ATARI 410 Program Recorder**
- **Invitation to Programming™ Cassette**
- **8K RAM Memory Module™**
- **10K ROM Operating System**
- **Power Supply**
- **TV Switch Box**

SPECIFICATIONS:

High resolution color graphics
57 key full stroke keyboard
Built-in RF modulator for channel 2/3 operation with standard TV set
Composite video output for use with monitor
Internal Speaker
Two cartridge slots for rapid program insertion
Four internal slots for expansion up to 48K RAM
6502B Microprocessor
High speed serial 1/0 port

Atari 800 Computer System $995.95

ATARI® 820™ PRINTER

High resolution dot matrix impact printer
Uses standard ⅜ inch roll paper and ribbon
40 characters per line
Speed: 40 characters per second
UL approved

Atari 920 Printer $599.95

ATARI® 810™ DISK DRIVE

Uses standard 5¼ inch diskettes
88K bytes storage per diskette
Up to four disk drive units can operate with the system
Average data access time: 236 milliseconds
Power: AC adapter; UL approved

Atari 810 Disk Drive $699.95

Upgrade your computer with additional memory. (Note that the Atari 800 Computer comes with 8K of RAM memory and will accept up to 48K.)

Atari 8K RAM Memory Module $124.95
Atari 16K RAM Memory Module $199.95

NCE/CompuMart

Dept. RC50,
270 Third St.,
Cambridge, Mass.
02142

TO ORDER CALL:
1-800-343-5504
IN MASS. CALL: (617) 491-2700

IMPORTANT ORDERING INFORMATION
All orders must include 4% shipping and handling. Mass. residents add 5% sales tax; Mich. residents 4%, for sales tax.

Phones open from 8:30 a.m. to 5:30 p.m. EST, Mon.-Fri. • P.O.'s accepted from D&B rated companies—shipment contingent upon receipt of signed purchase order • All prices are subject to change without notice • Most items in stock for immediate shipment—call for delivery quotation • In the Ann Arbor area? Our retail store is open 11:00 a.m. to 7:00 p.m. Tues.-Fri., 10:00 a.m. to 5:00 p.m. Saturdays (closed Sun. and Mon.)

****IF NOT SATISFIED RETURN PURCHASE WITHIN 10 DAYS FOR A FULL REFUND.***