Sprite Graphics And Sound Synthesis On The Commodore 64

COMPUTE!

$2.50
September
1982
Issue 28
Vol. 4, No. 9
63379 £1.85 in UK

The Journal For Progressive Computing™

The Sounds Of TRON

Using Apple and Atari In The Production Of A Computer Fantasy

A Turtle PILOT Compiler In BASIC For Personal Computers
Part 1: The Apple Version

Automatic Disk Boot For Commodore Computers

Joystick Tutorial For The VIC-20

An Atari Rainbow
Three GTIA Graphics Articles

Graphic Display For The Radio Shack Color Computer
Extending The Utility Of The Energy Monitor Program

09
0 74470 63379

REVIEWS: Space Games For VIC And Atari, PET/CBM Data Base Manager And More.

THE LEADING EDGE IN PRINTERS

ONE GREAT LINE. ONE GREAT WARRANTY.

Finally, there's one full family of printers that covers every business or word processing application—all from C. Itoh, a company known for packing more product into less price; and all distributed exclusively by Leading Edge, a company known for searching out and providing that very thing. Which means that one call to one source can get you any printer, any time you need it, for any purpose. All backed by a full years' warranty from Leading Edge. (Try *that* on any other line of printers.)

THE PRO'S.

The Prowriters: business printers—and more. The "more" is a dot-matrix process with more dots. It gives you denser, correspondence quality copy (as opposed to business quality copy, which looks like a bad job of spray-painting).

Prowriter: 120 cps. 80 columns dot matrix compressable to 136. 10" carriage. Parallel or serial interface.

Prowriter 2: Same as Prowriter, except 15" carriage allows full 136 columns in normal print mode. Parallel or serial interface.

THE STAR.

The Starwriter F-10. In short (or more precisely, in a sleek 6" high, 30-pound unit), it gives you more of just about everything—except bulk and noise—than any other printer in its price range. It's a 40 cps letter-quality daisy-wheel with a bunch of built-in functions to simplify and speed up word processing. It plugs into almost any micro on the market, serial or parallel.

THE MASTER.

The Printmaster F-10. Does all the same good stuff as the Starwriter except, at 55 cps, the Master does it faster.

Distributed Exclusively by Leading Edge Products, Inc., 225 Turnpike Street, Canton, Massachusetts 02021. Call: toll-free 1-800-343-6833; or in Massachusetts call collect (617) 828-8150. Telex 951-624.

IF YOU'RE WAITING FOR THE PRICE OF WORD PROCESSORS TO FALL WITHIN REASON,

Everyone expected it would happen sooner or later... with WordPro PLUS™ it already has! Now all the marvelous benefits of expensive and advanced word processing systems are available on Commodore computers, America's largest selling computer line. WordPro PLUS, when combined with the new 80 column CBM 8032, creates a word processing system comparable to virtually any other top quality word processor available—but at savings of thousands of dollars!

New, low cost computer technology is now available at a fraction of what you would expect to pay. This technology allowed Commodore to introduce the new and revolutionary CBM 8032 Computer.

WordPro PLUS turns this new CBM 8032 Computer into a sophisticated, time saving word processing tool. With WordPro PLUS, documents are displayed on the computer's screen. Editing and last minute revisions are simple and easy. No more lengthy re-typing sessions. Letters and documents are easily re-called from memory storage for editing or printing with final drafts printed perfectly at over five hundred words per minute!

Our nationwide team of professional dealers will show you how your office will benefit by using WordPro PLUS. At a price far less than you realize.

Invest in your office's future...
Invest in WordPro PLUS...
Call us today for the name of the WordPro PLUS dealer nearest you.

Professional Software Inc.
51 Fremont Street
Needham, MA 02194
(617) 444-5224
TELEX: 95 1579

TM WordPro is a Registered Trademark of Professional Software, Inc. WordPro was written by Steve Punter.
All specifications subject to change without notice.

We could tell you we make great games. We don't have to.

All for the Apple and Atari on diskette.

You keep Brøderbund at the top of the best seller lists, so you must know what it takes to make a great game. We think our APPLE and ATARI games are uniquely challenging with action and graphics second to none. But you keep asking for Brøderbund games because they're just plain fun.

Enjoy the magic of our growing collection. Lead a daring rescue mission in CHOPLIFTER. Or survive the maze and monsters in slithery SERPENTINE and LABYRINTH. Blast through the future with STAR BLAZER or conquer the ultimate pinball game with DAVID'S MIDNIGHT MAGIC. Take up train robbing in TRACK ATTACK or tense your mind and reflexes in DUELING DIGITS.

Brøderbund

Brøderbund products are available at your retailer or by writing to:
Brøderbund Software, Inc., 1938 Fourth Street, San Rafael, CA 94901 • Tel: (415) 456-6424

Apple II is a registered trademark of Apple Computer, Inc. Atari is a registered trademark of Atari, Inc.

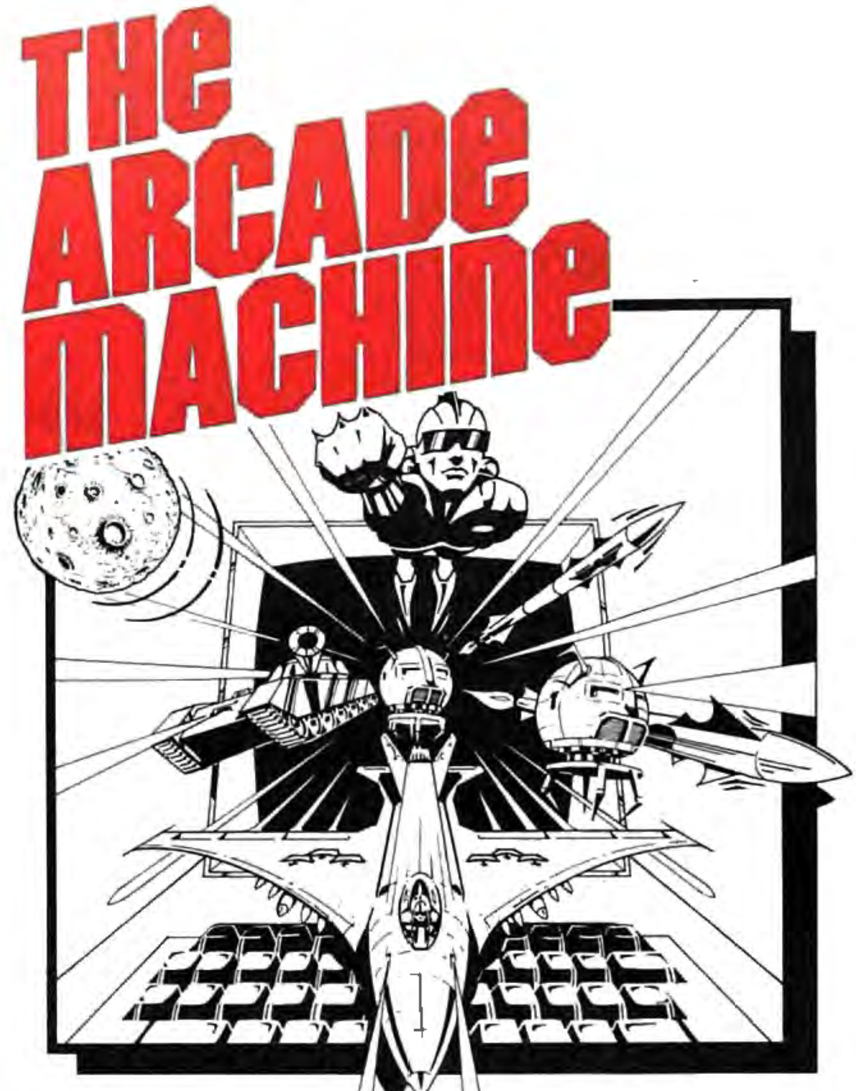
the finest computer games. You keep telling us.

SEAFOX
Sub against the convoy
Apple & Atari disk

DEADLY SECRETS
Hi-res adventure
Apple disk

APPLE PANIC
The arcade classic
Apple disk, Atari cassette & disk

STELLAR SHUTTLE
Monsters and meteors
Atari cassette & disk



PUT US OUT OF BUSINESS! THE ARCADE MACHINE lets you design and produce your own computer games, without any programming knowledge! Send us your best game and enter the BRØDERBUND ARCADE MACHINE CONTEST. We'll be giving away thousands of dollars worth of hardware and software in prizes. If you have a creative touch and an artistic eye, you too can be an arcade designer. Write to Brøderbund for contest details or visit your participating retail store! (Available on Apple disk.)

Software

All Brøderbund games are fully guaranteed. If they ever fail to boot, return the original disk to Brøderbund for a free replacement. If you have physically damaged the disk please include five dollars for replacement.

WHY THE MICROSOFT RAMCARD™ MAKES OUR SOFTCARD™ AN EVEN BETTER IDEA.

Memory — you never seem to have quite enough of it.

But if you're one of the thousands of Apple owners using the SoftCard, there's an economical new way to expand your memory dramatically.

16K ON A PLUG-IN CARD.

Microsoft's new RAMCard simply plugs into your Apple II,® and adds 16k bytes of dependable, buffered read/write storage.

Together with the SoftCard, the RAMCard gives you a 56k CP/M® system that's big enough to take on all kinds of chores that would never fit before (until now, the only way to get this much memory was to have an Apple Language Card installed).

GREAT SOFTWARE: YOURS, OURS, OR THEIRS.

With the RAMCard and SoftCard, you can tackle large-scale business and scientific computing with our COBOL and FORTRAN languages. Or greatly increase the capability of CP/M applications like the Peachtree Software accounting systems. VisiCalc™ and other Apple software packages can take advantage of RAMCard too.

And RAMCard gives you the extra capacity to develop advanced programs of your own, using the SoftCard and CP/M. *Even with the RAMCard in place, you can still access your ROM BASIC and monitor routines.*

JOIN THE SOFTCARD FAMILY.

The RAMCard is just the latest addition to the SoftCard family — a comprehensive system of hardware and software that can make your Apple more versatile and powerful than you ever imagined.

Your Microsoft dealer has all the exciting details. Visit him soon, and discover a great idea that keeps getting better.

Microsoft Consumer Products, 400 108th Ave. N.E., Suite 200, Bellevue, WA 98004. (206) 454-1315.

MICROSOFT

Table of Contents

September 1982 Vol. 4, No. 9

AP = Apple, **AT** = Atari, **P** = PET/CBM, **V** = VIC-20, **O** = OSI, **C** = Radio Shack Color Computer, * = All or several of the above.

COMPUTE! The Journal for Progressive Computing (USPS: 537250) is published 12 times each year by Small System Services, Inc., P.O. Box 5406, Greensboro, NC 27403 USA. Phone: (919)275-9809. Editorial Offices are located at 625 Fulton Street, Greensboro, NC 27403. Domestic Subscriptions: 12 issues, $20.00. Send subscription orders or change of address (P.O. form 3579) to Circulation Dept., **COMPUTE!** Magazine, P.O. Box 5406, Greensboro, NC 27403. Second class postage paid at Greensboro, NC 27403 and additional mailing offices.

ISSN 0194-357X.

The Editor's notes...

Robert Lock, Publisher/Editor-In-Chief

A Very Personal Aside: 100,000 Here We Come!

Those of you who remember when we broke through 10,000 paid circulation will be excited to know that we are rapidly approaching the 100,000 mark. Our October press run will exceed 100,000 for the first time in our history, and we forecast breaking 100,000 paid in November.

Practically speaking, that 5,000 magazine growth from 95,000 to 100,000 is no different than that same amount of growth from 90,000 to 95,000. Emotionally it's glorious, and we're rather beside ourselves with excitement. Thank you all for your continued input and support. If you should happen to call in early September and hear a bit of background uproar, you'll understand why.

The Commodore 64

As you read this, the first production line Commodore 64's should be appearing in your local stores. We know they exist. Our Features Editor, Tom Halfhill, spent an entire day working with one at Commodore's Valley Forge, PA headquarters to produce the article in this issue. Rumor has it that outside software vendors have been working hard on an impressive array of support software for this new arrival from Commodore.

The Mass Market Micros

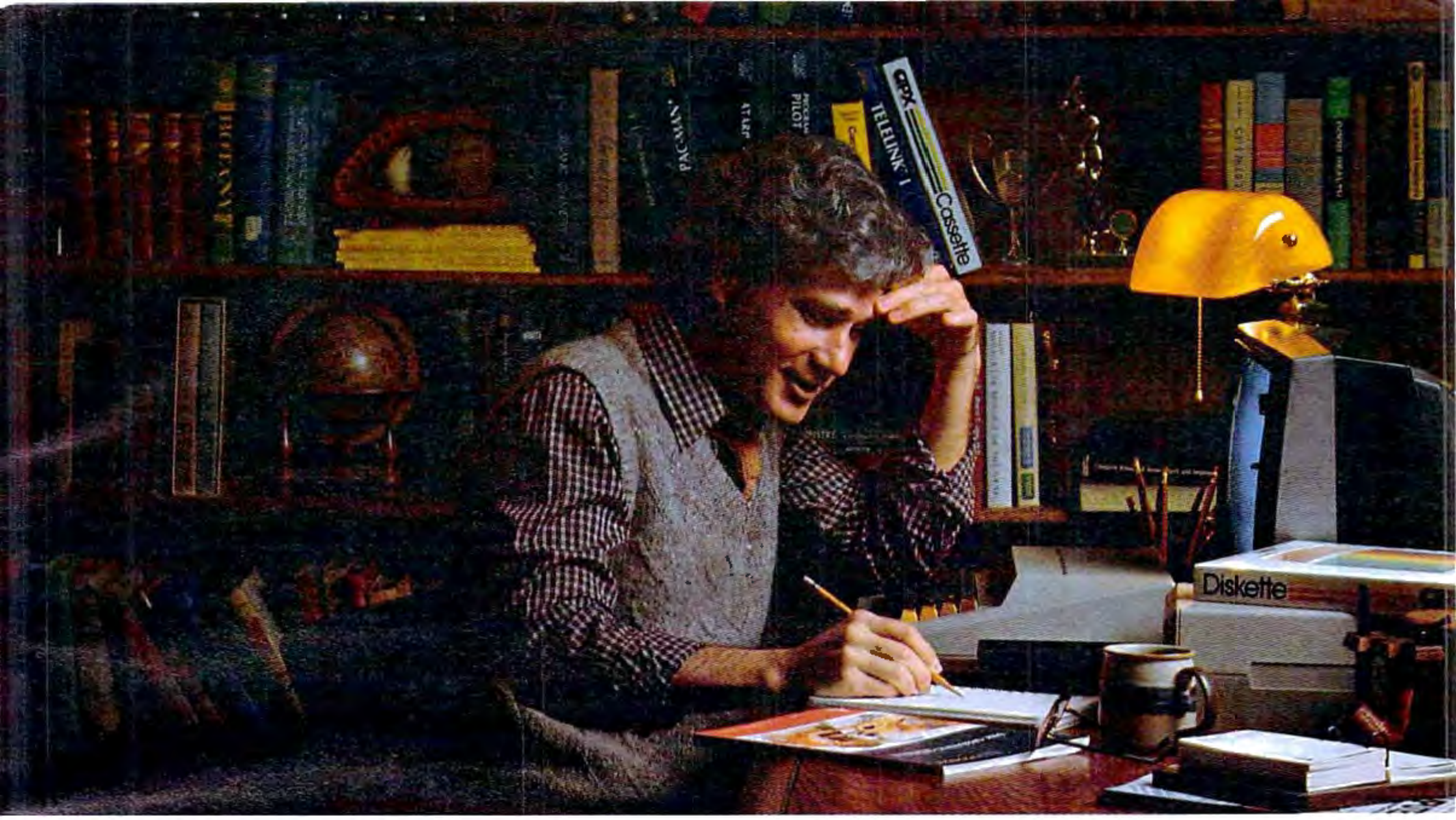
The competition continues: Atari has dropped the base price of the Atari 400 to $299.95 and entered a major marketing agreement with Sears. K mart has announced the doubling of the number of stores carrying the Commodore VIC-20. You'll find VIC's in 1100 K mart locations now. K mart is selling the Texas Instruments personal computer as well. Commodore also is selling VIC-20's through Montgomery Ward stores. Radio Shack/Tandy has decided to drop its traditional approach to the consumer market, and expand the potential customer base for their Color Computer. They will be selling a version of it (different color case) through department stores, etc.

Our sources indicate that between the two of them, Atari and Commodore are currently shipping over 60,000 VIC-20's and Atari 400's a month. By the time you factor in the Atari 800, the Commodore 64, Texas Instruments, Sinclair, and the Radio Shack Color Computer, you can conservatively estimate 200,000 consumer computers a month by October. Imagine... there were some who thought we'd go the way of the CB radio.

The October Issue: Games

Michael Day combines telecommunications with games for an exciting column on telegaming; Bill Wilkinson articulates machine language gaming on the Atari; Tom Halfhill interviews industry leaders on the future of games. You'll find all of this and much more in our very special October theme issue, including some exciting games ready to type into your computer.

Our July issue featured an action game and accompanying article entitled "Gold Rush!" This game should not be confused with an arcade-graphics game for both the Apple and Atari of the same name produced and marketed by Sentient Software of Aspen, Colorado. No comparison or confusion was intended regarding the products of Sentient Software, and readers should be aware that these two games are entirely different products. In any future use of this article and action game, we will refer to it as "Gold Miner." ©



MORE THAN EVER, ATARI HOME COMPUTERS ARE SPEAKING YOUR LANGUAGE.

With more program languages than ever to choose from, you now have more opportunities than ever to utilize the amazing capabilities of the ATARI 800™ Home Computer.

Whether you're a beginning programmer, or at the forefront of the art, you'll find an ATARI programming language that can make your task quicker and simpler than ever before. Just look at what ATARI has to offer:

ATARI Microsoft BASIC – Now we offer the industry standard, the most powerful Microsoft BASIC yet. With simple commands, it allows you to take advantage of unique ATARI hardware features such as our well-known player/missile graphics. For ease of programming, it includes n-dimensional arrays for numerics and string handling. And importantly, conversion procedures are simple.

ATARI Macro Assembler – Faster and more powerful than any ATARI language before, the ATARI Macro Assembler also allows you to access more memory space. And it's excellent for I/O interface and manipulation of such features as: player/missile graphics, sound registers and peripherals. In addition, the macro processor and "include" file library features speed-up program development considerably.

Fig-FORTH* – For specialized programming needs, such as educational or game applications, ATARI Fig-FORTH is uniquely effective. Fig-FORTH combines power and simplicity in an efficient 10K size, with characteristics of an interpreter and the speed of machine language code.

ATARI BASIC – An affordable and easy to use BASIC that requires only 8K of memory. It allows you to take advantage of the spectacular ATARI graphics and sound capabilities. And its immediate mode error messages greatly simplify debugging.

ATARI Assembler Editor – An excellent tool to assist the assembler-programmer in creating, editing and debugging assembly programs.

PILOT – ATARI PILOT is an exceptional learning language, with built-in "turtle" graphics to let you create spectacular designs and pictures with very short programs. Simple one or two-letter commands allow you to create a dialogue with the computer. And a single "match" command can perform complex text evaluation and pattern-matching instantly.

ATARI Pascal* – An excellent high-level language for teaching structured programming, and for developing and maintaining programs. In addition to offering all the features of the ISO Pascal standard, ATARI Pascal offers unique extensions that allow you to take advantage of ATARI graphics and sound capabilities.

ATARI is constantly developing new ways to help you get more out of your ATARI 800 Home Computer. So watch for more innovative and exciting programming languages from ATARI in the future.

For more information, write to ATARI, Inc.,
Dept. C4Z, P.O. Box 16525,
Denver, CO 80216.

ATARI

*Available from the ATARI Program Exchange

ATARI HOME COMPUTERS

We've Brought The Computer Age Home.™

Subscription Information

COMPUTE! Circulation Dept.
P.O. Box 5406
Greensboro, NC 27403

TOLL FREE
Subscription
Order Line
800-334-0868
In NC 919-275-9809

COMPUTE! Subscription Rates

(12 Issue Year):

US	(one yr.)	$20
	(two yrs.)	$36
	(three yrs.)	$54
Canada and Foreign Surface Mail		$25
Air		
Europe, US Possessions		$38
Middle East, Central America and North Africa		$48
South America, South Africa, Far East and Australia		$88

Small System Services, Inc. publishes:

COMPUTE!
The Journal For Progressive Computing

COMPUTE! Books

Corporate office:
625 Fulton Street,
Greensboro, NC 27403 USA

Mailing address: COMPUTE!
Post Office Box 5406
Greensboro, NC 27403 USA

Telephone: 919-275-9809

In British Columbia, Canada call 408-354-5553.

Elsewhere in Canada or outside North America call 919-275-9809.

In Alaska or Hawaii call 408-354-5553.

Phoebe Thompson and Associates
101 Church Street, Suite 13
Los Gatos, CA 95030
PHOEBE THOMPSON

Phoebe Thompson and Associates
2556 Via Tejon
Palos Verdes Estates, CA 90274
JOANN SULLIVAN

GB & Associates
P.O. Box 335
Libertyville, IL 60048
GORDON BENSON

COMPUTE! Home Office
625 Fulton Street
Greensboro, NC 27403
HARRY BLAIR

The Gittelman Company
Statler Office Building
Suite 518
20 Providence Street
Boston, MA 02116
AL DALTON

The Gittelman Company
Summit Office Center
7266 Summit Avenue
Fort Washington, PA 19034
ANDY MEEHAN

Address all advertising materials to:
Alice S. Wolfe
Advertising Production Coordinator
COMPUTE! Magazine
625 Fulton Street
Greensboro, NC 27403 USA

Coming In The Special October Games Issue Of COMPUTE!

National Public Radio: Computer Programs Via FM

Future Games

Superchase

Space Math

Meteor Storm

Laser Barrage

and much more...

ADD POWER™ TO YOUR COMMODORE COMPUTER

$89.95

POWER produces a dramatic improvement in the ease of editing BASIC on Commodore's computers. POWER is a programmer's utility package (in a 4K ROM) that contains a series of new commands and utilities which are added to the Screen Editor and the BASIC Interpreter. Designed for the CBM BASIC user, POWER contains special editing, programming, and software debugging tools not found in any other microcomputer BASIC. POWER is easy to use and is sold complete with a full operator's manual written by Jim Butterfield.

POWER's special keyboard 'instant action' features and additional commands make up for, and go beyond the limitations of CBM BASIC. The added features include auto line numbering, tracing, single stepping through programs, line renumbering, and definition of keys as BASIC keywords. POWER even includes new "stick-on" keycap labels. The cursor movement keys are enhanced by the addition of auto-repeat and text searching functions are added to help ease program modification. Cursor UP and cursor DOWN produce **previous** and next lines of source code. COMPLETE BASIC program listings in memory can be displayed on the screen and scrolled in either direction. POWER is a must for every serious CBM user.

Call us today, for the name of the Professional Software dealer nearest you.

Professional Software Inc.
51 Fremont Street
Needham, MA 02194
Tel: (617) 444-5224 Telex #951579

Ask The Readers

Robert Lock, Richard Mansfield, And Readers

COMPUTE! *welcomes questions, comments, or solutions to issues raised in this column. Write to: Ask The Readers,* **COMPUTE!** Magazine, P.O. Box 5406, Greensboro, NC 27403. **COMPUTE!** *reserves the right to edit or abridge published letters.*

Handicapped Computerists

Do there exist any large-screen CRT's for those with sight impairment? Or magnifying devices to enlarge a given CRT?

More generally, has anyone constructed a data base of products and ideas that concern the problems of the handicapped? I'd like to be able to reference and sort such a database with a home computer. Has anything been done in this direction?

Kevin Sinclair

Since most home computers attach to ordinary televisions, a large-screen TV might be one solution. Also, check with local television dealerships to see if they have a relatively inexpensive, free-standing magnifier now being marketed. Each month new clubs, newsletters, telecommunications "bulletin boards," and special interest data bases emerge. Also, **COMPUTE!** *publishes a bi-monthly column, "Micros With The Handicapped," and you might want to write to its author, columnist Susan Semancik, c/o* **COMPUTE!**.

Lowercase On Comprint

I have a PET with a Comprint 912 printer. The printer is capable of printing lowercase, but I have not been able to find anyone who knows how to write a program which would send lowercase to it from the PET.

M. Souza

*If your printer responds to standard ASCII code, you should be able to communicate with it by using the suggestions in the article "PET ASCII To ASCII" (***COMPUTE!***, April 1982, p. 126). This is a short machine language program which makes the necessary conversion between PET's way of coding and ASCII. Here's a sample BASIC program which would use this conversion routine. Don't forget the comma in line 30.*

```
10 OPEN4,4
20 INPUTA$
30 SYS634,A$
40 PRINT#4,A$
50 CLOSE4
```

PET To Epson

In the "Ask The Readers" column (**COMPUTE!**, June 1982), Hank Roth asked how to make his Epson printer print single line feeds instead of double line feeds. You can achieve this by ending each PRINT# line in your program with: CHR$(13);

(Don't forget the ending semicolon.) This can be added in two ways:

1. **100 PRINT#1,"(text here)"CHR$(13);**

or

2. **100 PRINT#1,"(text here)";**
   110 PRINT#1,CHR$(13);

The first method requires the least added memory, but adds line length.

Norman Girard

Computers In Medicine

In **COMPUTE!**, May 1982, #24, one of your readers, L. Thomas, wrote in to ask where to obtain a copy of the book *Computers In Medicine*. Unfortunately the title is not exact. I have written three books *Computers In Medicine* – an Introduction ($16), *Computers In Laboratory Medicine* ($28.00), and *Microcomputer Programs In Medicine* ($55.00). The books may be obtained from Medical Communications, Suite 10E, 860 Fifth Ave., New York. I hope that this information is helpful.

Derek Enlander, M.D.

A New VIC Champion

In **COMPUTE!**, June 1982, #25, "Ask The Readers," Mary Payne said she had the record for best score on Vixel's "Fire" game of seven seconds. Well, I put the fire out in six seconds, with 750 gallons remaining. So who's got the record now? I also think I have the high score on Vixel's "Race" game of 6421.

Mike DeLuca

Apple Baseball Scorekeeper

My problem is this: I have seven teams in a baseball game and I keep the win-loss record of each team in a text file on my Apple II Plus. I wrote a program that updates the files for me, but could not find a way to make the computer print the seven teams out in order by winningest to losingest record. What I am asking for is a routine that will sort the win-loss records of the teams, and print them out in the proper order. An example: Milwaukee currently has a 6 win-4 loss record, Boston is 6-4 also, New York is 5-5, and Pittsburgh is 3-7. I need

Your VIC 20 never had it so good!

VIC 20 owners rejoice! HES presents a complete range of software from our exciting series of games to our professional group of utility and language programs.

Our new cartridge programs include: HES MON, an indispensible monitor for assembly language programmers; HES WRITER, a word processing program; Turtle Graphics, a fun and easy way to learn computer programming; and VIC FORTH, a powerful language that is many times faster than BASIC, yet easier to use than assembly language.

HES is committed to offering high-quality, well-documented computer programs on a continual basis. Look for our cartridge and cassette based software at your local dealer.

HES

Human Engineered Software
71 Park Lane
Brisbane, California 94005
Telephone 415-468-4110

VIC 20 is a registered TM of Commodore

a routine that will sort these four, usually seven, teams and print them in the order: Milwaukee, Boston, New York, Pittsburgh.

Greg Palen

This program will read the seven teams (you only mentioned four, so three others were arbitrarily chosen) into the arrays N$,WINS, and LOSSES which respectively hold the team names, wins, and losses. The subroutine at 370 prints the list of the teams in table form, and includes the percentage, which is calculated with:

WINS/(WINS+LOSSES)

Note line 420, which rounds the variable A to three decimal points.

The sort, a simple bubble sort, uses the percentage to order the list. Teams with a higher percentage will appear before teams with a lower percentage. Since three arrays are moved around in the sort, it can be slow with large lists. You can change the number of teams by adding or deleting DATA items, and changing the variable NUM, the number of teams, at line 100.

```
100 NUM=7:REM NUMBER OF TEAMS IN DATA STATEMEN
    TS
110 DIM N$(NUM):REM NAME OF TEAMS
120 DIM WINS(NUM),LOSSES(NUM)
130 REM DATA STATEMENTS ARE: TEAM NAME, WINS,
    LOSSES
140 DATA MILWAUKEE,31,29,NEW YORK,29,29,BOSTON
    ,37,23,PITTSBURGH,27,30
150 DATA CLEVELAND,29,29,DETROIT,35,22,BALTIMO
    RE,31,28
160 FOR I=1 TO NUM
170  READ N$(I),WINS(I),LOSSES(I)
180 NEXT I
190 GOSUB 370:REM PRINTOUT
200 PRINT:PRINT "BEFORE SORT"
205 GETA$:IFA$=""THEN205
210 REM BUBBLE SORT ON PERCENTAGE
220 EXCHANGED=0
230 FOR I=1 TO NUM-1
240  A=WINS(I)/(WINS(I)+LOSSES(I))
250  B=WINS(I+1)/(WINS(I+1)+LOSSES(I+1))
260  IF A>=B THEN 310
270  TEMP$=N$(I):N$(I)=N$(I+1):N$(I+1)=TEMP$
280  TW=WINS(I):WINS(I)=WINS(I+1):WINS(I+1)=TW

290  TL=LOSSES(I):LOSSES(I)=LOSSES(I+1):LOSSES
    (I+1)=TL
300  EXCHANGED=1:REM NOTE EXCHANGE
310 NEXT I
320 IF EXCHANGED=1 THEN 220
330 REM NO EXCHANGES FOR ENTIRE LIST, SO SORT
    COMPLETE
340 GOSUB 370
350 PRINT:PRINT"SORTED"
360 END
370 PRINT CHR$(147);:REM USE HOME ON APPLE II+

380 PRINT "TEAM";TAB(15);"WINS";TAB(20);"LOSSE
    S";TAB(30);"PERCENTAGE":PRINT
390 FOR I=1 TO NUM
400  PRINT N$(I);TAB(15);WINS(I);TAB(20);LOSSE
    S(I);TAB(30);
410  A=WINS(I)/(WINS(I)+LOSSES(I))
420  A=INT(A*1000+.5)/1000:PRINT A:REM ROUND T
    O 3 DIGITS
430  PRINT
440 NEXT I
450 RETURN
```

Atari Microsoft Cartridge

Is there a possibility that Microsoft BASIC will be released as a ROM cartridge? It would be nice not to have to buy a disk drive and not to have to boot up for each use.

Eric Gallion

We know of no work being done on such a cartridge at this time. Atari says they know of no plans for one.

VIC User Groups

In the June 1982, #25 issue of **COMPUTE!**, a question from Fred S. Dart was printed in the "Ask The Readers" column. Mr. Dart asked whether any VIC users groups existed and how he could get in touch with them.

I have recently found out that there are now several in existence. I know of four which are primarily VIC users groups and one which is a PET users group, but now deals with the VIC as well.

Here is a list of these five users groups and their addresses:

Paradox Group
39-41 North Road
London N7 9DP
England

TBH VIC-NIC's
P.O. Box 981
Salem, NH 03079

VIC-20 Computer Club
c/o Ray Thigpen
4071 Edgewater Dr.
Orlando, FL 32804

The VIC-20 User's Group
c/o Roberto Morales, Jr.
655 Hernandez St.
Miramar, Puerto Rico 00907

Toronto Pet Users Group
c/o Chris Bennett
381 Lawrence Ave. W.
Toronto, Ontario
Canada M5M 1B9

Many other Commodore users groups seem to be taking on the VIC also.

Michael Kleinert
©

CBM/PET INTERFACES

The Connecting Links

CmC provides the link to increase your computer's functional ability. The following models come complete with case and cables and have a 30 day money back trial period.

PRINTER ADAPTERS

- addressable-switch selectable upper/lower, lower/upper case
- work with BASIC, WORDPRO, VISICALC and other software
- IEEE card edge connector for connecting disks and other peripherals to the PET
- power from printer unless otherwise noted

RS-232 SERIAL ADAPTER — baud rates to 9600 — power supply included
MODEL-ADA1450a $149.00

CENTRONICS/NEC PARALLEL ADAPTER — Centronics 36 pin ribbon connector
MODEL-ADA1600 . . . .$129.00

CENTRONICS 730/737/739 PARALLEL ADAPTER — 40 pin card edge connector
MODEL-ADA730 . . . .$129.00

EPSON MA-70 PARALLEL ADAPTER — handles graphics — BASIC 4.0 required
MODEL-ADAX70. . . . .$129.00

BYTEWRITER PARALLEL ADAPTER — Bytewriter connector — power supply included
MODEL-ADAB30a. $149.00

COMMUNICATIONS ADAPTER — serial & parallel ports — true ASCII conversion — baud rates to 9600 — half or full duplex-X-ON, X-OFF — selectable carriage return delay — 32 character buffer — centronics compatible — power supply included
MODEL-SADI $295.00

ANALOG TO DIGITAL CONVERTER — 16 channels — 0 to 5.12 volt input voltage range — resolution 20 millivolts per count — conversion time less than 100 microseconds per channel
MODEL-PETSET1 . . . .$295.00

US Dollars Quoted
$5.00 Shipping & Handling
MASTERCHARGE/VISA

All prices & specifications subject to change without notice

MENTION THIS MAGAZINE WITH ORDER AND DEDUCT 5% FROM TOTAL

IN THE USA order from:
Connecticut microComputer, Inc.
Instrument Division
36 Del Mar Drive
Brookfield, CT 06804
203-775-4595 TWX: 710 456-0052

IN CANADA order from:
Batteries Included, Ltd.
71 McCaul Street
F6 Toronto, Canada M5T2X1
(416) 596-1405

Dealer Inquiries Invited

What if you want more assurance your valuable data won't fade away?

Rely on SYNCOM diskettes with Ectype® coating. Balanced coercivity means long-lasting signal life.

Syncom diskettes assure excellent archival performance in the following ways.

First, with calibrated coercivity – a precisely balanced blend of milled ferrous oxides that allows Ectype® coating to respond fully to "write" signals, for strong, permanent data retention.

Then, a burnished coating surface to boost both signal strength and packing density.

Carbon additives drain away static charge before it can alter data.

And, finally, every Syncom diskette is write/read-back certified to be 100% error free.

To see which Syncom diskette will replace the one you're using now, *send for our free "Flexi-Finder" selection guide* – and the name of the supplier nearest you.

Balanced coercivity of Ectype® coating allows write current to saturate fully.

Syncom, Box 130, Mitchell, SD 57301.
800-843-9862; 605-996-8200.

SYNCOM

Manufacturer of a full line of flexible media

A Division of Schwan's Sales Enterprises, Inc.

A Monthly Column

Computers And Society

David D. Thornburg
Associate Editor

Chicago, Chicago ...

Each June I make my pilgrimage to the ancestral homeland for the purpose of attending the summer Consumer Electronics Show in Chicago. While I am in town, I also look up a few friends to see how the midwest is faring. To tell the truth, it is easy to fall into the trap of thinking that the San Francisco Bay Area is the intellectual capital of the universe. But a day spent walking the floors of the Chicago Art Institute, or talking with people like Sam Savage, quickly dispels that myth.

Sam is a mathematician/computer scientist/folk singer/inventor/entrepreneur whose recent fame has sprung from his company Shmuzzles, Inc. Shmuzzles, for those who haven't seen them, are jigsaw puzzles in which each identically shaped piece is in the shape of a salamander. (The connection between his design and the interlocking salamanders of artist M.C. Escher is covered in Dr. Savage's talk "Gerbils, Escher, Schlock," but I digress.)

Patterns that interlock perfectly to cover a plane (such as bathroom tiles) are called tesselating figures. The fact that Shmuzzles are both sophisticated mathematically and successful commercially is delightful.

While the projects Sam and I have cooking will be covered at a later time, I did want to share a short piece he has written on the social impact of technology:

Hey Dad I'm Getting A Disk Read Error

I became acutely aware of the electronic revolution over a year ago when I was having dinner at a friend's house. We were sipping wine on the patio when his ten year old son called from inside, "Hey Dad, I'm getting a disk read error" – the kind of computer lingo I was used to hearing from thirty year old technicians.

Where will it lead? Does history provide any parallels? With the development of the textile mills of the industrial revolution I am sure many people lamented the loss of the spinning wheel. Some, no doubt, observing the dwindling number of young women engaged in the spinning of wool, predicted that within a generation the entire country would be left naked.

So it is with the electronic revolution. Some are appalled that the calculator has diminished our ability to do arithmetic in our heads and predict a nation of numbskulls. Others rejoice that they can now do arithmetic to eight decimal places of accuracy on their forty dollar wrist watches. Some see today's video games as the final dissolution of civilization. Others see their five year olds mastering typewriter keyboards in order to peck out RUN SPACEINVADERS. Some are outraged that a large Chevy costing $3000 has been replaced by a small one costing $6,000 and complain that our standard of living will soon be reduced to that of Neandrethal man. Others are amazed that computers costing $300,000 have been replaced by those costing $3,000 and feel a sense of power and control over their lives unknown to previous generations.

Whatever the case, man's electronic extensions of himself are still only in their infancy.

Sam L. Savage 8 June 1982

Post Script – Upon presenting the above to my word processor's spelling checker, it politely informed me that I should have written "Neanderthal".

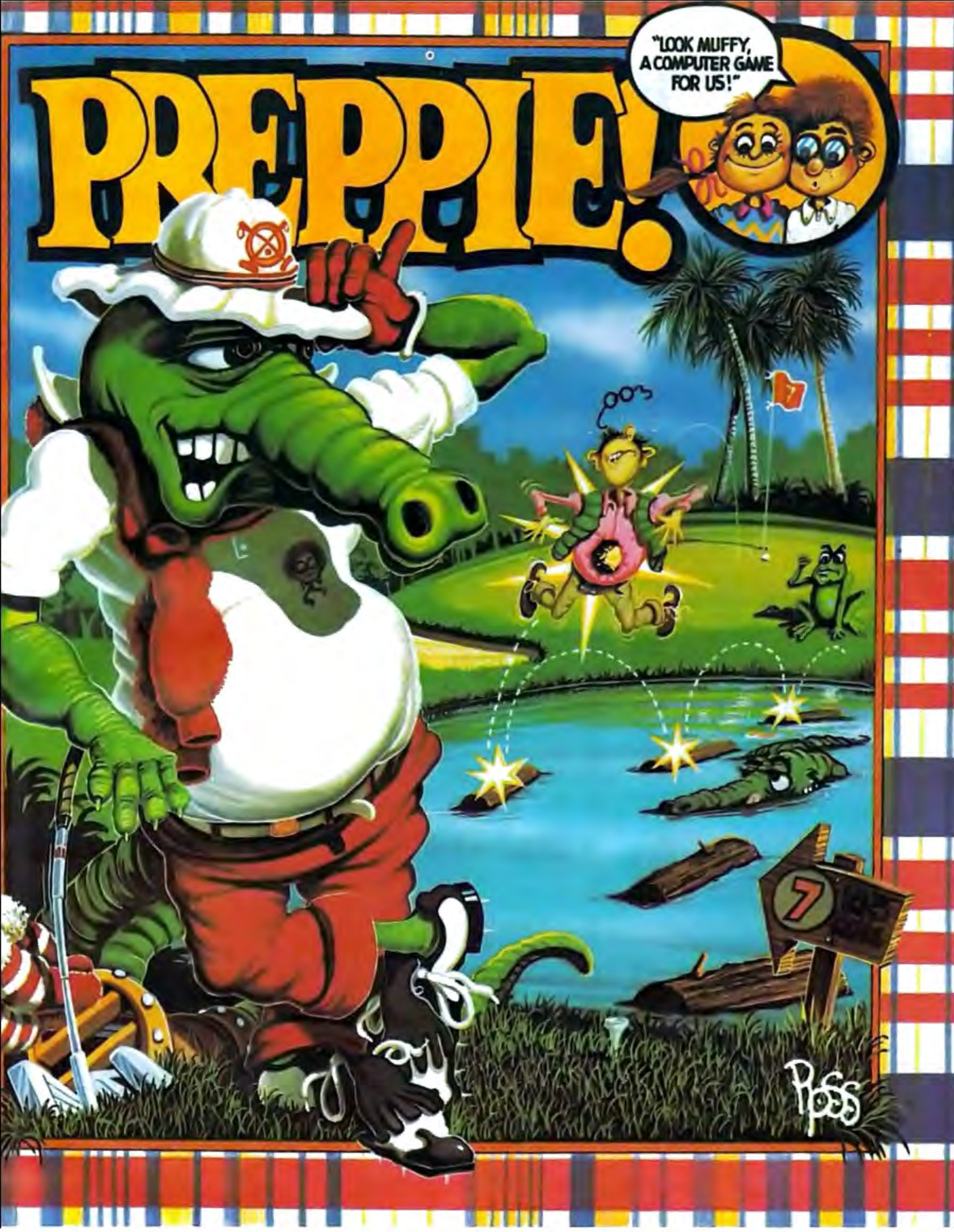
While children with access to computers are still in the minority, the trend is clear. A survey by a major merchandizing magazine showed that 5% of a national sample of families already had a personal computer in the house, and that an additional 5% were planning on purchasing one in 1982! What technology is it that will be moving into people's homes this year? For insight on that topic the CES is a tremendous resource.

You may recall that my report from the January CES in Las Vegas discussed the Ultimax from Commodore – a $150 computer to be introduced this year. Commodore made two changes between shows. They changed the name to the Max Machine and raised the price of the 2.5K RAM version of the machine to $179.95. Given the mass market channels (such as Toys R Us) through which the VIC-20 is presently being sold (how does a worldwide volume of 40,000 units per month grab you?), I would not be surprised to see the Max discounted to the original $150 price shortly after its October release.

I was also curious to see what Mr. Sinclair had up his sleeve. With his present computers being made by Timex at its plant in Dundee, Scotland, clearly he could devote his energies to the next product. He has.

The New Spectrum

The new Sinclair computer is called the ZX Spectrum – a color computer with 16 or 48K of RAM. This computer is already being sold in the UK for



Good Golly! What A GAME!

The original arcade game **PREPPIE!** will give you hours of toe-tickling fun. **You** take the controls, moving your young prepster about on the golf course. And what a golf course! It's unlike any you have ever played on before. There are dangers **everywhere,** and only you can keep the little Ivy Leaguer from a fate worse then Penn State!

PREPPIE! is written in state-of-the-art machine language, is joystick compatible and fully utilizes the Atari's sound and graphics capabilities. Quite frankly, it will give you the most fun you can have on an Atari microcomputer.

ATARI 16K TAPE — $29.95
ATARI 32K DISK — $29.95

PREPPIE! is available at fine computer, book, and hobby stores everywhere.

An exclusive game from

ai Adventure INTERNATIONAL

A DIVISION OF SCOTT ADAMS, INC.

To order, see your local dealer. If he does not have the program you want, then call 1-800-327-7172 (orders only) or write for our free catalog.

Published by ADVENTURE INTERNATIONAL
a division of Scott Adams, Inc.
BOX 3435 • LONGWOOD, FL 32750 • (305) 862-6917

"My Atari never did things like this before!"
—Holister Townsend Wolfe

"I had so much fun I almost blew my doughnuts."
—Theodore Boston III

"I haven't had this much fun since Buffy and I went to Princeton for the weekend."
—Martha Vineyard

under $350. The US price for the 16K version was not announced, but will probably be $199.95. Unlike the ZX-81, the Spectrum uses a nice mechanical keyboard with well-spaced medium travel keys. It sports a high resolution graphics mode with eight colors. Given the memory size, display capabilities, and mechanical (rather than membrane) keyboard, the Spectrum is a strong response to the Commodore Max.

Even if the Spectrum costs more than Max, the Sinclair distribution channels might more than make up for this deficiency. For example, a one-shot campaign for the ZX-81 was undertaken by American Express. The immediate response was about 2,000 orders per day. Not bad for a product with the ZX-81's limitations. With Timex selling through its 100,000 retail outlets, it will only be a matter of time before the ZX Spectrum will be available at your corner drugstore.

Perhaps Sinclair's *next* product will be a pocket TV to use as a display for these computers.

If nothing else, the battle between Sinclair and Commodore should dispel the myth of the so-called Japanese invasion – at least in this price range. At a higher price range, the NEC PC-6000 is quite impressive. This 16 to 48K RAM computer has three graphics modes as well as a 32 character by 16 line text mode. A printer interface and joystick ports are standard, as is a built-in 16K BASIC and a screen editor. While the price was not announced, I would guess that this product will retail for under $450. A graphics tablet (about $200) was also demonstrated for this computer.

Speaking of input devices, this CES was unique in the number of companies that were selling advanced joysticks and trak balls. WICO, a principal manufacturer of arcade game controllers, has adapted its devices for home use. For a modest $70 expense, your home Atari games can now use the same trak ball controllers found at the local pizza parlor arcade. Cynex, of Hillsdale, NJ, introduced GAME MATE II, a pair of radio remote control joysticks that let you control your computer without wires from up to 20 or 30 feet away. Aside from reducing the clutter of cables, this product opens the door to some very interesting applications for handicapped computer users.

And if anyone still needed proof that computers were mass-market items, what better example could one have than the fact that entertainment giants such as Thorn EMI (from England) and 20th Century Fox are now in the computer software business?

Yes, it promises to be an interesting year for that additional 5% of the US population that will also hear their children say, "Hey, I'm getting a disk read error!" ©


THEY CAME OUT OF THE BLUE OF THE WESTERN SKY
and it's up to you to stop them!
REAR GUARD
FEATURING ARCADE ACTION GRAPHICS AND SOUND
You know those alien residents who look like blue jello and smell like a cow pasture on a warm day? Well, they're up in arms about something and have declared war! Their weapons are Cyborg ships with their intelligent parts suspended inside. These Cyborgs pursue like crazed bats to try and kamikaze your crew pods. You're armed with an unlimited supply of Guided Energy Darts, but even when you blast 'em, their doomsday missiles come right at you! Good luck. (You'll need it!)
REAR GUARD features many extras which are usually found only on coin-operated video arcade machines. These include advanced play levels, running high score (with high score "clear" option), two player option, brilliant horizontal scrolling, and great sound effects.
APPLE VERSION
by John Anderson
48K DISK $29.95
ATARI VERSION
by Neil Larimer
Assisted by Sparky Starks
WE ARE OVERRUN
...ALL IS LOST
16K TAPE $19.95 24K DISK $24.95
TRS-80 VERSION
by Wayne Westmoreland and Terry Gilman
16K TAPE $19.95 32K DISK $24.95
TO ORDER!
Adventure INTERNATIONAL
A DIVISION OF SCOTT ADAMS, INC.
BOX 3435, LONGWOOD FL 32750
(305) 862-6917 (QUESTIONS)
ORDER FROM YOUR FAVORITE DEALER
or CALL TOLL FREE (800) 327-7172
(ORDERS ONLY PLEASE) 20

The Atari and Apple produce sounds of greater purity than some synthesizers, according to Frank Serafine, one of the creators of the TRON *sound track. He used these computers to create some of the movie's most memorable sound effects.*

The Sounds Of TRON

Tom R. Halfhill
Features Editor

His eyes locked in the grim intensity of combat, the video warrior takes aim at his opponent, slowly winds up like a baseball pitcher, and violently unleashes a glowing discus. Whooshing and screeching, the disc arcs over the arena toward its frantically dodging target – who has dodged too late. The disc demolishes the enemy gladiator in a split-second burst of light and sound.

The victorious warrior reaches up to retrieve his deadly disc, which zooms back to him like a boomerang. Standing tall in his glowing armor, TRON has triumphed again over the forces of the sinister Master Control Program.

This early scene from Walt Disney Productions' summer release, *TRON*, is typical of the film's pioneering use of animated computer graphics. But while *TRON*'s stunning visual effects have attracted the most attention from critics and audiences, fewer people are aware that the film's sound effects break new ground, too. The sounds, like the visuals, also are the product of high-technology computerization.

Apple And Atari Sounds

What is even more interesting, at least for personal computer enthusiasts, is the equipment used to fashion those sound effects: many were generated with an off-the-shelf Atari 800 and an Apple II.

Not only that, but an Atari 800 running a commonly available data base manager program was used to store, categorize, index, and instantly retrieve all the special sound effects collected for *TRON*. How many was that? "Oh, my gosh – thousands, just thousands of sound effects," says one of their key creators, Frank Serafine, of Serafine FX Music/Sound Design.

SFX – you can see its credits roll by at the end of *TRON*, if you watch closely – is a Los Angeles-based sound design studio hired by Disney to collect and create most of the special sounds heard in *TRON*. Equipped with the most advanced audio components and sound-dedicated computers available, SFX previously had done sound effects for *Star Trek: The Motion Picture* and *The Fog*. But *TRON* was the studio's most innovative and involved project by far – SFX labored for a year and three months shaping the film's sounds.

Electronic Sound Assembly

Serafine says the job would have taken even longer had he used the established method, improved little since the 1930s, of cutting and splicing bits and pieces of the soundtrack on mechanical film editors. Faced with a staggering task, Serafine turned to computers.

With his broad array of audio and computer equipment, including the Atari and Apple, Serafine designed a process he calls "Electronic Sound Assembly." Used for the first time in *TRON*, it does for the creation of sound what word processing does for writing: it allows the manipulation and fine-tuning of the work on a video screen. Serafine can digitalize sound effects, feed them into a Fairlight CMI (Computer Musical Instrument), plot the waveforms on a monitor and tinker with them almost endlessly. In fact, he can actually alter the sound directly on the screen with a light pen.

The results are sounds honed to an unbelievable degree of detail. Serafine says he was inspired by the sound-layering techniques pioneered by the Beatles in the 1960s with considerably less sophisticated equipment. "They achieved a subliminal effect, something which made you want to listen to their music over and over again to hear every sound. That's what I tried to do with the sound effects in *TRON*."

Consider the roar of the "light cycles," the futuristic motorcycles on which the gladiators of *TRON* duel while racing along the circuit grids of the computer in which they are trapped. To make the light cycles seem real, Serafine assembled and combined more than 50 different sounds. When a cycle makes a 90-degree turn to cut off an opponent, for example, the sound effect is a combination of video game tones generated on the Atari and a recording of a buzz saw.

Likewise, the sound of the sleek tanks which prowl the circuit canyons in *TRON* are a compilation of dozens of noises made with the Atari, all layered together. Serafine first tried recording a real army tank, but was disappointed by the clanking rattle. "I wanted something that sounded more turbine-like, more computer-controlled," he explains.

Screaming Monkeys

Other notable sound effects in *TRON* generated by

Introducing the PERCOM Alternative to ATARI Disk Storage

Your Atari 800 is the finest home computer on the market. Now you can own a floppy disk system that measures up — an RFD mini-disk storage system from Percom.

At Percom we've been making disk storage systems since 1977.

Our designs are proven, our quality is well known. And we back our dealers with service know-how. Expect more from Percom. You won't be disappointed.

- Operate in either single- or double-density storage mode using Atari DOS 2.OS. In double-density you can store almost 184 Kbytes (formatted) on one side of a 40-track diskette.
- Connect your Percom RFD first-drive system directly to your computer or connect into your system through your Atari 810 Disk Drive.
- Add an RFD first-drive system with its versatile four-drive controller, then connect as many as three more low-cost RFD add-on drives.
- Write application programs that can query and set up your system to operate a different type drive at each cable position — that can even change configuration as the program executes.
- Get quality and state-of-the-art capability at competitive prices. Percom first-drive RFD systems are priced from only $799, first add-on drive is only $459. Cables included.

Watch for announcement of a new, powerful, easy-to-use disk-operating system for your Percom-equipped Atari 800 computer.

Minimum system requirements — are an Atari 800 computer with 24-Kbytes of RAM and compatible video display system; Atari's disk-operating system (ver 2.OS) and owner's manual; and, for add-on drives (if used) an optional disk drives interconnecting cable available from Percom.

For the best thing next to your computer, see your Atari dealer about a Percom RFD floppy disk storage system. For the name of your nearest dealer, call Percom toll-free 1-800-527-1222.

PERCOM

PERCOM DATA COMPANY, INC.

11220 PAGEMILL RD • DALLAS, TX 75243 • (214) 340-7081

PRICES AND SPECIFICATIONS SUBJECT TO CHANGE WITHOUT NOTICE.

Yes . . . I'd like to know more about Percom RFD disk drives for my Atari 800 Computer. Rush me free literature.

Send to: **PERCOM DATA COMPANY, Inc.**, Dept. 18-C01
11220 Pagemill Road, Dallas, Texas 75243

name

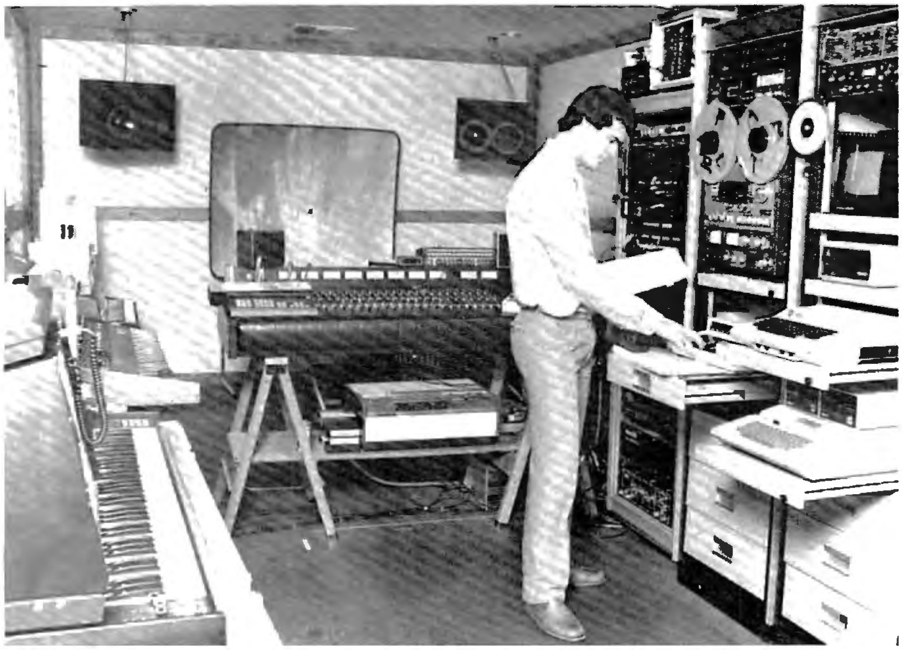
street

city state

zip phone number

I plan to add a hard disk system to my Atari: ☐ yes ☐ no.

MAIL TODAY!



Serafine in his studio, flanked by his Atari and Apple.

"bonging" sound was overdubbed with recordings of a bullwhip and of monkeys screaming at the San Diego Zoo.

"The director [Steven Lisberger] demanded a concept for each sound effect," explains Serafine. "I couldn't just go around doodling with sound effects. We had to sit around with the director in discussion sessions to talk over the concept of each sound effect. Like, for the disc-throwing sound, we came up with the concept that they had to sound beautiful, yet sad – sad because something so beautiful can at the same time kill. So overlaying the monkey screams lets you know that, although this flying disc is really beautiful, you also know you'd hate to be hit by it."

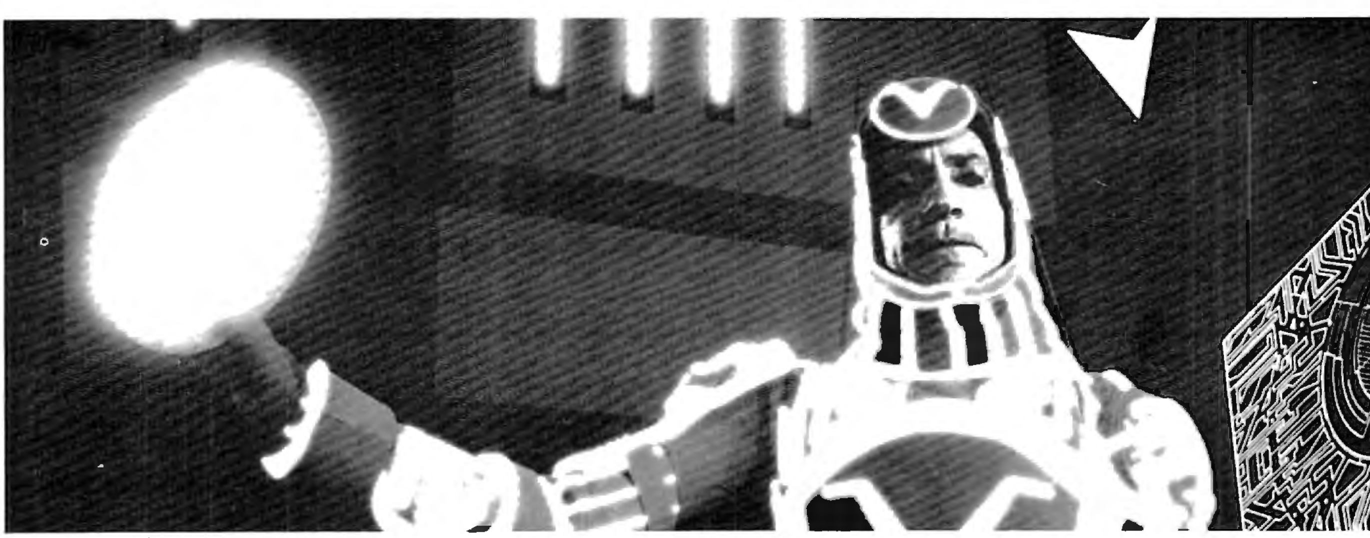
the Atari include numerous video game bleeps and zaps, the sound of the "grid bugs" which arise from the circuits below the "solar sailer," the shock prods wielded by the Master Control Program's guards, and the collisions of tanks with walls (combined with recordings of real military explosions).

The Apple II was used with plug-in sound cards from Mountain Hardware and an Alpha Centauri keyboard. These add-ons made the Apple capable of a wider range of sound effects than the Atari, says Serafine, but also made it harder to use. Still, the added capability was an advantage when programming certain video game sounds and the "bonging" noise of a thrown discus.

The discus sound is a good example of how much work went into each effect. The Apple's

It was originally Disney's idea to involve personal computers in the sound production. Serafine's background is in audio and multi-media presentations, not computers. He was designing planetarium shows in Colorado in 1976 when he first attracted Disney's attention. Disney hired him to put together a multi-media presentation for the grand opening of Space Mountain at Disneyland in 1977.

A few years later, after Serafine founded SFX, Disney hired him to create the sound effects for *TRON* and sent him to "Silicon Valley" in California, the home of America's microcomputer industry. Disney figured it was the ideal place to find video game sound effects for a movie whose central theme was to be video games. Serafine met with representatives from Apple and Atari, who set him



Sark, assistant to the arch-villain Master Control Program, introduces the killer disc.

GHOST ENCOUNTERS™

A 16K Assembly Language Real-Time Adventure Game For ATARI® 400/800™ Computer

GHOST ENCOUNTERS™ is the second in a series of real-time, joystick controlled, adventure games following ACTION QUEST,™ presenting a brand new collection of dangers and puzzles. You, as a ghost, travel through a network of 30 rooms in search of valuable prizes while, at the same time, try to survive the many perils encountered. Of course, a mere ghost cannot overcome all the evil powers striving to block his journey. Luckily, you are not a mere ghost, but are equipped with the power of transmutation, allowing you to take on the form of other, sometimes more useful, objects. Who knows when a torch or a hammer might be just what the witch doctor ordered.

Locating all 30 rooms and the 20 prizes scattered throughout requires a player with fast response, physical coordination, and intellectual resourcefulness. Of the 20 prizes, no two can be acquired through exactly identical methods. Regardless of which factors most interest you in a real-time action game, you are sure to find them among the wide variety of challenges located in GHOST ENCOUNTERS.™

Available from your local Atari retailer or send $29.95 in check or money order (California residents add 6½% sales tax) to JV Software Inc.

ALSO AVAILABLE

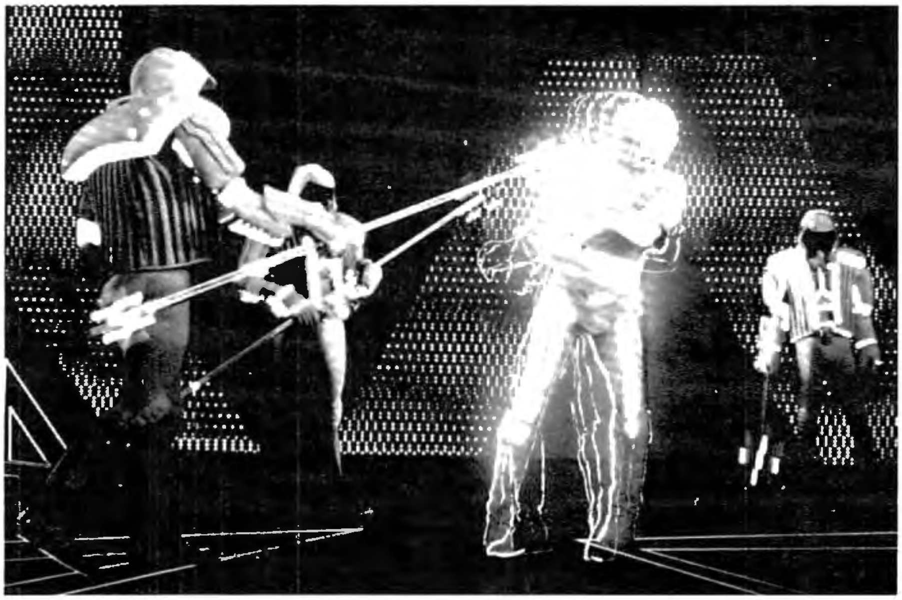
Don't forget to ask for the original ACTION QUEST,™ with its own unique and imaginative assortment of puzzles, prizes, and perplexing predicaments. Another challenging journey through an entirely different network of rooms and prizes.

Also available in 16K for $29.95 from your local retailer or JV Software Inc.

Atari® and 400/800™ are trademarks of ATARI, Inc.

JV SOFTWARE, INC

3090 MARK AVE. SANTA CLARA, CA 95051



Crack Programmer Flynn finds himself at the mercy of distinctly user-unfriendly guard programs.

up with computer systems. Atari also lent him an Atari Sound Development Disk, a well-guarded, powerful utility package rarely entrusted to anyone outside Atari itself.

Most of the Atari sounds were not actually programmed by Serafine, though. He got help from Ed Rotberg, an ace Atari programmer who has since left Atari to form his own company, and from another expert – Laurent Basset, who is 17. Basset is the son of one of Serafine's friends.

"He's a whiz kid," says Serafine. "This kid was actually able to do anything I wanted done on the machine. I would dream of a sound or a concept, and he would come back to me the next day with the finished programs."

All the thousands of sound effects created on the computers or collected on tape were cataloged on the Atari with *FileManager 800*, a data base program by Synapse Software. Serafine says it saved his studio hours of tedious filing. The record for any sound effect, listing its characteristics, source, and location on tape, could be retrieved in 1.5 seconds.

A Clean, Pure Sound

Serafine, who had only limited previous experience with personal microcomputers, was also impressed with the sound quality of the machines. "The amazing thing to me is the purity of the sound that comes out of the Atari, and also the Apple. Their sound chips produce,an extremely clean, pure sound which is even superior to some synthesizers I've worked with. We had no trouble using those sounds in the movie."

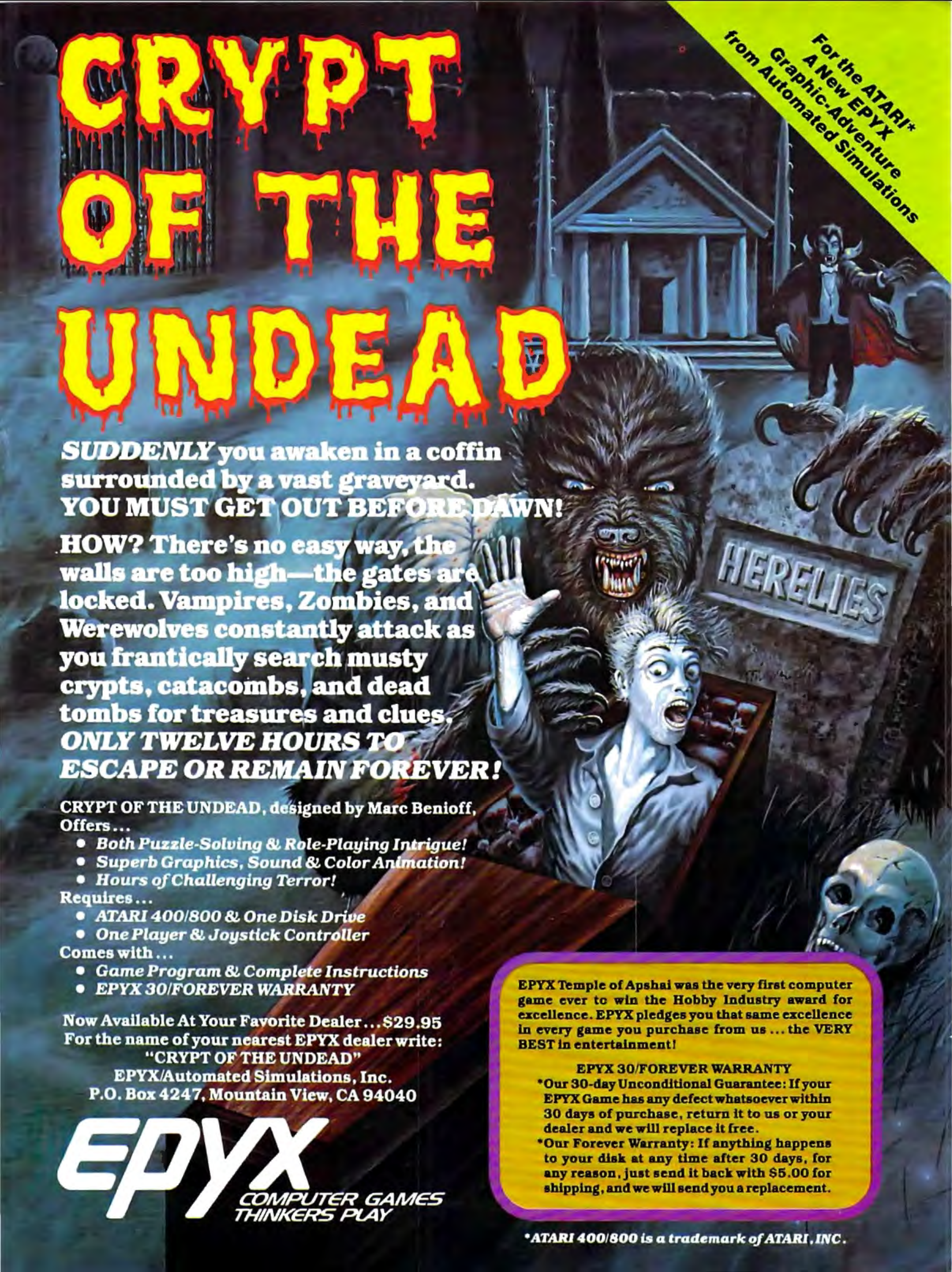
In fact, he thinks the sound capabilities of the computers are underused, partly because the proper tools are not available. "I think Atari and Apple will look at the success of the personal computers [in *TRON*] and develop better sound development disks as a result. For example, Atari has a *Music Composer* which gives you all the tools you need to compose music except really good sound. You can't access all the good sounds in the machine with the *Music Composer*. Someone at Atari ought to combine the *Music Composer* with features of the Sound Development disk, and they'd really have something. Maybe if they read this...."

Serafine sees a future for personal computers in other productions involving video game-like effects. In fact, his current project is a *Pac-man* commercial for Seven-Up.

Unfortunately, most people take sound effects for granted, he says. The sounds in *TRON* come and go so fast that almost nobody realizes the amount of labor involved. "Several people spend a week of intense work to create something," he sighs, "and it lasts only one second."

Three Atari sound effects by Laurent Basset, who helped program many of the Atari sounds in *TRON*.

```
10 ? CHR$(125):POKE 710,80:POKE 755,0:? "
THUNDER & RAIN"
14 FOR TIME=0 TO 1
15 B=INT(255*RND(0)+50):X=RND(0)*200
20 FOR PITCH=1 TO B:SOUND 0,PITCH,8,15:NEXT PITCH
25 FOR T=1 TO X:NEXT T:NEXT TIME:SOUND 0,0,0,0
27 FOR RAIN=0 TO 15 STEP 0.2:SOUND
1,0,0,RAIN:FOR W=1 TO 20:NEXT W:NEXT RAIN:FOR
W=1 TO 2000:NEXT W:SOUND 1,0,0,0
40 FOR X=1 TO 100:NEXT X
50 ? CHR$(125):POKE 710,80:POKE 755,0:? "
HEART BEAT"
60 FOR Y=1 TO 5:FOR W=1 TO 40:SOUND
0,12,3,15:NEXT W:FOR W=1 TO 150:SOUND
0,0,0,0:NEXT W:NEXT Y:SOUND 0,0,0,0
70 FOR X=1 TO 200:NEXT X
80 ? CHR$(125):POKE 710,80:POKE 755,0:? "
STEAM LOCOMOTIVE"
90 X=0.1
100 FOR W=1 TO 150
110 FOR LOUD=15 TO 4 STEP -X:SOUND
0,15,0,LOUD:NEXT LOUD
120 X=X+0.01:IF X>0.7 THEN X=0.7
130 NEXT W:SOUND 0,0,0,0
135 FOR S=1 TO 2
140 SOUND 1,40,10,10:SOUND 2,10,10,8:SOUND
3,90,10,10
145 FOR X=1 TO 200:NEXT X
147 SOUND 1,0,0,0:SOUND 2,0,0,0:SOUND 3,0,0,0
148 FOR X=1 TO 60:NEXT X
150 NEXT S
```


For the ATARI*
A New EPYX
Graphic-Adventure
from Automated Simulations
CRYPT OF THE UNDEAD
HERE LIES
SUDDENLY you awaken in a coffin surrounded by a vast graveyard. YOU MUST GET OUT BEFORE DAWN!
HOW? There's no easy way, the walls are too high—the gates are locked. Vampires, Zombies, and Werewolves constantly attack as you frantically search musty crypts, catacombs, and dead tombs for treasures and clues. ONLY TWELVE HOURS TO ESCAPE OR REMAIN FOREVER!
CRYPT OF THE UNDEAD, designed by Marc Benioff, Offers...
• Both Puzzle-Solving & Role-Playing Intrigue!
• Superb Graphics, Sound & Color Animation!
• Hours of Challenging Terror!
Requires...
• ATARI 400/800 & One Disk Drive
• One Player & Joystick Controller
Comes with...
• Game Program & Complete Instructions
• EPYX 30/FOREVER WARRANTY
Now Available At Your Favorite Dealer...$29.95
For the name of your nearest EPYX dealer write:
"CRYPT OF THE UNDEAD"
EPYX/Automated Simulations, Inc.
P.O. Box 4247, Mountain View, CA 94040
EPYX
COMPUTER GAMES THINKERS PLAY
EPYX Temple of Apshai was the very first computer game ever to win the Hobby Industry award for excellence. EPYX pledges you that same excellence in every game you purchase from us ... the VERY BEST in entertainment!
EPYX 30/FOREVER WARRANTY
*Our 30-day Unconditional Guarantee: If your EPYX Game has any defect whatsoever within 30 days of purchase, return it to us or your dealer and we will replace it free.
*Our Forever Warranty: If anything happens to your disk at any time after 30 days, for any reason, just send it back with $5.00 for shipping, and we will send you a replacement.
*ATARI 400/800 is a trademark of ATARI, INC.

The Beginner's Page

How Computers Think

Richard Mansfield
Senior Editor

Computers don't yet think in the broadest sense. Some believe that computers never will match the human mind in overall mental ability. Others argue that artificial intelligence is inevitable and might come to pass within the next ten to fifteen years. Before we can consider this question, however, we should first look briefly at the mechanics of computer thinking. Your personal computer does "think" in a way. It can remember, it can play a good game of chess, it can make certain kinds of decisions, and it performs math calculations far faster than we do.

How does your computer do the "thinking" that makes it a worthy chess opponent? For the answer, we have to get down to the simplest level. If we could enter the silicon chips inside the machine, we would see a vast pattern of intersections and pathways, like Los Angeles seen at night from a jet. Which path the electricity follows determines what happens. The computer controls this electric traffic with *gates*. There are great numbers of gates (the intersections), and they are the heart of the computer's decision-making process. Each gate makes a yes or no decision, like a traffic cop who signals "stop" or "go," to the electricity trying to flow past it.

How does this "gate deciding" take place in physical terms? The symbol of a gate looks like this:

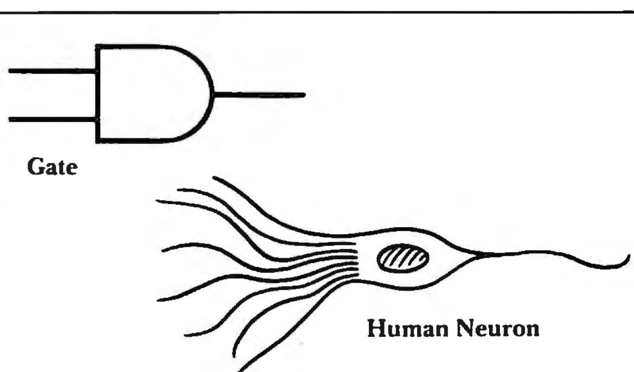


Figure 1.

with two lines (wires) coming into the gate and one line going out. The incoming lines are the facts, the basis for the judgment, and the line out announces the decision. A gate is a transistor (or diode). It functions like an automatic traffic light which turns green only after two cars have pulled up to the intersection. Beneath the pavement, in each lane, is a sensor which can tell if a car is sitting on it. The light stays red until both sensors are switched on.

If there is only one car in either lane, or no cars, the light stays red. With only these weight sensors connected by wires to the gate, a true decision can be made. The wire coming out of the gate is "turned on" when both incoming wires are on. This "out" wire is connected to the traffic light. In this way, electricity flowing through wires and gates can *decide* things, can think.

A gate which says "yes" when both of its incoming wires are on is called an AND gate. Both one wire AND the other must be on for the AND gate to say "yes." It is easy to see that this gate could have many uses. How would your computer decide whether to put a capital or a lowercase "C" on your TV screen when you press "C" on the keyboard?

Using Gates In BASIC

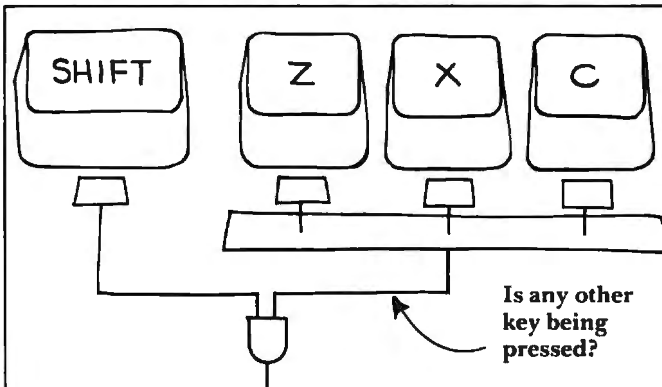


Figure 2.

There are multitudes of AND gates inside your computer. One of them could always be checking to see if a signal is coming in from the shift key at the same time as another signal is coming from a regular key. If *both* of these wires are on at the same time, the shift decision is made, so the wire coming out of the "Shift AND Gate" turns on and the computer displays an uppercase "C."

Less common, but also important, is the OR gate. It looks roughly the same as AND with two wires in and one out, but it says "yes" if *either* incoming wire is on. It might be useful for night traffic. If there is a car in either lane, the light switches to green.

There are times when the best way to solve a

programming problem in BASIC is to set up your own gates. If your computer's BASIC includes the words AND and OR, you can use them to experiment with this concept of gates. You could try:

```
10 INPUT A: INPUT B
20 IF A = 10 AND B = 5 THEN PRINT" AND ": GOT
    O 10
30 IF A = 10 OR B = 5 THEN PRINT" OR ": GOTO
    10
40 PRINT" NEITHER AND NOR OR ": GOTO 10
```

or you could get more complicated by, for example, checking to see if both numbers are less than 100 and B is 1/2 as large as A:

```
20 IF A < 100 AND A/2 = B THEN PRINT "CONDITI
    ONS SATISFIED"
```

The computer's "mind" is built of AND and OR gates, lots of them, arranged in various ways, to form the pathways along which an electric impulse flows. This spark races through the computer, darting now left, now right, at the gates. Things work similarly in the human brain, but there are significant differences.

A Thousand Ideas About Orange

In our brain, the wires leading into a gate are nerve fibers, the gates themselves are neurons, and the wire coming out of the gate (with the answer) is called an axon. Perhaps the most important difference is that a computer normally has two wires bringing facts into a gate for a decision. Some neurons have up to 100,000 incoming wires. A single decision can be influenced by that many "facts." Fortunately for us, the facts don't all need to be "true" to switch an AND neuron. The outgoing axon can say "yes" based on a percentage of the incoming votes.

Another significant difference contributing to the sophistication of human thought is that we can think many things at the same time. When you imagine an orange, your brain activates up to perhaps 1,000 separate thinking processes simultaneously. It provides pictures (of both the color and the fruit and possibly throws in an "associated" idea or two like a photo of an orangutan ape), taste memories, and hundreds of other pieces of information it has wired to the word *orange*. And, at the same time, it is selecting which of these thoughts to ignore, which to connect, and which have the highest value at the moment.

Computer Gates Can Switch A Million Times Each Second

Your computer, by contrast, can process only one "thought" at a time. This sort of thinking (A leads to B which leads to C) lends itself very well to math problems, but more or less eliminates poetry, common sense, or any thinking involving creativity or flexibility. At present, computers are excellent at solving problems where there is only one right answer, but weak with ambiguous tasks (including mastery of languages with their shadings and nuances).

Underlying these differences are some interesting contrasts between your brain and your computer's chips. Some estimates place the total number of gates in the brain at 10,000,000,000. Personal computers have far less density. The brain's memory may have 12,500,000,000,000,000,000 bytes (memory cells). We consider ourselves lucky if our computer has 48,000 bytes. This imbalance in size and complexity is somewhat offset by the great speed of the computer. A computer's gate can swing open or slam shut one million times every second. Our gates can open or close only 100 times per second. (But 1,000 of them can be operating at once, so it's not as bad as it sounds.)

At the vanguard of computer science, however, the gap is closing. The newest chips are coming close to the density of the human brain. Advances are being made in creating "parallel processors," computers which can, like us, handle many thoughts simultaneously. Gates with multiple incoming wires are being built. The latest computer speed records leave us hopelessly in the dust: some working computers can switch a gate in a nanosecond (one billionth of a second).

Will computers ever become intelligent in our sense of the term? Will a new life form, *silicon sapiens*, dominate Earth? Nobody knows. There is one thing to consider though: our mental machinery isn't getting any bigger or faster. We have had our present brain capacity and speed since prehistoric times. The modern electronic computer first appeared in the 1940's, it has rapidly improved since, and we cannot now say that there is any known limit to the ultimate power of a computer mind. ©

This program (for Atari and Microsoft BASICs) will aid in improving and maintaining speed reading skills.

Peripheral Vision Exerciser

Ron Kushnier
Richboro, PA

We have all seen or heard about people who can read a lengthy novel in an hour. Their eyes scan the pages faster than most of us can read a sentence. How do they do it? How much comprehension is really taking place?

Those who have taken speed reading courses have said that it is a matter of extreme concentration and practice, mostly practice. Their reading speed depends on the complexity of the reading material – a novel goes much faster than a technical article. Use of the technique must be constant. Once a reader strays from the method, he quickly forgets it.

The major technique used in speed reading is the concept of the word picture. When we see a word like *the*, we do not see it as " t...h...e ," but rather as an entire word picture. If we extend this idea and add another word, like *on the*, most of us can still see the entire picture rather than separate words. If we add *on the hill*, we now have a complete thought in one word picture.

Speed reading builds on this technique, starting with single words and progressing to newspaper columns, and finally to full-length sentences as found in standard texts.

A recent TV program devoted to the subject showed several speed reading tools. These consisted of charts of random letters appropriately spaced to exercise the peripheral reading vision of the student. I immediately thought, "Wow! What a terrific application for a computer!" Not only could I produce such a chart, but I could also animate it.

Those who are interested in trying speed reading may find this Peripheral Vision Exerciser of interest.

The program generates lists of random letters formatted in such a way that the field of view is gradually expanded. The eyes take in more letters at one time in this way.

The program consists of two levels: beginner and advanced. Concentrate on the screen and try to read the groups of letters with one eye fixation, that is, without moving your eyes *across* the screen. Press the SPACE key to call up another set of letters. As you progress, press the "M" key (more separation) to make things a little harder. To stop, or to go into another mode, just hit the STOP key, then hit RUN.

Program 1. Microsoft Version

```
1ØØ REM **PERIPHERAL VISION EXERCISER**
11Ø REM ** BY RON KUSHNIER **
12Ø INPUT"{CLEAR}ARE THESE CAPITAL LETTERS(Y/N
    ) ";A$
13Ø IFLEFT$(A$,1)="Y"THENPOKE59468,14:    F1=1
    :GOTO15Ø
14Ø IFLEFT$(A$,1)<>"N"THEN12Ø
15Ø INPUT"{CLEAR}DO YOU WANT LEVEL ONE(1) OR T
    WO(2)";A
16Ø IFA=1THENF=1:GOTO19Ø
17Ø IFA=2THENF=Ø:GOTO19Ø
18Ø GOTO15Ø
19Ø INPUT"{CLEAR}STARTING SEPARATION( >=2 )  "
    ;B
2ØØ PRINT"{CLEAR}"
21Ø IFF1=1THENA$="ABCDEFGHIJKLMNOPQRSTUVWXYZ":
    GOTO23Ø
22Ø A$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
23Ø X=1
24Ø DEF FNL(X)=INT(26*RND(X)+1)
25Ø FOR I =1 TO 1Ø
26Ø C$=MID$(A$,FNL(X),1)
27Ø IF F=1THEN3ØØ
28Ø C$=C$+MID$(A$,FNL(X),1)
29Ø C$=C$+MID$(A$,FNL(X),1)
3ØØ D$=MID$(A$,FNL(X),1)
31Ø IF F=1THEN33Ø
32Ø D$=D$+MID$(A$,FNL(X),1)
33Ø E$=MID$(A$,FNL(X),1)
34Ø IFF=1THEN36Ø
35Ø E$=E$+MID$(A$,FNL(X),1)
36Ø IF F=1THEN PRINT TAB(18- B)D$; TAB(18)C$;T
    AB(18+B)E$;CHR$(13):GOTO38Ø
37Ø PRINT TAB(18- B)D$; TAB(18)C$;TAB(19+B)E$;
    CHR$(13)
38Ø NEXT
39Ø POKE158,Ø
4ØØ GETA$:IFA$=""THEN4ØØ
41Ø IF A$="M"THENIFB<18THENB=B+1:GOTO2ØØ
42Ø GOTO2ØØ
```

Program 2. Atari Version

```
100 REM *** PERIPHERAL VISION EXERCISER
    **
110 OPEN #1,4,0,"K:":GRAPHICS 0:POKE 752
    ,1
150 ? CHR$(125);"Do you want level one (
    1) or two (2)":GET #1,A:A=A-48
155 IF A<1 OR A>2 THEN 150
160 F=1:IF A=1 THEN F=0
190 ? CHR$(125);"Starting separation (2-
    18)";:INPUT B
195 IF B>18 THEN B=18
200 ? CHR$(125);:POKE 559,0
250 POSITION 18-B,10:V=F*2:GOSUB 500
260 POSITION 18,10:V=F*3:GOSUB 500
270 POSITION 18+B,10:V=F*2:GOSUB 500
280 POKE 559,34
290 GET #1,A
300 IF A=ASC("M") THEN IF B<18 THEN B=B+
    1:GOTO 200
310 GOTO 200
500 FOR I=1 TO V
510 ? CHR$(65+INT(26*RND(0)));
520 NEXT I
530 RETURN
```

©

Though written for the PET, this in-depth exploration is useful to any computerist whose BASIC includes the DEF FN command.

User-defined Functions: Defined

Myron D. Miller
Indianola, PA

Have you ever written a program in which a certain formula had to be used quite often? Perhaps the fundamental expression was always the same, but a variable would need to change. Some formulas can get quite lengthy, and repetitious entry can become tedious. There is an alternative besides subroutines for this situation: the *user-defined function*. Let's take a look at the user-defined function.

I will be inventing some names in the article to help explain the operation of the function. Any term which is of my creation will be followed by "(my jargon)" when introduced. The first invented term is the abbreviation *UDF* (my jargon) for *U*ser-*D*efined *F*unction. Also, there are a lot of short and simple examples included in the article. Sit close to your PET and try the examples when you come across them. I think you will learn more by seeing the UDF in action.

What Is A UDF?

A UDF is an arithmetic function that is defined by the user. Once defined, the UDF is implemented just like a normal, resident BASIC function, and can be used at any point within the program. A UDF is somewhat like a variable, but rather than being a variable for data, it is a variable for an arithmetic formula. Like a variable, a UDF can be defined, called up, and redefined as often as desired. While similar to a variable, a UDF is not a variable, and you should avoid thinking of it as a variable. A UDF is for number crunching operations only. As such, string variables, string functions, and other non-arithmetic operations must not appear in a UDF. The string symbol ($) must not appear in the UDF's syntax. Forget about strings when working with UDFs.

When should you use a UDF? Whenever it is convenient and practical to do so. Generally, they are used for complex formulas which are repeated many times throughout the program. However, they can be as simple as you like. It is chiefly a question of practicality. If a formula appears only once in a program, a UDF is clearly impractical. If it appears many times, it may then become practical to implement a UDF. Another feature of the UDF is that one of the variables of the formula is made available for independent substitutions. Thus, values can be plugged into the UDF without changing variables found in the program. I think you will find the UDF to be an interesting as well as powerful programming tool.

UDF Syntax

Let's take a look at a UDF (see Figure 1 and Example 1). First, the UDF has two parts: 1. the definition statement – DEF FN...–. 2. the *execution statement* (my jargon) – FN...–. The definition statement tells the computer: "Remember this formula, we will need it later." This is similar to a LET for variables. The execution statement tells the computer: "Use that formula that I told you to remember." This is similar to calling up a variable for use. Figure 1 shows the required syntax incorporated into both statements of the UDF. Example 1 shows a simplified use of a UDF. (Note: the examples given in this article will emphasize simplicity rather than practicality.) We will take a detailed look at each segment of the UDF's syntax.

BASIC Keywords. DEF FN (*define function*) is the keyword that states: remember this function. It must have a line number; that is, you cannot use a

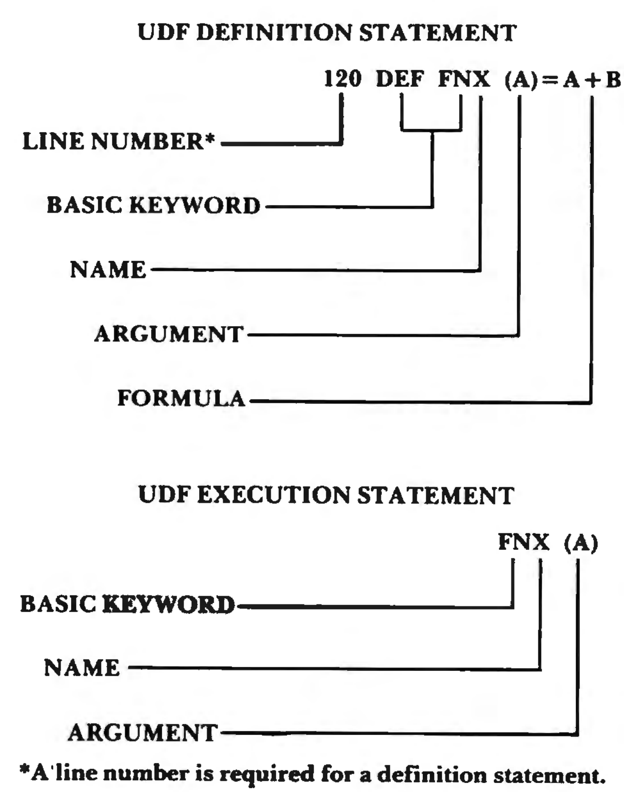


Figure 1. Syntax for a User-Defined Function. The definition statement must appear before the execution statement in the program.

Make your fun and games computer get serious
For only $79.95

VIC-20[1] The Wonder Computer of the 1980's. Less than $300. One heckuva lot of fun. But if you're using your VIC-20 strictly for recreation and fun, we at Micro-Systems think you're not getting your money's worth. Because we've designed the Micro-Systems VIE Cartridge which allows you to interface VIC to all existing Commodore IEEE peripherals (8050, 4040, 2031, and printers) for only **$79.95.** In addition, we've designed the V-232 RS-232 Interface board to allow the VIC-20 to communicate with various serial devices such as printers, modems, and other systems—**NOT JUST A BUFFER DRIVER**—for only **$45.00.** Also available are a full range of Ram Expansion Cartridges in increments of 8K, 16K, and 24K of ram expansion memory at very competitive prices. If you need to access more than one cartridge in your VIC-20 expansion port, we have the solution. The V-Expander is now available with 6 additional expansion ports DIP selectable.

And if you want to keep the fun in your computer. Micro-Systems has developed five all new video games **(VICTOROIDS, MOBILE ATTACK, CRACK, GUARDIAN, and QUARK)** Available in Cartridge; Cassette (requires 8K expansion); and disk (requires IEEE interface above and 8K expansion); all compatible with VIC-20.

Available from Micro-Systems

VIE Cartridge (IEEE-488)	$ 79.95	V-Expander (6-slot)		$ 75.00
VIC-20 Color Computer	$259.00	V-232 RS-232 Interface		$ 45.00
VIC-1540 Single Disk Drive	$499.00	Joy Stock (Arcade Quality)		$ 29.95
VIC-1515 Graphic Printer	$325.00	VTE Terminal Emulator		
VIC Modem	$ 89.95		Tape	$ 8.00
VIC-1210 3K Expansion	$ 35.00		Disk	$ 15.00
V8K Ram Expansion	$ 49.95	GAMES	Cartridge	$ 29.95
V16K Ram Expansion	$ 95.00		Cassette	$ 19.95
V24K Ram Expansion	$149.00		Disk	$ 19.95

DEALER INQUIRIES INVITED

Yes, Please send me:

QTY.	MODEL#	NAME	PRICE
TOTAL	(In Texas, add 5% sales tax)		

Your Name ______

Address ______

City ______ State ______ Zip ______

payment method: Check ☐ Card-Exp. Date ______

Master Card # ______

Visa # ______

American Express # ______

Signature ______

Micro-Systems • 11105 Shady Trail #103 • Dallas, Texas 75229 • (214) 484-7836

[1] Registered Trademark of Commodore Int.

SUNRISE SOFTWARE

LEADER IN DISCOUNT SOFTWARE — 18 - 30% OFF

ATARI

Action Quest	D	23.00	K-Razy Shootout	C	39.95
Adventure 1-12 ea.	T	15.95	Lisp Interpreter	D	120.00
Adventure on a Boat	D	19.95	Lords of Karma	T	16.00
Ali Baba & 40 Thieves	D	26.35	Lunar Lander	T	12.00
Angle Worms	T	12.00	Match Racers	D	23.96
Andromeda	D	31.96	Missle Command	C	32.00
Assembler Editor	C	47.95	Mountain Shoot	T	12.00
Asteroids	C	32.00	Mouseattack	D	29.95
Bishops Square	D	23.95	Next Step	D	31.95
Cavern of Mars	C	32.00	No. Atl. Convoy Raider	T	13.00
B-1 Nuclear Bomber	T	13.00	Nukewar	T	13.00
Bug Attack	D,T	23.95	PacMan	C	35.95
Centipede	C	35.95	Pathfinder	D	27.95
Character Generator	D	15.95	Planet Minders	T	13.00
Conflict 2500	T	13.00	Pool 1.5	D	29.95
Compu-Read	D,T	23.95	Programmer Kit	C	63.88
Compu-Math/Decimals	D,T	31.95	Q.S. Forth	D	63.96
Compu-Math/Fractions	D,T	31.95	Raster Blaster	D	23.95
Crush, Crumble & Chomp	D,T	23.95	Rear Guard	T	15.95
Crossfire	D,T	23.95	Rescue at Rigel	D,T	23.95
Datastones of Ryn	D,T	15.95	Ricochet	D,T	15.95
Deflection	T	11.95	Shattered Alliance	D	31.95
Dung Beetle	D	23.95	Softporn Adventure	D	23.95
Empire of Overmind	D	28.00	Space Invaders	C	32.00
Entertainer Kit	C	82.00	Starbase Hyperion	D	18.35
Fort Defiance	T	16.00	Star Raiders	C	36.00
Frogger	D	29.95	Star Warriors	D,T	31.95
Galactic Empire	T	15.95	Tanktics	D	23.20
Galactic Trader	T	15.95	Temple of Apshai	D,T	31.95
Ghost Hunter	T	24.00	Threshold	D	31.95
Ghostly Manor	D	19.95	3D Tic-Tac-Toe	T	15.95
HI-RES Mission Asteroid	D	19.95	3D Supergraphics	D,T	31.95
HI-RES Wiz & Princess	D	26.35	Treasure Quest	T	15.95
Invasion Orion	D,T	19.95	VersaWriter Tablet	A	240.00
Jawbreaker	D	23.95	Visicalc	D	200.00
Kayos	D,T	29.95	Warlocks Revenge	D	28.00

VIC/20

Amok	T	19.95	Skymath	T	12.00
Alien Blitz	T	21.95	Space Division	T	12.00
Kosmic Kamikazee	T	21.95	Spiders of Mars	C	42.00
Cloud Burst	C	34.95	Super Hangman	T	15.15
Renaissance	C	42.45	3D Maze	T	12.00
Sub Chase	T	21.95	VICalc	T	12.95
Outworld	C	34.95	VICat	T	19.95
The Alien	T	21.95	VICheck	T	19.95
Meteor Run	C	42.00	VITerm A	T	15.95
			RS232 Board	A	44.95
			8K Cartridge	A	71.95

PET

B-1 Bomber	T	13.00	Guns Fort Defiance	T	16.00
Computer Acquire	T	16.00	Midway Campaign	T	13.00
Computer Stocks & Bonds	T	16.00	N.A. Convoy Raider	T	13.00
Conflict 2500	T	13.00	Planet Miners	T	13.00
Nuke War	T	13.00	Lords of Karma	T	16.00
Tantics	T	20.00			

D-Disk, C-Cartridge, A-Accessory, T-Cassette

SUNRISE SOFTWARE • 2126 SHADY BROOK DR., 1000 OAKS, Ca 91360

24 HOURS A DAY
7 DAYS A WEEK

ORDER TOLL FREE
800 - 854-0561 EXT. 812
800 - 432-7257 (CALIF.)

NO HASSLE! GUARANTEED REPLACEMENT POLICY

We take MASTERCARD or VISA. (Include card # and expiration date.) California residents add 6% tax. Include $2.00 for postage. Prices subject to change.

1 Mhz - 12 Bit A/D

for your Apple II Computer

The APPLESCOPE-HR12 analog to digital converter uses a high stability buried zener voltage reference and a flash A/D to give 12 bit accuracy with a 14 bit dynamic range

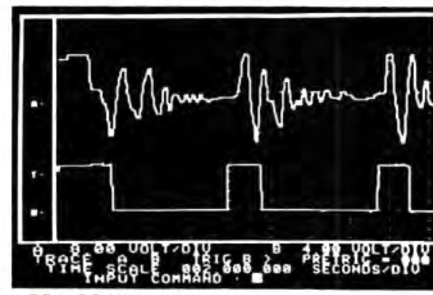
- DC to 1 Mhz Programmable Sample Rate
- 2048 Sample Buffer Memory
- Pretrigger Viewing
- Continuous or Single Sweep
- 4 Channel Software Support (requires additional power supply)
- External Trigger Input

The standard software provided with each APPLESCOPE-HR 12 includes all of the functions necessary to turn your Apple II computer into a high quality digital storage oscilloscope. In addition all of the SCOPE DRIVER options are being up-graded to handle the higher resolution data.

Price per channel **$695**

The original APPLESCOPE still provides the optimum price/performance trade off for those users requiring 8 bit converter resolution

APPLESCOPE INTERFACE

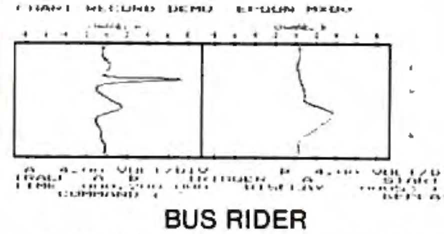


- DC to 3.5 Mhz sample rate
- 1024 byte buffer memory
- Pretrigger Viewing
- Programmable Scale Select
- Continuous and Single Sweep Modes
- Single or Dual Channel Trace

Price for the two board Applescope system is **$595**
EXTERNAL TRIGGER ADAPTER **$29**

SCOPE DRIVER Advanced software for the APPLESCOPE analog to digital converters makes full use of the computing power of the Apple II to create a total data acquisition system. Available options include:

- Signal Averaging-Acquires 1 to 999 signal sweeps and displays the averaged result.
- Digital Volt Meter - Allows use as real time DVM or use to measure points on an acquired sweep.
- Disk Storage - Allows automatic storage and recover of acquired data on floppy disks.
- Spectrum Analyzer-Calculates and displays frequency spectrum of acquired data.



BUS RIDER
LOGIC ANALYZER for the APPLE II

The BUS RIDER circuit card silently rides the Apple II peripheral bus and allows real time tracking of program flow. Software provided allows set up of trace parameters from the keyboard and read back of disassembled code after a program has been tracked

- 32 bit by 512 sample memory buffer
- Monitors Data and Address bus plus 8 external inputs
- Trigger on any 32 bit word or external trigger
- Pretrigger viewing

The BUS RIDER is an invaluable development tool for anyone working with Apple II or Apple II+ computers.

Price **$395**

RC ELECTRONICS INC.
5386 Hollister Ave., #D, Santa Barbara, CA 93111
(805) 968-6614

LISTING

```
110 A=3:B=5
120 DEF FNX (A)=A+B
130 PRINT FNX (A)
```

COMPUTER'S ACTIONS

110 A=3:B=5
120 Define a function named *X*.
FNX (A)=A+B
130 Execute the function named *X*.
FNX (A)=A+B
FNX (A)=3+5
FNX (A)=8 Print on screen.

RESULT – COMPUTER DISPLAYS:

```
8
READY.
```

Example 1. A simple usage of a User-Defined Function. Note: do not worry about the arguments for now. They will be explained later in the article.

definition statement in the direct (calculator) mode. The definition statement must appear in the program before the execution statement. FN is the keyword that states: use the remembered function. The execution statement must appear after the definition statement. It can be used in direct mode, but only after the definition statement has been *run* in a program – *run*, not just listed.

Rules For BASIC Keywords

1. DEF FN must have a line number.
2. DEF FN must appear before FN in the program.
3. FN may be used in direct mode if a DEF FN has been run in a program.

Names: A UDF must have a name. I chose the name "X" for the UDF in Figure 1 and Example 1. A program can have many different UDFs in it; we are not limited to one. Thus, the name identifies which particular UDF we wish to use. Even if the program contains only one UDF, it still requires a name. Also, both the definition statement and the execution statement must use the same name for any specific UDF.

UDF names follow the same rules as variable names. Thus we can use: a single letter – A,B,C...Z; or two letters – AA, AB, AC...AZ, BA, BB, BC...ZZ; or a letter followed by a number – A0, A1, A2...A9, B0, B1, B2...Z9. Like a variable, a UDF can use a longer name, and it will be plagued with the same problems that crop up when long names are used with variables. *First* – the entire definition statement (line number, keyword, name, argument, and formula) must fit on one 80 character line, just like any other program line. Thus, the more space that is used by a name, the less space left for a formula. *Second* – long names are used in the listing only – *not in the program run*. During the run, the computer looks at only the first two characters of the function or variable name. Thus, if two UDFs have long names that begin with the same first two characters (e.g., *RA*DIUS and *RA*DIANS), the first UDF that was entered will be redefined and lost when the second UDF is defined. Hence, no matter which of the two UDFs is called up, only the second UDF will be executed. To further complicate matters, the listing will appear to be correct because long

LISTING

```
210 A=2:B=3:C=7:D=9
220 DEF FNRADIUS (Z)=A+B
230 PRINT FNRADIUS (Z)
240 DEF FNRADIANS (Z)=C+D
250 PRINT FNRADIANS (Z)
260 PRINT FNRADIUS (Z)
270 PRINT FNRABBIT (Z)
```

COMPUTER'S ACTIONS

210 A=2:B=3:C=7:D=9
220 Define a function named *RA*.
FNRA (Z)=A+B
230 Execute the function named *RA*.
FNRA (Z)=A+B
FNRA (Z)=2+3
FNRA (Z)=5 Print on screen.
240 Define a function named *RA*.
FNRA (Z)=C+D
250 Execute the function named *RA*.
FNRA (Z)=C+D
FNRA (Z)=7+9
FNRA (Z)=16 Print on screen.
260 Execute the function named *RA*.
Same action as line 250.
270 Execute the function named *RA*.
Same action as line 250.

RESULT – COMPUTER DISPLAYS:

```
5
16
16
16
READY.
```

Example 2. This program demonstrates the problem of using two UDFs with names that begin with the same first two characters. FNRADIUS is redefined and lost in line 240. Thus, FNRADIUS and FNRABBIT execute the same as FNRADIANS. This is because the computer operates with only the first two characters of the name – FNRA – during the run. The long names are used in the listing only. Again, for now, don't worry about the arguments.


STAR.
WE'RE MAKING A

NAME FOR OURSELVES.

Building a reputation for quality printers of astronomical value, from components on up.

Making a name for ourselves in printers is only natural when we've been making the highest quality precision parts for over 30 years. And although Star's reputation in microperipherals has only begun to grow, the watch and camera industries worldwide have been relying on Star components for nearly 3 decades.

Quality manufacturing is what we're building our reputation on. Star is a true manufacturer, not merely an assembler. We even design and manufacture the computer controlled machines that make our parts. With unlimited production capabilities, Star's factories can operate 365 days a year, 24 hours a day, producing more than 100 million parts a month. It is this kind of productivity and total quality control that assures you of getting the very finest printer for the very lowest price.

And the best product for less is what you can depend on getting from Star. Star offers you a full line of reliable printers, printers with divergent technologies to meet your specific needs. And there's more. You get a low cost, high quality printer with a wide spectrum of interfaces to choose from—backed by Star's superior service network, there if you need it.

Star. We're going to be the quality printer you can count on, from components on up. And that's no pie in the sky promise.

star
micronics • inc

MAKING A NAME FOR OURSELVES

200 PARK AVENUE, NY, NY 10166

names are retained in the listing. (See Example 2.) *Third* – BASIC keywords cannot be nested (contained within) a two or more character name. Thus, the following names will cause a syntax error: *GO*, *INT*EGRATE, SUB*TOT*AL, KI*LOG*RAM, *FRE*QUENCY, FACT*OR*, SUBTRACTI*ON*, *STAN*DARD. (See Example 3.)

LISTING

```
310 A=2:B=3
320 DEF FNSUBTOTAL (Z)=A+B
330 PRINT FNSUBTOTAL (Z)
```

COMPUTER'S ACTIONS

310 A=2:B=3

320 Define a function named *SU*. TO ???? Computer thinks: Why is this TO here? I didn't see a FOR statement or a GO. I do not understand this. Print: ?SYNTAX ERROR IN 320. Terminate program execution.

330 Not executed because of SYNTAX ERROR IN 320. Will also cause a syntax error if 320 is corrected.

RESULT – COMPUTER DISPLAYS:

```
?SYNTAX ERROR IN 320
READY.
```

Example 3. This program demonstrates what happens when a BASIC keyword is nested in a UDF name. During the program run, line 320 tells the computer to define a function named *SU*. Then, before the argument and the formula are given, the line tells the computer to perform a TO. The computer does not understand this instruction, and lets you know with a syntax error message. Try changing the *O* in SUBTOTAL to an *A* (SUBTATAL) in line 320 only, and run it. You will still have a syntax error, but now located in line 330. Do the same in line 330 and the program should run with no problems. You should now get:

```
5
READY.
```

In my opinion, long function or variable names have no socially redeeming value other than to keep aspirin manufacturers busy. Avoid them! If you need documentation, use REM statements (REMarks). Don't forget that some two-character names can get you into trouble also (IF, GO, TO, ON, FN).

The name of the UDF serves only to identify the function. The name does not relate to or affect any variables in any manner. Thus, a program may contain UDFs and variables with the same name, and such variables may be used in the UDF. The function will operate normally because there is no interaction between UDF names and variable names. Also, the % (integer) sign and the $ (string) sign must not appear in the UDF name, or an error message (SYNTAX for %, TYPE MISMATCH for $) will result. A UDF is an arithmetic function, not a variable. So, integer and string signs have no meaning and are forbidden in the UDF's name. The best advice on UDF and variable names is: keep it simple. You can't go wrong with a single letter or a letter followed by a number (A3, X9, etc.) for a name.

Rules For Names

1. A UDF must have a name.
2. Both the definition statement and the execution statement must have the same name.
3. UDF names follow the same rules as variable names.
4. BASIC keywords, %, and $ must not appear in the name.
5. Names for different UDFs must not begin with the same first two characters.
6. UDF names do not relate to or affect any variable.
7. (Recommendation) Limit the names to one or two characters.

Arguments: The arguments for UDFs behave somewhat differently from the other BASIC functions. The *CBM User Manual* and the *PET/CBM Personal Computer Guide* call the arguments dummy variables or dummy assignments. They are not that simple! The arguments are extremely useful, and contribute a great deal to the power and flexi-

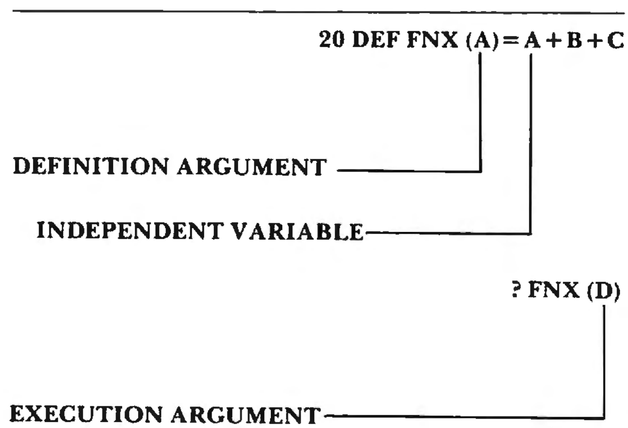


Figure 2. The UDF's arguments. The definition argument selects a variable in the formula to be the independent variable. The value of the independent variable is determined by the execution argument. The independent variable is a separate entity from program variable of the same name. In the above function, the value of *D* will be plugged into the independent variable during the execution of the function. Thus, the equivalent formula for the above function is FNX=D+B+C.

bility of the UDF. Let's see how they operate.

Both the definition statement and the execution statement must have an argument contained in the parenthesis of each statement. Each statement must contain only one argument. For simplicity, let's refer to the definition statement's argument as the *definition argument* (my jargon), and the execution statement's argument as the *execution argument* (my jargon). See Figure 2.

The definition argument selects *one* variable from the UDF's formula to be the *independent variable* (my jargon). The independent variable is no longer a program variable and can exist only within the formula of the UDF. It is a new entity and, as such, it neither affects nor is affected by a program variable with the same name. Its operation is independent of the program variable. Another way of looking at it is that the definition argument selects a location within the formula for the independent variable. The location is marked by the variable name in the definition argument. The independent variable may appear in the formula as often as required, but there can be only one variable name in the definition argument.

The execution argument provides a value to be "plugged," or substituted, into the independent variable during the function execution. See Example 4. The other variables of the formula (those not specified in the definition argument) will operate with their current values as assigned in the program. So, the definition argument selects an independent variable, and the execution argument provides a value for it.

LISTING

```
410 A=2:B=3:C=7
420 DEF FNX (A)=A+B
430 PRINT FNX (C)
440 PRINT A
```

COMPUTER'S ACTIONS

```
410 A=2:B=3:C=7
420 Define a function named X.
    FNX=I.V.+B
430 Execute the function named X.
    FNX=C+B
    FNX=7+3
    FNX=10 Print on screen.
440 A=2 Print 2 on screen.
```

RESULT – COMPUTER DISPLAYS:

```
10
2
READY.
```

Example 4. This program demonstrates the operation of the UDF's arguments. The argument in line 420 selects the variable for the independent variable. The definition argument actually selects a *location* in the formula for the independent variable. The location is indicated by underscoring in the comments to the right (I.V.). The execution argument in line 430 provides the value for the independent variable. Notice that *B*, *C*, and even *A* (line 440) do not change values.

LISTING

```
510 A=2:B=3:C=7:D%=9:E=-25
520 DEF FNX (A)=A+B+C
530 PRINT FNX (A)
540 PRINT FNX (D%)
550 PRINT FNX (E)
560 PRINT FNX (K)
570 PRINT FNX (150)
580 PRINT FNX (SQR(D%↑3) )
```

COMPUTER'S ACTIONS

```
510 A=2:B=3:C=7:D%=9:E=-25
520 Define a function named X.
    FNX=I.V.+B+C
530 Execute the function named X.
    FNX=A+B+C
    FNX=2+3+7
    FNX=12 Print on screen.
540 Execute the function named X.
    FNX=D%+B+C
    FNX=9+3+7
    FNX=19 Print on screen.
550 Execute the function named X.
    FNX=E+B+C
    FNX=-25+3+7
    FNX=-15 Print on screen.
560 Execute the function named X.
    FNX=K+B+C
    FNX=0+3+7
    FNX=10 Print on screen.
570 Execute the function named X.
    FNX=150+B+C
    FNX=150+3+7
    FNX=160 Print on screen.
580 Execute the function named X.
    FNX=SQR(D%↑3)+B+C
    FNX=    27     +3+7
    FNX=37 Print on screen.
```

RESULT – COMPUTER DISPLAYS:

```
12
19
-15
10
160
37
READY.
```

Example 5. This program shows a variety of execution arguments in action. In line 560, *K* is an unassigned program variable. Therefore, *K*=0.

The definition argument must specify *one* floating point variable only. String ($), integer (%), and subscripted (from arrays) variables must not appear in the definition argument. The same is true for BASIC keywords, numerical values, and arithmetic expressions. The execution argument may use floating point, integer, and subscripted variables. However, string variables must not be used in the execution argument. Direct numerical values, arithmetic expressions, or a program variable with the same name as the independent variable may also be used in the execution argument. The value of such items will be substituted in the same manner as the variables.

A UDF may be executed as often as desired in the program, and each time a different execution argument can be used. Thus, a variety of values can be plugged into the function's independent variable by changing nothing other than the execution argument. If an unassigned variable is entered as an execution argument, zero will be substituted into the independent variable. Any variable equals zero until the program LETS, READS, INPUTS, or otherwise assigns a value to it. Example 5 shows the various possibilities for execution arguments.

If it is necessary to change the assignment of the independent variable to another variable within the UDF's formula, then the UDF must be redefined using the desired variable as a definition argument. (See Example 6.) If the original UDF is also required, then a new UDF, with a different name, should be defined for the change. Also, it is possible not to have an independent variable. If the definition argument contains a variable which is not used in the formula, there will be no independent variable in the formula. The function will operate on the current values of the variables found in the formula. No substitutions will take

LISTING

```
610 A=2:B=24:C=8
620 DEF FNX (A)=B/A
630 PRINT FNX (C)
640 PRINT FNX (12)
650 DEF FNX (B)=B/A
660 PRINT FNX (C)
670 PRINT FNX (12)
```

COMPUTER'S ACTIONS

```
610 A=2:B=24:C=8
620 Define a function named X.
    FNX=B/I.V.
630 Execute the function named X.
    FNX=B/C
    FNX=24/8
    FNX=3 Print on screen.
640 Execute the function named X.
    FNX=B/12
    FNX=24/12
    FNX=2 Print on screen.
650 Define a function named X.
    FNX=I.V./A
660 Execute the function named X.
    FNX=C/A
    FNX=8/2
    FNX=4 Print on screen.
670 Execute the function named X.
    FNX=12/A
    FNX=12/2
    FNX=6 Print on screen.
```

RESULT – COMPUTER DISPLAYS:

```
3
2
4
6
READY.
```

Example 6. This program demonstrates the reassignment of the independent variable to another variable in the formula. Note that the only item changed is the definition argument (line 650). Yet, look at the difference in results.

LISTING

```
710 A=2:B=3:C=7
720 DEF FNX (Z)=A+B+C
730 PRINT FNX (C)
740 PRINT FNX (100)
```

COMPUTER'S ACTIONS

```
710 A=2:B=3:C=7
720 Define a function named X.
    FNX=A+B+C
730 Execute the function named X.
    FNX=A+B+C
    FNX=2+3+7
    FNX=12 Print on screen.
740 Execute the function named X.
    FNX=A+B+C
    FNX=2+3+7
    FNX=12 Print on screen.
```

RESULT – COMPUTER DISPLAYS:

```
12
12
READY.
```

Example 7. This program demonstrates a UDF without an independent variable. The definition argument, *Z*, is not found in the formula. Therefore, there is no independent variable, and the function does not use the execution argument. The same result will occur, regardless of value of the execution argument. Note: both arguments must still be included in their respective statements, or a syntax error will result.

place in any variable. The execution argument, although still required, will have no effect on the function, and hence, is a dummy assignment. (See Example 7.)

Examples 4, 5, 6, and 7 demonstrate the powerful effects that the arguments have on the UDF. Note the differences in the results of the examples caused by manipulating the arguments. Obviously, if you are to effectively use UDFs, you must have a clear understanding of the operation of the arguments. Try devising some UDFs of your own on paper; see if you can predict the results for specific arguments, and then try them on your computer.

Rules For Arguments

Definition Argument

1. There must be one, and only one, argument in the definition statement.

2. The argument must be contained in parentheses.

3. The argument must be a floating point variable only; no %, $, or subscripted variables and no numbers, BASIC keywords, or expressions are allowed in the definition argument.

4. The definition argument selects *one* variable from the formula to be the independent variable.

5. To change the independent variable selection within the formula, the UDF must be redefined.

6. If the argument does not appear in the formula, there will be no independent variable.

Execution Argument

1. There must be one, and only one, argument in the execution statement.

2. The argument must be contained in parentheses.

3. The argument can be a floating point, integer, or subscripted variable. Numbers and arithmetic expressions are allowed. No string variables.

4. The value of the execution argument is substituted into the independent variable.

5. A new argument may be used each time the function is executed.

6. If the argument is an unassigned program variable, 0 will be substituted into the independent variable.

7. If the formula does not contain an independent variable, the execution argument, though still required, will have no effect on the formula.

Formulas: What can we use in the formula? A general rule: if it has something to do with math, it can appear in the UDF's formula. Let's see what we can use.

Variables: floating point, integer (%), and subscripted variables may be used in the formula. String ($) variables must not appear in the formula.

Numbers: any numerical value within the normal range of the computer can be used. Also the symbol π (pi) may be used.

BASIC Commands: (CLR, LIST, LOAD, ... etc.) may not appear in the formula.

BASIC Statements: (DEF FN, DIM, FOR/NEXT, ... etc.) may not appear in the formula.

String Functions: (ASC, CHR$, LEFT$, ... etc.) and String Concatenation (+) may not be used in the formula.

Arithmetic Functions: (ABS, ATN, COS, ... etc.) may be used in the formula.

Arithmetic Operators: (+, -, *, /, ↑, - negation) can be used.

Boolean Operators: (AND, OR, NOT) can be used in bit-oriented operations.

Relational Operators: (=, <, >, <=, >=, <>) can all be used in the formula.

Exceptions: There are some non-arithmetic functions that will work in the formula. All of these functions are of a "return a value" nature. The following is a list of these exceptions: ASC, LEN, VAL, FRE, PEEK, POS, ST, TI, USR. I suggest using these functions with caution. Thoroughly experiment with the functions before including them in a program.

A UDF can have only one formula. A variable, number, operator, or function can be used as often as needed within the formula, but the entire definition statement must fit on one 80-character line. If there is a bug (error) in the formula, it will not show up until the function is executed. The appropriate error message will be displayed, but it will be referenced to the line of the execution statement, not the definition statement. Thus, the following program would result in a ?DIVISION BY ZERO ERROR IN 30. Note the error is referenced to line 30, but the correction will have to be made in line 20.

```
10 A=2:B=3:C=7
20 DEF FNX (A)=A+(B/0)
30 PRINT FNX (C)
```

Keep this in mind; it will make your debugging effort much easier. Don't forget about the order of evaluation (multiplication is performed before addition, etc.) and the other rules applicable to formulas. UDF formulas behave just like any other formulas with the exception of the independent variable.

Rules For Formulas

1. A UDF can have only one formula.
2. The formula and the other elements of the definition statement must fit on one 80-character line.
3. Any variable except strings ($) may be used.
4. Any number and π may be used.
5. Any arithmetic function or operator may be used except DEF FN.
6. Non-arithmetic functions may not be used. (See text for exceptions.)
7. Relationals and bit-oriented Boolean functions may be used.
8. A bug in the formula will show up in the execution statement line, not the definition.
9. To modify the structure of the formula, the UDF must be redefined.
10. Any variable or function may appear as often as needed in the formula.

How Are UDFs Used?

UDFs are used in the same way that any other BASIC function in the program is. The first thing to realize is that no action takes place during the definition statement other than the function being stored in memory. The definition statement performs nothing; the function is performed in the execution statement. This is why errors in the formula show up at the first execution statement, rather than the definition statement. Don't try to use the function with a definition statement (e.g., J = DEF FNX (A) = A + B + C); it will not work. The UDF is always used, that is, *performed* (and the result used), with an execution statement. Also, bear in mind that in order to execute a UDF, you must first define it. Failure to do so results in an ?UNDEF'D FUNCTION ERROR. Watch out for GOSUBs, IF/THENs, GOTOs, and anything else that may block the definition statement.

You must do something with the execution statement. Try this:

```
10 A=2:B=3:C=5
20 DEF FNX (A)=A+B+C
30 PRINT FNX (A)
40 FNX (A)
```

The above program will give you a 10 (from line 30) followed by ?SYNTAX ERROR IN 40. Why? Line 30 proves that the UDF is OK. So, why the syntax error? The PET, like humans, resents pointless work. In the orderly world of machine logic, everything is done for a reason. Computers do not entertain idle thoughts, which is precisely what line 40 is: an idle thought. It does nothing with the result of the function, and the computer does not understand what it is supposed to do. Imagine a stranger approaching you on the street; his only words are "FNX (A)", and he then impatiently waits for your reply. You would probably deliver a syntax error message of greater magnitude than the computer's. Quite simply, you have to do something with the result of the UDF execution statement. What can you do with it? You can do anything except treat it as a variable and try to assign a value to it (e.g., let FNX (A) = 3, READ FNS (A), INPUT FNX (A) are all no-no's). Remember, the execution statement provides a result, or an output; it cannot be used as an input. Figure 3 shows some of the ways you can use a UDF in a program.

USE IN PROGRAM	COMMENTS
DEF FNX (A) = A + B	**Definition statement, can be used only to define a function.**
PRINT FNX (3)	**Print result on screen.**
J = FNX (S)	**Sets variable *J* equal to result.**
IF FNX (D%) = Y THEN 500	**Testing result in an IF/THEN.**
Z5 = FNX (F) AND 64	**ANDing result with 64.**
FOR B = 1 TO FNX (Q)	**Limit on a FOR/NEXT statement.**
R2 = SQR(FNX(Q))	**Argument for another arithmetic function.**
ON FNX(P1) GOSUB 200,300,	**Index for an ON GOSUB.**
POKE32767 + FNX(T),99	**Address calculation for a screen POKE.**
PRINT VTAB(FNX(G))U	**Argument for a TAB.**
DEF FNY (D) = FNX (N) + D + E	**Definition of another UDF. See nested UDFs in text.**

Figure 3. (This is not a program.) Shown above are some of the ways that UDFs may be implemented in a program. The definition statement cannot perform the function; the execution statement must be used to implement the UDF. Some action must occur with the execution statement. And the execution statement must always act as an output of data. That is, the UDF cannot be a receiver of data like a variable, so you cannot LET FNX, INPUT FNX, READ FNX, or GET FNX.

Nested UDFs: The execution statements of one or more *previously defined* UDFs may appear in the formula for a new UDF. See the last line of Figure 3. Why next UDFs? It is a very effective way to create formulas that exceed the 80-character limitation on a definition statement. Nesting allows very long formulas to be used in the program. Also, nesting is the programming convenience. If you have two formulas, the second containing the first (e.g., *A+B+C* and *A+B+C+D+E*), the definition of the second UDF can contain the execution of the first UDF. This saves re-entry of the same formula. See Example 8.

LISTING

```
810 A=2:B=3:C=5:D=7:E=9:F=10:G=20
820 DEF FNX1 (A)=A+B+C
830 PRINT FNX1 (F)
840 DEF FNX2 (D)=FNX1 (F)+D+E
850 PRINT FNX2 (G)
860 DEF FNX3 (D)=FNX1 (D)+D+E
870 PRINT FNX3 (G)
```

COMPUTER'S ACTIONS

```
810 A=2:B=3:C=5:D=7:E=9:F=10:G=20
820 Define a function named X1.
    FNX1=I.V.+B+C
830 Execute the function named X1.
    FNX1=F+B+C
    FNX1=10+3+5
    FNX1=18 Print on screen.
840 Define a function named X2.
    FNX2=FNX1 (F)+I.V.+E
850 Execute the function named X2.
    FNX2=FNX1 (F)+G+E
    Execute the function named X1.
    FNX1=F+B+C
    FNX1=10+3+5
    FNX1=18 Send to FNX2.
    FNX2=18+G+E
    FNX2=18+20+9
    FNX2=47 Print on screen.
860 Define a function named X3.
    FNX3=FNX1 (I.V.)+I.V.+E
870 Execute the function named X3.
    FNX3=FNX1 (G)+G+E
    Execute the function named X1.
    FNX1=G+B+C
    FNX1=20+3+5
    FNX1=28 Send to FNX3.
    FNX3=28+G+E
    FNX3=28+20+9
    FNX3=57 Print on screen.
```

RESULT – COMPUTER DISPLAYS:

```
18
47
57
READY.
```

Example 8. Nested UDFs. Note in line 860 that the execution argument of FNX1 is an independent variable for FNX3. Compare the execution of FNX3 with FNX2.

UDF nesting is not without problems. If too many UDFs are nested, an ?OUT OF MEMORY ERROR will result. Try Example 9. Notice how many free bytes are left. How can you be out of memory? As each function is executed, data is entered on the *stack.*

The stack is a section of memory containing 256 consecutive locations. The processor uses the stack to store addresses and data when it is called to perform another function before completing the current operation. When the interrupting operation is completed, the processor removes the stored information from the stack and continues where it left off. If the processor is interrupted while working on an interrupt, then interrupted while working on that interrupt, etc., at some point the stack will be filled and the program will terminate with an ?OUT OF MEMORY ERROR.

When the stack is full, an ?OUT OF MEMORY ERROR results regardless of how much RAM is available. The program can be further compounded by GOSUBs, FOR/NEXTs, and any other operation

```
910 A=2:B=3:C=7
920 DEF FNXA (A)=A+B
925 DEF FNXB (A)=FNXA (A)+C
930 DEF FNXC (A)=FNXB (A)+C
935 DEF FNXD (A)=FNXC (A)+C
940 DEF FNXE (A)=FNXD (A)+C
945 DEF FNXF (A)=FNXE (A)+C
950 DEF FNXG (A)=FNXF (A)+C
955 DEF FNXH (A)=FNXG (A)+C
960 DEF FNXI (A)=FNXH (A)+C
965 DEF FNXJ (A)=FNXI (A)+C
970 DEF FNXK (A)=FNXJ (A)+C
975 PRINT FNXJ (A)
980 PRINT "FREE BYTES="FRE(0)
985 PRINT FNXK (A)
```

RESULT – COMPUTER DISPLAYS:

```
68
FREE BYTES=31318          Exact value will vary.
?OUT OF MEMORY ERROR IN 985
READY.
```

Example 9. This program demonstrates consumption of the stack by excessive nesting of UDFs. The *68* results from line 975, thus executing FNXJ did not fill the stack. Line 980 proves that there is plenty of RAM left. (The exact value depends on memory size, and if there is other programming stored.) Executing FNXK in line 985 fills the stack, and the program terminates with the ?OUT OF MEMORY ERROR message.

that uses the stack. If this happens, you have no alternative other than to reduce the complexity of the nested UDFs. You can always set a variable equal to a lower function, and then use it to pass the data to the higher UDFs (for Example 9, change line 970 to: 970 Q=FNXJ (A):DEF FNXK (A)=Q+C).

When nesting, a second UDF cannot be defined with the definition statement of the first (e.g., 10 DEF FNX2 (A)=DEF FNX1 (A)=A+B+C). An error will result when either function is executed. When a UDF is nested within a second UDF's formula, the first UDF must be defined before the second UDF is executed. That is, the computer must see the definition statement of the first UDF (as well as the second) before the execution statement of the second UDF can be performed. Failure to do so will result in an ?UNDEF'D FUNCTION ERROR when the second UDF is executed. Also, a UDF cannot redefine itself by nesting the original execution statement in the formula for the new UDF (e.g., 100 DEF FN*X1* (A)=FN*X1* (A)+B+C+D). In other words, the UDF in the formula must have a different name than the UDF that is being defined, or an ?OUT OF MEMORY ERROR will result.

Example 8 shows that the execution argument of the UDF in the formula can be an independent variable of the UDF being defined. This can lead to some powerful and interesting possibilities, and it can also lead to some real debugging problems if you are not careful. If you do not want other values substituted into the nested UDF, be sure that the execution argument is not an independent variable of the UDF being defined. The UDF FNX2 (in Example 8) has FNX1 nested in its formula. The execution argument of FNX1, *F*, is not an independent variable of FNX2.

When FNX2 is executed (line 850), FNX1 will still operate with *F* as the execution argument. FNX3 is the same as FNX2 except that the execution argument of FNX1 in the formula has been changed to *D*, the independent variable of FNX3. Compare the execution of FNX3 (line 870) with the execution of FNX2 (line 850). Quite a difference! Imagine the mess that could evolve out of nesting four or five complicated UDFs. You must be careful when selecting arguments for nested UDFs.

UDFs In Immediate Mode

The PET is a very poweful calculator when used in the direct mode. Unfortunately, typing in a few dozen SQR, SIN, EXP, etc, can get to be a real drag. You may be able to use a UDF to eliminate some of that typing. Let's see how.

Let's assume that we have a series of calculations to perform. We don't have time to fool around with a program and, besides, the calculations vary, so a program is not practical. Throughout the series of calculations appears the equation:

$$C=\sqrt{A^2+B^2}$$

The good, old Pythagorean theorem. In BASIC the equation appears as:

```
C=SQR(A↑2+B↑2)
```

Now who in their right mind wants to type that in a dozen times or so? Let's create a UDF to calculate the equation. We cannot define a UDF in the direct mode; we would get an ?ILLEGAL DIRECT ERROR. No problem. We will define the function with a one line program:

```
10 DEF FNC (A)=SQR(A↑2+B↑2)
```

Now we must RUN the one line program. Nothing appears to happen, but an action did occur: the UDF was stored in the memory. OK, we can now use our UDF in the direct mode.

Type in:	**B=4:?FNC(3)**	**Press RETURN.**
Result:	**5 is displayed.**	
Type in:	**B=557:?FNC(332)**	**Press RETURN.**
Result:	**648.438895 is displayed.**	

Anytime we need to use the above equation in our calculations, all we have to do is assign a value to *B* and execute the UDF. The value for *A* is entered in the execution argument.

Do you need cube roots in direct mode? If so, try this:

```
20 DEF FNA (X)=EXP(LOG(X)/3)
```

RUN the program, and try some direct mode cube roots.

Type in:	**?FNA(27)**	**Press RETURN.**
Result:	**3 is displayed.**	

Try the same thing with 343; you should get seven. (The question mark, ?, is an abbreviated PRINT command.) If you replace the three in the above equation with a variable, say *Y*, you can use it to find any root. FNA will return the "Yth" root of *X*; that is, FNA will equal $\sqrt[Y]{X}$

The examples I have given are rather simple, but they demonstrate the idea. You can have definition statements up to 80 characters, and you can nest UDFs for direct mode as well as in the program mode. Thus, UDFs can transform your PET into a super-calculator for direct mode calculations. Just remember to define the UDFs with line numbers, and RUN the program before attempting to execute the UDFs. You should have to RUN the definition program only once, unless there is a power failure.

Where do you go from here? *Experiment!* Use UDFs in your programs. Your formulas don't have to be *Einsteinian* to qualify for UDFs. If you run into something that I didn't cover, or find a difference in operation with other ROMs, or, if you have a unique application, send it in to **COMPUTE!**.

If you do a lot of work with trig, you may find pages 62 ad 63 of the *CBM User Manual* (see references) of interest. Commodore lists about 20 different UDFs involved with trig. If you like games, I have one more technique for you. Games use a lot of random numbers, quite often with a variety of ranges within the same program. Here is a UDF that will give you a random integer in a range of 1 to *X*, where *X* is the maximum desired number.

```
10 DEF FNR (X)=INT(X*RND(1)+1)
```

To use the function in a program, simply execute the UDF with the maximum desired number placed in the execution argument. For example: PRINT FNR (500) will print a random integer between one and 500. In the same program you may have:

```
POKE32767+FNR (1000),42.
```

This would display an asterisk in a random location on the screen. You can use the function as often as required, and the range is a simple matter of choosing an execution argument.

References

Butterfield, Jim. "Commodore ROM Systems: Terminology." **COMPUTE!**, *July 1981, #14, pp. 146-148.*

Donahue, Carroll S., and Janice K. Enger. PET/CBM Personal Computer Guide. *First Edition. Berkeley: Osborne/ McGraw-Hill, 1980.*

CBM User Manual. *First Edition. P/N 320856-3. Commodore Business Machines. Santa Clara, CA. 1979.*

©

In this in-depth profile of **COMPUTE!** *Associate Editor and Columnist Jim Butterfield, he discusses the future of computing, his background and hobbies, and his views on topics ranging from computers in education to the appeal of programming.*

Meet Jim Butterfield

Gail Hook
Barris, Ontario

The Butterfield homestead is a modest brick house within walking distance of downtown Toronto. It is comfortably cluttered with books, plants, computers, and three cats. Even the attic is pressed into service as storage space for whatever books and computers Jim Butterfield cannot cram into his small office.

The office, in fact, resembles a crowded depot for a changing assortment of computers – including four Commodore PETs of varying screen sizes and ages, a VIC-20, an Atari 800, a KIM, a Rockwell AIM, and an Alpha, a European machine. Stacked next to the computers is a "disk tower" consisting of two Commodore double disk drives (a 4040 and an 8050), an Atari 810 drive, and an ancient Commodore 2023 printer perched on top. Bookshelves along one wall are overflowing, and every available inch of floor space is carpeted with piles of diskettes, papers, and still more books. Yet, amazingly, Butterfield always seems to know into which pile to dive for what he needs.

One of the three cats, the Siamese, possesses a similar instinct. With a feline knack for homing in on the center of warmth and attention, she often dozes atop whichever PET is on and humming.

The main occupant of the office – Butterfield – meshes with the environment, too. He speaks with a gravelly voice in the measured phrases of someone used to teaching or being quoted for publication. Middle-aged and greying, he brings to microcomputing an almost childlike curiosity and sense of delight, a fascination which led him first to an absorbing hobby and finally, in early 1981, to a new career as a freelance writer, consultant, and teacher. Today he is recognized as a premier expert on Commodore computers, as a prolific writer, and perhaps most of all as an unusually coherent voice in the seemingly impenetrable technical thicket of personal computing.

A Change of Careers

Like most career changes, the switch surprised Butterfield as much as anybody. For 24 1/2 years he worked for Canadian National/Canadian Pacific Telecommunications. He quit solely because the company decided to move far away from central Toronto, and he would have spent so much time commuting there would have been none left for his hobby. For Butterfield, it was no contest. "When faced with that choice, I really had no choice and I quit."

Actually, it was while working for CN/CP in 1964 that Butterfield was first introduced to computers – although personal microcomputers were still undreamt-of in those days. Butterfield spent a year as a programmer of a rather specialized computer, a Collins C8401. FORTRAN and COBOL were coming into use at the time, but the Collins didn't use any such advanced languages. Programmers had to do almost everything in machine language. Butterfield soon moved into other areas of the company, but a little more than ten years later his interest was rekindled by a new invention – microcomputers.

"I decided to find out what this 'micro' stuff was all about and started watching the current magazines," he says. "I finally decided to purchase when I saw a completely pre-built machine called a KIM-1, which had a 6502 microchip in it. That turned out to be like a return to the past. Everything we had been doing a dozen years before on the large $1.5 million computer, we were doing again on this little $250 board – including making the same mistakes."

KIM And The Start Of Social Computing

One machine led to another, and Butterfield began sharing his knowledge with other microcomputer users, as well as writing about his discoveries. He had gained some writing experience many years before in western Canada, where he was born, as a "continuity writer" for a couple of radio stations. (Butterfield smiles, "That means I spent about a year of my life writing commercials.")

As the users of early microcomputers began comparing notes, it wasn't long before a cult of sorts sprang up. Indeed, the emergence of microcomputers as a basis of social, and not merely technical, interaction is the facet of the field that Butterfield enjoys most. In the earliest days of "roll-your-own-computers," he notes, everyone had a different machine, which crimped the sharing of information. "Suddenly, along came the KIM. Everybody had the same computer. An amazing thing happened – and this is multiplied many times over in the Commodore line – people built a social life around microcomputers."

The thriving Toronto PET Users Group (TPUG) is a case in point. Butterfield had what he calls a "Machiavellian influence" on TPUG founder Lyman Duggan, whom Butterfield persuaded to hold the first meeting in his basement one summer evening. While Butterfield firmly rejects any organizing chores, he contributes a great deal as a friend of the club, speaking at monthly meetings and sharing his expertise.

"An amazing thing happened ... people built a social life around microcomputers."

Butterfield admits, "It's getting harder to know what to talk about at those meetings. There are a number of people who have the ability to track down any part of the machine they want to go after, and who are quite skilled at machine language. As a result, my sympathy is with the beginner. I'd rather bore ten experts than lose the bulk of people, so I try to keep things fairly simple."

Butterfield's sympathy for beginners is well known and shows in his articles. His writing is informal and witty in spite of its technical content. "I try to write it as I would say it. I do a lot of presenting material to both kids and adults, and I try to keep the same style in my writing. Also, whenever I can, I slip in a simple example program. Then, even if the readers can't understand what I mean, they can run the programs."

Light Consulting

Butterfield also indulges in what he calls "light consulting," principally for Commodore. In the spring he went on a western Canadian promotional tour for the VIC-20 computer. He's also frequently invited to shows, such as the PET Show in London he attended in June. He finds this part of his work "really great fun" because it provides opportunities for travel.

Lecturing and teaching, such as the machine language course he conducts each month for a special interest division of TPUG, provide him with feedback about problems and areas where people need more information. He has a reputation for being generous with his time, and his phone is open from 10 a.m. to 10 p.m. Monday to Friday. "If somebody phones me up and asks a question which shows they just haven't bothered trying it themselves, then I will sometimes be a little short, because it does seem like a waste of my time," he says. "But most people who call do so because they're stuck on something. It's just a question of getting another opinion. If I get a number of inquiries in a certain area, that's usually a signal that it's time for me to write an article about it. It's a very good way of keeping posted on what's bothering people at the moment."

Butterfield is equally generous with his software. He rarely sells any of his programs. "I would like to foster an environment where people pass out their software with reasonable generosity. I think that by showing a good example, I might sort of lead the way in that." Often he distributes his work on TPUG's library disk.

Still, Butterfield vehemently supports an author's copyright: "I believe very strongly that the person writing an original program has the right to do as he chooses with that program. If he chooses to sell it or to request that it not be copied except for a fee, then he has absolutely that right."

However, he feels that a person who takes money for software is obligated to support that program by upgrading it and furnishing the means to modify it, if necessary. "That's another good reason to give programs away. I really feel that most people who put down a lot of money for software feel that they are not buying a disk or cassette tape, but they are buying a service."

Interestingly, Butterfield believes the problem of software piracy might lessen, not grow, with the increasing business use of microcomputers. He laughs, "If an employee ran to the boss and said, 'Chief, I think you should give me a raise because I just saved you $500, I lifted a copy of a program,' I really don't think very many businesses would stick a cigar in my mouth and give me a promotion. They would more likely start keeping an eye on me."

Butterfield thinks that renting software eventually may be the best way to distribute it. A yearly fee could be charged for its use. In return, the user would receive continuing support on such things as upgrades, newsletters, information, warranty, and documentation.

Something Unprecedented In Education

Given his multiple interests in computing, writing, teaching, and making life easier for beginners, it's only natural that Butterfield is a strong advocate of introducing children to computers early in school. "As I understand the writings of Seymour Papert [author of *Mindstorms: Children, Computers, and Powerful Ideas*], the earlier a child becomes exposed to computers, the better it is likely to be," he says. "I have seen no evidence to contradict this. It seems to me that more important than anything formalized we teach young people about computers is


CBM™
Model 8032
commodore
computer
NEECO
JUST A SAMPLE OF THE MANY PRODUCTS WE CARRY, CALL US FOR OUR NEW 80-PAGE
WE WILL MATCH SOME ADVERTISED PRICES ON CERTAIN PRODUCTS LISTED

Tomorrow's Technology Today

Commodore Computer

VIC 20 Personal Computer	$ 299.95
VIC 1515 Graphic Printer	$ 395.00
CBM 4016	$ 995.00
CBM 4032	$1295.00
CBM 8032	$1495.00
CBM SuperPet 9000	$1995.00
CBM 8050 Dual Disk Drive	$1795.00
CBM 4040 Dual Disk Drive	$1295.00
CBM 2031 Single Disk Drive	$ 695.00
CBM 4022 Tractor Printer	$ 795.00
CBM 8010 IEEE Modem	$ 279.00
CBM Datasette	$ 74.95
CBM-IEEE Interface Cable	$ 39.95
IEEE-IEEE Interface Cable	$ 49.95

NEC Spinwriter Printer

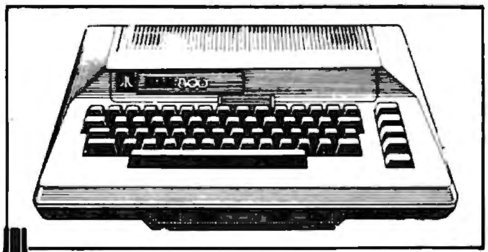
NEC 7730	$3085.00
NEC 7710	$3085.00
NEC 7720	$3610.00
NEC 3510	$2290.00
NEC 3500RD	$1895.00

Tractor Feed Options are available

Professional Software

WordPro™ — A Family of CBM Word Processing Programs

WordPro 2 Plus	$ 199.95
WordPro 3 Plus	$ 295.00
WordPro 4 Plus	$ 450.00
WordPro 5 Plus (for CBM 8096)	$ 450.00
The Administrator (DataBase for CBM)	$ 650.00
POWER™ (Programmer's Utility ROM)	$ 89.95
InfoPro™ (DataBase for CBM)	$ 295.00



Atari

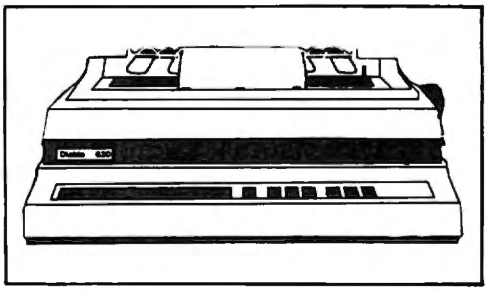
Atari 400 16K	$ 399.00
Atari 800 16K (incl. BASIC cartridge)	$ 899.00
Atari 410 Recorder	$ 99.95
Atari 810 Disk Drive	$ 599.95
Atari 822 Thermal Printer	$ 299.95

Amdek Monitors

Video 100G (Limited Quantity)	$ 179.00
Video 300G	$ 200.00
Color 1	$ 449.00
Color 2	$ 999.00

Diablo 630 Printer

Diablo 630	$2710.00
Tractor Option	$ 350.00



Epson Printers

MX-70	$ 299.95
MX-80 w/graphtrax	$ 645.00
MX-80 FT	$ 745.00
MX-100 FT	$ 945.00
INTERFACE CARDS	
8141 RS-232 Interface Board	$ 75.00
8145 RS-232 Interface Board w/2K Buffer	$ 149.00
8151 RS-232 Interface Board w/X/ON-X/OFF	$ 170.00
8161 IEEE Interface Board	$ 55.00
8131 Apple Interface Card	$ 85.00
8232 Apple Interface Cable	$ 35.00
8220 TRS-80 Cable	$ 35.00

CMD Mupet

MC-800A Mupet Controller (Multi-User Controller for CBM Computers)	$ 995.00
CM-100 Channel Module	$ 250.00
Printer Module	$ 350.00

Qume

Spring 9/45	$2495.00
Tractor Option	$ 210.00

WordPro, POWER and InfoPro are registered trademarks of Professional Software Inc.

NEECO
679 Highland Ave.
Needham, MA 02194

(617) 449-1760
TELEX: 951021

MON-FRI 9:00 - 5:30 E.S.T.

MASTERCARD • VISA

that we get them familiar with the concept, we get their fears allayed, we make sure that the usefulness of computers is understood at an early age. By the time a student gets to high school, computers are an oddity. There's something not quite natural about them – something manufactured and solemn. If you use computers in grade two or three you simply understand that they're around and they're going to help you whenever you feel like using them."

Teachers are faced with devising methods of guiding computer studies and providing resources for students, some of whom could soon outstrip them in programming ability. This can be an intimidating task, but student enthusiasm should make it stimulating and challenging as well. "We have in the microcomputer one of the most incredible forces that has ever happened in education," says Butterfield. "I'm not talking about games; games don't last very long. Students are begging for access to this logic device. It has no precedent. I don't know what specific educational objectives are precisely to be served. All I know is there must be something in the whole phenomenon, some need in the young mind that causes an intense urge to interface with the computer, to try things, to make the computer do things."

Part of the appeal, he believes, comes from the creative nature of programming. "Programming is creative not necessarily in the most visible sense. If you write yourself another *Space Invaders* it might end up looking like everybody else's. I sometimes like to compare programming, especially machine language programming, which is more exacting, to doing a jigsaw puzzle. Why would you sit there for two or three days and put in all this effort when you know that the end result will be a rather crummy-looking picture? The point is that you will have felt you have accomplished something, that you have brought together a number of skills, and even though it's the same as everyone else's, in a sense you have created it. It's the same thing with programming – you feel so good when it all comes together, when it all works."

Expert Debugging

But what if it doesn't work? When you're the ranking expert, what do you do when you get stuck on a problem? "Well, when you reach a certain stage, and it really isn't all that hard to achieve, then you have control of all parts of the machine. Once you get to that point, and there are many people who have achieved that, you don't have to ask anybody. You can go in there and look for yourself. One of the messages that I try to deliver to people is, 'If I can do it, you can do it.' Because often there isn't anything in the problem that logically you can't look at."

The Future Of Personal Computing

As personal computer enthusiasts grow wiser and more mature in the next few years, so will their machines, Butterfield predicts. Memory will be cheap, machines more powerful, and at the same time less expensive. The biggest single change will probably be a move toward better human interface. Full-screen editing, color, sound, and graphics will be almost universal and easier to use. Peripherals such as light pens, paddles or joysticks will simply plug in. Features such as upper/lowercase letters, now viewed as optional by some companies, will be standardized. There will be some moves toward better languages, but, Butterfield says, "BASIC appears to be indestructible at present."

More specifically, Butterfield offers some opinions on the future of microcomputer manufacturers: "I think we can say with some certainty that IBM will survive, not necessarily because of the merit of its products, but because IBM will gather around itself a massive amount of support. Radio Shack is very strong. Like IBM it will probably survive for reasons not directly associated with quality. This is not a reflection on its quality, but it has access to so many outlets of its own that it can support continuing sales. Atari has so far suffered from its games image.

"One of the most interesting phenomena could be Sinclair," he says. "Sinclair has introduced a series of small, not very powerful, but remarkably inexpensive computers. While people who are used to the speed of, say, a PET or a VIC would find some of the existing Sinclair computers very slow, we can't ignore the fact that Sinclair through Timex is going to sell an astonishing number of machines."

Butterfield foresees a very interesting battle between these less expensive machines, which are likely to be sold in every corner drug store, and the more powerful products. He notes that people tend to be loyal to a product line, and so far Sinclair's line has a clearly defined top end. Whether this situation will change as a result of demands from buyers of machines such as the ZX-81 who want to upgrade their systems remains a matter for speculation.

As computer prices drop, it is likely that people will begin to see a computer as an affordable tool for the family's financial management, entertainment, and education. Wider distribution of machines will affect society in several ways. Already, of course, people use home computers in a limited way for business, and more commonly for enjoyment and exercise of mental agility. "People test themselves against their computers by asking,

'Can I make the computer do this task?' People also go to the computer for something resembling relaxation.

"I was talking to a microcomputer owner who is having difficulties in his business. He told me he goes home, speaks to no one, and works on his computer for an hour or so. Only when he shuts the machine off does he say 'hello' to everyone. He finds the computer a very great pacifier in some sense – perhaps he takes his energies out on it. He feels that he comes out of that environment more of a human being, and his family is very understanding of it."

Butterfield also feels that people armed with the facts rather than the myth of computers are better equipped to cope with society.

"The most important change that small computers have brought is they have restored to the individual a sense that he has control over the events around him. Not only can his computer calculate a mortgage as well as his bank can, but he has control in that he will not simply accept any nonsense the computer prints and mails to him. Essentially, it's related to the question of competence. If you can handle these little beasts, then in one sense, at least, you are more competent. You understand more about some of the things which are happening in the world around you. That in itself is probably one of the most profound things microcomputers do."

As we become more aware of a computer's true capabilities and limitations, we also may better assess the complex arguments about artificial intelligence. Butterfield defines it very simply: "A computer which adapts its behavior based on what it has learned from external sources is showing artificial intelligence." He cites a game called "Animals" as a simple example of a program which learns from the user. "Animals says it will guess any animal you can name. The first few times, you're going to name an animal it has never heard of. It will ask you for more information about the animal and put it in its list. Eventually you will run out of animals you know, and then it will know as much as you do."

Videotex is another computer-based system with great possibilities for the future – one which he fears will not reach its potential. "I wish I could see a stronger future for videotex. Things like Telidon, Prestel and so on have a conceptual problem for me. They seem to be predominantly one-way only communications systems, perhaps a little bit like television, only not as effective. You have a few people communicating to a lot of people. I don't view that as a good move, or even a typical move in this day where people are getting competence in their own hands. I think that if Telidon were more of a two-way interface, if more people could contribute, then you might have more of what I would call a lively medium."

Rest And Diversion

Now that Butterfield finds himself constantly occupied with computers, he must force himself to get away from them for relaxation. Prowling around whatever city he happens to be visiting is one of his favorite diversions. He adds, "I do play the piano quite badly. Occasionally I go and dig dandelions out of the garden if I have time. But there is a little bit of change in the order of things. Since my hobby has become my work, I can't do it all the time."

In many ways, Butterfield has achieved celebrity status. He is much sought after by the microcomputer community around the world and does enjoy the travel. Yet he remains very approachable. "It's really great fun. It's nice to be invited over to England. But simply if any part of it is intimidating to others – if I hear people say 'Well, that's all right for Jim Butterfield' – then I feel ... not good. Essentially, what I'm trying to say is, 'If I can do it, you can do it.' " ©

KMMM Pascal for PET/CBM $85

A subset of standard Pascal with extensions.
- Machine language Pascal Source Editor with cursor oriented window mode
- Machine Language P-Code Compiler
- P-Code to machine language translator for optimized object code
- Run-time package
- Floating point capability
- User manual and sample programs

Requires 32K Please specify configuration.

EARL for PET (disk file based) $65

Editor, Assembler, Relocator, Linker

Generates relocatable object code using MOS Technology mnemonics. Disk file input (can edit files larger than memory). Links ,multiple object programs as one memory load. Listing output to screen or printer. Enhanced editor operates in both command mode and cursor oriented "window" mode.

RAM/ROM for PET/CBM

4K or 8K bytes of soft ROM with optional battery backup.

RAM/ROM is compatible with any large keyboard machine. Plugs into one of the ROM sockets above screen memory to give you switch selected write protectable RAM.

Use RAM/ROM as a software development tool to store data or machine code beyond the normal BASIC range. Use RAM/ROM TO LOAD A ROM image where you have possible conflicts with more than one ROM requiring the same socket. Possible applications include machine language sort (such as SUPERSORT), universal wedge, Extramon, etc.

RAM/ROM - - 4K	$85
RAM/ROM - - 8K	120
Battery Backup Option	30

SUBSORT by James Strasma $35

Subsort is an excellent general purpose machine language sort routine for PET/CBM computers. Sorts both one and two dimensioned arrays at lightning speed in either ascending or descending order. Other fields can be subsorted when a match is found, and fields need not be in any special order. Sort arrays may be specified by name, and fields are random length. Allows sorting by bit to provide 8 categories per byte. The routine works with all PET BASICs, adjusts to any memory size, and can co-exist with other programs in high memory.

SuperGraphics 2.0

NEW Version with TURTLE GRAPHICS

SuperGraphics, by John Fluharty, provides a 4k machine language extension which adds 35 full featured commands to Commodore BASIC to allow fast and easy plotting and manipulation of graphics on the PET/CBM video display, as well as SOUND Commands. Animations which previously were too slow or impossible without machine language subroutines now can be programmed directly in BASIC. Move blocks (or rocketships, etc.), or entire areas of the screen with a single, easy to use BASIC command. Scroll any portion of the screen up, down, left, or right. Turn on or off any of the 4000 (8000 on 8032) screen pixels with a single BASIC command. In high resolution mode, draw vertical, horizontal, and diagonal lines. Draw a box, fill a box, and move it around on the screen with easy to use BASIC commands. Plot curves using either rectangular or polar co-ordinates (great for Algebra, Geometry and Trig classes.)

The SOUND commands allow you to initiate a note or series of notes (or even several songs) from BASIC, and then play them in the background mode without interfering with your BASIC program. This allows your program to run at full speed with simultaneous graphics and music.

Seven new TURTLE commands open up a whole new dimension in graphics. Place the TURTLE anywhere on the screen, set his DIRECTION, turn him LEFT or RIGHT, move him FORWARD, raise or lower his plotting pen, even flip the pen over to erase. Turtle commands use angles measured in degrees, not radians, so even elementary school children can create fantastic graphic displays.

Specify machine model (and size). ROM type (BASIC 3 or 4)

SuperGraphics (disk or tape)	$ 40
SuperGraphics in ROM	55

Volume discounts available on ROM version for schools.

FLEX-FILE for PET/CBM Computers

Self Calculating
DATA BASE
REPORT WRITER
MAILING LIST

FLEX-FILE is a set of flexible, friendly programs to allow you to set up and maintain a data base. Print files with a versatile Report Writer or a Mail Label routine. Programmers will find it easy to add subroutines to their own programs to make use of Data Base files.

RANDOM ACCESS DATA BASE

Record size limit is 250 characters. The number of records per disk is limited only by the size of each record and the number of records per disk is limited only by the size of each record and the amount of free space on the disk. File maintenance lets you step forward or backward through a file, add, delete, or change a record, go to a numbered record, or find a record by specified field. The Find command locates any record when you enter all (or a portion of) the desired key. Field lengths may vary from record to record to allow maximum packing of information. Files may be sorted by any field, and any field may be specified as a key. Sequential files from other programs may be converted to Flex-File format, and Flex-File records may be converted to sequential (WordPro, PaperMate, other word processors may also use Flex-File data). Maximum record size, fields per record, and order of fields may be changed at any time.

MAILING LABELS

With typical record size of 127 characters, each disk can handle over 1000 records (about 2800 with 8050 drive). Labels may be printed any number wide, and may begin in any column position. There is no limit on the number or order of fields on a label, and two or three fields may be joined together on one line (like first name, last name, and title). A "type of customer"-field allows selective printing.

REPORT WRITER

Print any field in any column. For numeric fields, use decimal point justification (and round to any accuracy). Define any column as a series of mathematical functions performed on other columns. These functions include arithmetic operations and various log and trig functions. Pass results of operations such as running total from row to row. At the end of the report, print total and/or average for any column. Complete record selection, including field within range, pattern match, and logical functions can be specified individually or in combination with other parameters.

FLEX-FILE BY Michael Riley $80

Please specify equipment configuration when ordering.

COLORCHART Color Video Board $125

Up to 8 colors and high resolution graphics (128 x 192 in 2 colors). Generates composite video for use with monitor

PEDISK II from cgrs Microtech

5" 40 track, 1 drive, 143K	$525
5" 40 track, 1 drive, 286K	690
8" IBM 3740 format, 77 track, 250K	995

PROGRAM YOUR OWN EPROMS

Branding Iron for PET/CBM $79

EPROM Programmer with software for all ROM versions. Includes all hardware and software to program or copy 2716 and 2532 EPROMs.

PORTMAKER DUAL RS232 SERIAL PORT 63

Two ports with full bipolar RS232 buffering. Baud rates from 300 to 4800 For PET/CBM, AIM, SYM

CBM Software

Legal Time Accounting Package	445
Medical Accounting Package	
Dow Jones Portfolio Management	129
Personal Tax Calculator	65
Wordcraft 80 Wordprocessor Package	325
Pascal Development Package	235
Assembler Development Package	99
Intelligent Terminal Emulator	30
Softpac-1 (Competitive Software)	29

FORTH for PET

BY L. C. Cargile and Michael Riley $50

Features include:
- full FIG FORTH model.
- all FORTH 79 STANDARD extensions.
- structured 6502 Assembler with nested decision making macros.
- full screen editing (same as when programming in BASIC).
- auto repeat key.
- sample programs.
- standard size screens (16 lines by 64 characters).
- 150 screens per diskette on 4040, 480 screens on 8050.
- ability to read and write BASIC sequential files.
- introductory manual.
- reference manual.

Runs on any 16K or 32K PET/CBM (including 8032) with ROM 3 or 4, and CBM disk drive. Please specify configuration when ordering.

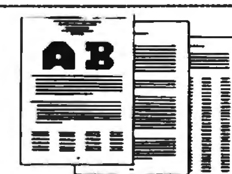
Available soon:

Metacompiler for FORTH $30

simple metacompiler for creating compacted object code which can be executed independently (without the FORTH system).

PaperMate 60 COMMAND WORD PROCESSOR

by Michael Riley



Paper-Mate is a full-featured word processor for CBM/PET. Paper-Mate incorporates 60 commands to give you full screen editing with graphics for all 16K or 32K machines (including 8032), all printers, and disk or tape drives.

For writing text, Paper-Mate has a definable keyboard so you can use either Business or Graphics machines. Shift lock on letters only, or use keyboard shift lock. All keys repeat.

Paper-Mate text editing includes floating cursor, scroll up or down, page forward or back, and repeating insert and delete keys. Text block handling includes transfer, delete, append, save, load, and insert.

All formatting commands are imbedded in text for complete control. Commands include margin control and release, column adjust, 9 tab settings, variable line spacing, justify text, center text, and auto print form letter (variable block). Files can be linked so that one command prints an entire manuscript. Auto page, page headers, page numbers, pause at end of page, and hyphenation pauses are included.

Unlike most word processors, CBM graphics as well as text can be used. Paper-Mate can send any ASCII code over any secondary address to any printer.

Paper-Mate functions with 16/32K CBM/PET machines, with any printer, and with either cassette or disk.

To order Paper-Mate, please specify configuration.

Paper-Mate on disk or tape 40.00

PaperMate works on 16K VIC Computer also

BASIC INTERPRETER $200

Designed to support the CBM 8096 (8032 with add-on 64K board). A full interpreter implementation to automatically take advantage of the extra memory available to the 8032.

BPI General Ledger – 8032/8050	300
BPI Accounts Receivable – 8032/8050	300
BPI Inventory – 8032/8050	300
BPI Job Costing	300
BPI Payroll	300

JINSAM Data Base Management System for CBM. Comprehensive version available for most configurations.

COPY-WRITER Word Processor 159

Works like expensive word processors, plus has added features like double column printing, and shorthand generator. For PET/CBM and Apple

CASH MANAGEMENT SYSTEM $45

Easy to use. Keeps track of cash disbursements, cash receipts, cash transfers, expenses for up to 50 categories.

252 Bethlehem Pike
Colmar, PA 18915 215-822-7727 **A B Computers**

WRITE FOR CATALOG

Add $1.25 per order for shipping. We pay balance of UPS surface charges on all prepaid orders. Prices listed are on cash discount basis. Regular prices slightly higher. Prices subject to change.

There are some solid arguments in favor of avoiding BASIC's INPUT command. This article offers some suggestions and substitutions for Apple, PET/CBM and VIC users.

Banish INPUT Statements!

Richard Cornelius
Department of Chemistry
Wichita State University
Wichita, KS

Nearly any useful microcomputer program requires the user to enter information through the keyboard. Game paddles, joysticks, light pens, and voice recognition all have their places, but rarely do these techniques displace keyboard input entirely in a program. Given the importance of keyboard input, attention should be given to handling it properly.

On microcomputers, the BASIC language supports two kinds of statements that can be used to transfer information from the keyboard to the program: GET and INPUT. Each of these statements has its strong points and its weak points. The principal advantage of the INPUT statement is that it is easy to use. Unfortunately, it has a serious weakness: when an INPUT statement is used, control of the program is relinquished to the computer. Until RETURN is pressed, the computer is in control and may do some things that are not desirable.

The dangers of INPUT are so great that I have totally abandoned its use when writing programs that others will use. In this article I will tell you why I don't use it and will show you how I do what INPUT statements do (and do it better) by using GET statements. My experience is limited primarily to the PET and Apple II computers, but the same principles can be applied to other computers as well. [*References here to "PET" apply as well to the VIC.*]

The INPUT Statement

First let us examine the INPUT statement to see why it is troublesome. The simplest form of the INPUT statement is illustrated by the following statement:

```
100 INPUT N
```

On either the Apple or the PET, when the program reaches this statement a question mark and a flashing cursor appear on the screen (provided, of course, that on the Apple text page one is being displayed), and program operation comes to a halt. The user may type in nearly any character, but the computer takes no action until RETURN is pressed. While the computer waits for that magic RETURN key stroke, even the STOP key on the PET or CTRL-C on the Apple are inoperative. When RETURN is pressed, the variable N is set to be whatever number was typed in. A more useful form of the input statement provides a prompting statement for the user. A typical statement might look like this on the Apple:

```
110 INPUT "ENTER YOUR NAME, PLEASE: ";
    NAME$
```

When the quotation marks are used on the Apple, the question mark of the INPUT statement is suppressed. If you want a question mark to appear, you can always add it within the quotation marks. In the sample statement 110 no question mark has been used since no question is asked. A space is placed within the quotation marks after the colon to prevent the prompting statement and the user's input from running together. On the PET the question mark cannot be suppressed. The best approach (if you must use INPUT statements) is to use a prompting statement in the form of a question. On the PET computer the function of statement 120 might be better served by a statement such as this one:

```
120 INPUT "WHAT IS YOUR NAME"; NAME$
```

When the program encounters this line, it will print on the screen

```
WHAT IS YOUR NAME?
```

because of the mandatory question mark provided by the computer.

Each of the BASIC statements that has been presented also results in the presence of a flashing cursor after the input prompt. The presence of the cursor on the screen can be highly distracting and may or may not be desirable. Programmers need a cursor during editing procedures and may be conditioned to expect one on the screen, but the best practice is for the programmer to decide whether a cursor should appear. The use of the INPUT statement removes that decision from the hands of the programmer.

A Greater Weakness

Although the removal of control from the programmer is undesirable, a more significant shortcoming of the INPUT statement is that it wrests control from the user. The INPUT statement essentially halts all action by the computer until RETURN is pressed. After RETURN is pressed,

the computer examines what has been entered and, if it can, assigns the data that have been entered to the variable in the INPUT statement. The computer may also take some other action, depending upon what has been entered. For example, if "THREE" (followed by a RETURN) is typed as a response to the question mark resulting from statement 100, the computer will respond with

?REENTER (on the Apple)

or

?REDO FROM START (on the PET)

If the number 11,324 is typed and then RETURN is pressed, the computer will come back with

?EXTRA IGNORED

because it understands the comma as a delineator which ends one input value and gets ready for the next. It tries to interpret the number 11,324 as the number 11 followed by the number 324.

Statements 110 and 120, which use the string variable NAME$, do increase somewhat the control the programmer has in formulating a response, since either numbers or letters may be typed in. However, if the user types in her name as "DOE, JANE" the computer will once again respond

?EXTRA IGNORED

as it assigns "DOE" to NAME$ and discards "JANE", for which it had no variable name. These error messages may be familiar and decipherable to a computer programmer, but they will likely appear strange and cryptic to the uninitiated.

Not only do these computer-generated responses perplex the user, but they also defy the programmer. If the user, after finding "THREE" an unsatisfactory input, tries "FOUR", "FIVE", etc., the text on the screen may be scrolled off the screen into oblivion. On the Apple the text window could be defined (by using POKEs to positions 34 and 35) to a single line, but then the coding becomes awkward and tedious. Furthermore, no additional explanatory message can be provided to aid the user. The only messages that will appear are those that the computer generates. As a result, the user cannot control the program because the program cannot control the machine.

Other related problems also occur. For example, the cursor control keys on the PET remain active while the INPUT statement has control of the machine. They can be used to move the cursor any place on the screen. If the insert key is pressed before the cursor control keys, the results can be totally meaningless to the novice user. Finally, CLEAR can wipe out the entire screen. On the Apple the left arrow key is used to back up in order to reenter characters, but if it is pressed before any characters are entered or after the cursor is moved back to the original spot, the cursor moves to the beginning of the next line. These movements are beyond the control of the programmer. Thus, the user may inadvertently enter data someplace other than where the programmer intended and could destroy a portion of a display that was intended to remain on the screen.

Most **COMPUTE!** readers have enough computer experience to understand how the problems with the INPUT statement arise and, as a result, will probably get into difficulty only rarely. The person just beginning, however, may become discouraged when greeted by uninterpretable responses and by disrupted screen images. That person may not come back to the computer again or may return less frequently. Even the experienced user will find that programs free of these problems are easier and more enjoyable to work with. The moral of the story is that if you want your programs to be widely used (who doesn't?), then steer clear of INPUT statements.

The GET Statement

INPUT is the only BASIC command that directly accepts multi-key entries and assigns to variables the values entered from the keyboard. If INPUT is to be avoided, what can be used to replace its function? The answer is the GET statement. Do you protest that GET is only for single character entry of data? The one-character-at-a-time feature of GET is precisely its advantage. By working this advantage to the fullest, the INPUT statement can be simulated and greatly improved upon.

The GET statement operates differently on the different computers. On the Apple the GET statement halts the operation of the program and displays the flashing cursor until a key is pressed. Any key press (except for the SHIFT or CTRL keys by themselves) will continue the operation of the program. The cursor appears (if the text screen on the Apple is showing) while the computer awaits action at the keyboard. On Commodore machines the GET statement works quite differently. When the statement is encountered in the program, the program is not halted. Instead, whichever key has been pressed (if any) is transmitted to the variable in the GET statement.

Although important differences exist in the way that the GET statement is handled on the two different machines, probably neither way should be regarded as superior. Each approach has its advantages and disadvantages. Perhaps more important is that the way that one computer handles the GET statement can be imitated by appropriate statements on the other computer. To make the computer halt its action, show a flashing cursor,

VOICE I/O THAT WORKS!

for the COMMODORE and APPLE II computers

Voice I/O has come a long way from the barely intelligible computer speech of only a few years ago. It is now possible to enter data or commands to your computer just by talking to it and the computer can talk back with clear, pleasant, human sounding voice.

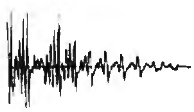
The COGNIVOX models VIO-1002 (for Commodore) and VIO-1003 (for the Apple II+) are at the forefront of a new generation of Voice I/O peripherals that are easy to use, offer excellent performance and are affordably priced.

SOME SPECIFICATIONS

COGNIVOX can be trained to recognize up to 32 words or phrases chosen by the user. To train COGNIVOX to recognize a new word, you simply repeat the word three times under the prompting of the system.

COGNIVOX will also speak with a vocabulary of 32 words or phrases chosen by the user. This vocabulary is independent of the recognition vocabulary, so a dialog with the computer is possible. Memory requirements for voice response are approximately 700 bytes per word.

For applications requiring more than 32 words, you can have two or more vocabularies and switch back and forth between them. Vocabularies can also be stored on disk.



HOW IT WORKS

COGNIVOX uses a unique single-chip signal processor and an exclusive non-linear pattern matching algorithm to do speech recognition. This gives reliable operation at low cost. In fact, the performance of COGNIVOX in speech recognition is equal or better to units costing many times as much.

For voice output, COGNIVOX digitizes and stores the voice of the user, using a data compression algorithm. This method offers four major advantages: First there are no restrictions to the words COGNIVOX can say. If a human can say it, COGNIVOX will say it too. Second, it is very easy to program your favorite words. Just say them in the microphone. Third, you have a choice of voices: male, female, child, foreign. Fourth and foremost, COGNIVOX sounds very, very good. Nothing in the market today can even come close to the quality of COGNIVOX speech output. You can verify this yourself by calling us and asking to hear a COGNIVOX demo over the phone. Hearing is believing.

A COMPLETE SYSTEM

COGNIVOX comes assembled and tested and it includes microphone, software, power supply, built in speaker/amplifier and extensive user manual. All you need to get COGNIVOX up and running is to plug it in and load one of the programs supplied.

It is easy to write your own talking and listening programs too. A single statement in BASIC is all that you need to say or recognize a word. Full instructions on how to do it are given in the manual.

COGNIVOX model VIO-1002 will work with all Commodore computers with at least 16k of RAM. Model VIO-1003 requires a 48k APPLE II+ with 1 disk drive and DOS 3.3.

ORDER YOUR COGNIVOX NOW

Call us at (805) 685-1854 between 9am and 4pm PST and charge your COGNIVOX to your credit card or order COD. Or send us a check in the mail, specifying your computer. Price for either model of COGNIVOX is $295 plus $4 shipping in the U.S. (foreign add 10% we ship AIR MAIL).

VOICETEK

Dept R, P.O. Box 388 Goleta, CA 93116

Also available for the AIM-65.
Call or write for details.

VIC-20

VIC-20 INTERFACING BLUE BOOK

Did you know that your VIC can be used to control a 99¢ toy motor so effectively that it runs like a precision machine? Or that you can build an accurate digital thermometer using the VIC and four other parts costing less than $5?

These and other 18 interfacing projects selected for usefulness, ease of construction and low cost are detailed in the VIC-20 Interfacing Blue Book, a veritable gold mine of practical information on how to build a variety of interfaces for your computer.

Projects include: Connecting VIC to your stereo; Pickproof digital lock; Capacitance meter; Liquid level sensor; Telephone dialer; Voice output; 8K/16K RAM/ROM expansion; 128K RAM expansion; 8-bit precision D/A; 8-bit A/D converter; MX-80 printer interface and more.

Written by a college professor in a friendly and informative style, the Blue Book gives you theory of operation, schematics, program listings, parts list, construction hints and sources of materials for each one of the 20 projects.

If you want to get the most out of your VIC this book is a must. Even if you don't plan to build any of the projects, the Blue Book is a valuable source of information on what can be done with the VIC. Cost is $14.95 (less than 75¢ per project!).

WORD WHIZ

Here is a no-frills word processor that does the job and is so small it leaves plenty of memory for your text. Yet it offers full screen editing and easy save of work in progress on cassette, by taking advantage of VIC's built-in text manipulation capabilities. WORD WHIZ prints out on the 1515 printer and is a bargain at $9.95.

WORD WHIZ/80

For classy looking output, this version of WORD WHIZ will drive an EPSON MX-80 (See interfacing info in Blue Book above). Get letter quality printing for only $14.95.

Above prices include postage in the U.S. CA res. add 6% tax. Foreign add $2.

micrDsignal

900 Embarcadero Del Mar, Unit A
Goleta, CA 93117

VIC-20

CBM™ CATALOG

Now there is a catalog reporting system for COMMODORE CBM computers.

FEATURES:

- realtime inquiry
- report generator
- enhanced directory function
- wildcard search for volumes and files
- written in basic
- fast and easy to use

cat no. r110-v01 $19.95
specify 4040, 8050, 8250

MSC software
1513 Aviation Blvd.
suite a1333
Redondo Beach, CA 90278

trademark
Commodore Business Machines

New Product

FOR COMMODORE SYSTEMS

The Commander

This 4K ROM contains exclusive programmable commands. These powerful commands contain an enhanced **COMMON** function which **RETAINS ALL VARIABLES AND ARRAYS.**
A list of some of these **COMMANDS WITH COMMON,** which until now were only available on large systems, are **INSERT, DELETE, APPEND, and RE-DIMENSION.**

INSERT - Loads a program or subroutine into the beginning, middle, or between specific line numbers of a running program, without losing variables or arrays. Program execution will continue at any line number, even a new line number just inserted. Insert also allows inserting any part of a program or subroutine.

DELETE - Deletes any portion of the running program between specified line numbers, under program control, with **COMMON** function, and continues execution. All deleted memory is reclaimed, and all variables/arrays are retained.

APPEND - Appends another program or subroutine (or any part thereof) to the end of the running program, and continues execution without losing variables.

RE-DIMENSION - Allows dynamic re-dimension of arrays, while program is running, without losing variables or any array data.

Also included are: **ENHANCED GET, STRING, FRAME, PRINTUSING and IMAGE, RETURN CLEAR, WINDOW, SPEED DATA** and **OVERLAY** commands.

Your Commodore needs THE COMMANDER©. These new commands give the Commodore system powerful features, all under program control. The commands are flexible and easy to use, in either program or direct mode.

THE COMMANDER©: $70.00

(Includes demo/subroutine diskette)
(Florida residents add 5% sales tax)
(Specify socket: $9000 or $A000)

master charge

METRON COMPUTER SYSTEMS
4505 Jackson Street • Hollywood, FL 33021
305-962-5183

VISA

and wait for a keystroke on the Apple requires a single statement:

140 GET G$

On the PET we can halt program action by creating a loop which is closed as long as no key is pressed:

150 GET G$
160 IF F$ = "" THEN 150

Creating a flashing cursor requires a little more work, but that too can be done:

```
200 CURSER$(1) =CHR$(18) + " " + CHR$(146) + CH
    R$(157)
210 CURSER$(2) = " " + CHR$(157)
220 PASS = 1
230 DELAY = 0
240 PRINT CURSER$(PASS);
250 DELAY = DELAY + 1
260 GET G$
270 IF G$ <> "" THEN 310
280 IF DELAY < 24 THEN 250
290 PASS = 3 - PASS
300 GOTO 230
310 PRINT G$
```

In this routine CURSER$ is spelled with an "E" to prevent the syntax error which arises if the variable name CURSOR$ were used. The error is due to the embedded BASIC word "OR" in "CURSOR$". CURSER$(1) is defined as a reverse field character [CHR$(18)] plus a space (" ") plus a reverse field off character [CHR$(146)] plus a cursor left [CHR$(157)]. CURSER$(2) is defined as a space plus a cursor left character. When CURSER$(1) is printed a white block appears, but the position of printing is moved back to the starting position. When CURSER$(2) is printed a space appears, and the printing position is shifted as for CURSER$(1). Thus, depending upon the number of times that the program has PASSed through statement 230, a white block is either printed or eliminated. Statements 250 through 280 cycle for a suitable time interval for the flash, all the while looking to see whether a key has been pressed. Changing the value of the constant in 270 changes the speed at which the cursor flashes.

To have the computer check "on the fly" whether a key has been pressed, and to record which key has been pressed, requires only a single statement on either machine. On a Commodore computer:

400 GET G$

On the Apple the same action can be accomplished with the aid of an IF statement:

410 IF PEEK (49152) >127 THEN GET G$

The PEEK to location 49152 checks to see whether a key has been pressed. If a key has been pressed, the value of that location will be 128 or greater.

GET Replaces INPUT

Now that both the INPUT and the GET statements have been examined, it is time to use GET statements to mimic the desirable features of the INPUT statement while eliminating all of the undesirable ones. Input subroutines for the Apple and the PET are given in Programs 1 and 2. The power of these subroutines is that they examine each character *before* it is printed. Characters that are "unacceptable" to the program are never printed to the screen.

Because of the way these particular subroutines are written, only the digits zero through nine and, depending upon the computer, the left arrow key or the DEL key, are acceptable keystrokes. Even the use of the latter keys is restricted so that they are inoperative if the length of the input string (INPUUT$) is zero. Simple changes in IF statements in these subroutines could be made to allow letters but not numbers, or to accept whichever characters are meaningful within the context of the program. Each of these subroutine listings will now be examined in detail.

In the first program, which is for the Apple computer, the first executable statements are in the section labelled "Initialization." Statements 1100-1120 define string variables with easily recognizable names using the CHR$ function. The variable BELL$ will be used to sound the built-in speaker to indicate invalid entry. The CHR$(7) used in defining BELL$ is a CTRL-G, which can even be used from the keyboard to make a bell sound. LEFTARROW$ is the left arrow key (or CTRL-H), which will be used to back up and make changes. The name REETURN$ has an extra "E" in it so that the BASIC interpreter will not give the syntax error that would result from having the valid BASIC word RETURN embedded in the name of a variable. ESC$ is defined as the ESCape key. The name DEELEET$ is another name in which extra E's prevent the generation of syntax errors (due to DEL and LET). When DEELEET$ is printed, the computer backspaces, prints a space, and then backspaces again so that the net effect is to shorten the input data by one character and reposition the (hidden) cursor.

The "Main Program" section prints out statements describing what the program does and then, in statement 1270, calls the input subroutine. Since the five-digit number requested is entered as a string variable in the subroutine, statement 1280 defines a number variable having the value of the number which was entered. When the program continues on to say "PRESS ANY KEY TO CONTINUE," statement 1330 prevents the cursor from appearing while the computer waits for a keystroke.

The heart of the program is the input subroutine which begins in statement 1400. Two initialization steps are the first actions taken. The POKE in statement 1410 clears any keystroke that might have been entered before the subroutine was entered, and the variable (INPUUT$) in statement 1420 is initialized to contain nothing. The PEEK in statement 1430 is the same as in statement 1330 and prevents the cursor from appearing on the screen. Statements 1450 to 1480 control the program flow in response to various keystrokes: if RETURN is pressed then the program RETURNs from the subroutine; if ESC is used, the program terminates; if the left arrow is pressed, the routine at statement 1600 is executed, and if something other than a number is typed, the program simply looks for another key. If a number is typed, then the program checks the number of digits that have been entered (statement 1490), prints out whatever member has been entered (statement 1500), and builds the input string INPUUT$ (so named to prevent a syntax error from the presence of INPUT in a variable name). Statements 1600 through 1650 backspace and delete a character both from the input string and from the screen (unlike a true INPUT statement) provided that something appears on the screen already.

Writing the version of the sample program on the PET is a little neater thanks to the DELete key on the keyboard. This key gives an ASCII code of 20 and in statement 2120 is assigned to the variable DELEETE$. Unlike on the Apple, DEL is not a valid BASIC word, so a single "E" can follow the "D" in DELEETE$. The structure of the two programs is the same, but small differences exist in the details. The mechanism for clearing the input buffer in the PET version is simply repeating the GET G$ statement ten times. The statements 2450 and 2460 make the program wait for a key to be pressed before going on. The only other significant difference is that "Q" for "Quit" is used to get out of the routine at any time. No parallel to the Apple's ESCape key exists on the 40-column PET.

When these routines are used in place of INPUT statements, the program retains control of the machine and, thus, the programmer can give control of the program to the user. In the specific example shown, commas cannot be a confusing issue because they never even appear on the screen. If, of course, a programmer wants commas to be accepted, then simple changes in the IF statements in the subroutines can be used to effect those changes. In addition, single keystrokes (the ESCape key on the Apple and "Q" on the PET) end the program. It is not necessary to hit RETURN as it would be if an INPUT statement were used.

Users operating a program containing GET-simulated INPUT statements may still be able to enter data which has no meaning to the program. The advantage over a standard INPUT statement is that these invalid entries can be handled by the program itself. Invalid entries with the actual INPUT statement may be totally beyond the control of the programmer and, therefore, the user can be powerless.

Program 1. PET Version

```
2100 REM **INITIALIZATION**
2110 REM
2120 DELEETE$ = CHR$(20)
2130 REETURN$ = CHR$(13)
2140 MAXLNGTH = 5
2200 REM
2210 REM **MAIN PROGRAM**
2220 REM
2230 PRINT CHR$(147)
2240 PRINT "HERE IS AN EXAMPLE OF HOW GET STATE
     MENTS"
2250 PRINT "CAN BE USED TO MIMIC AN INPUT STATE
     MENT."
2260 PRINT "PRESS 'Q' TO QUIT."
2270 PRINT : PRINT
2280 PRINT "ENTER A FIVE DIGIT NUMBER: ";
2290 GOSUB 2400: REM INPUT SUBROUTINE
2300 NUMBER = VAL(INPUUT$)
2310 PRINT: PRINT
2320 PRINT "YOU HAVE ENTERED THE NUMBER";NUMBER
     ;DELEETE$;"."
2330 PRINT
2340 PRINT "PRESS ANY KEY TO CONTINUE."
2350 GET G$
2360 IF G$ = "" THEN 2350
2370 IF G$ = "Q" THEN PRINT: PRINT "DONE.": END

2380 GOTO 2200
2400 REM
2410 REM **INPUT SUBROUTINE**
2420 REM
2430 FOR I = 1 TO 10: GET G$: NEXT:REM CLEARS I
     NPUT BUFFER
2440 INPUUT$ = ""
2450 GET G$
2460 IF G$ = "" THEN 2450
2470 IF G$ = REETURN$ THEN RETURN
2480 IF G$ = DELEETE$ THEN 2600
2490 IF G$ = "Q" THEN PRINT: PRINT: PRINT "DONE
     .": END
2500 IFG$ < "0" OR G$ > "9" THEN GOTO 2450
2510 IF LEN(INPUUT$) >= MAXLNGTH THEN GOTO 2450

2520 PRINT G$;
2530 INPUUT$ = INPUUT$ + G$
2540 GOTO 2450
2600 REM
2610 REM **DELETE IS PRESSED**
2620 REM
2630 IF LEN(INPUUT$) = 0 THEN GOTO 2450
2640 IF LEN(INPUUT$) = 1 THEN PRINT DELEETE$;:
     GOTO 2440
2650 INPUUT$ = LEFT$(INPUUT$,LEN(INPUUT$)-1)
2660 PRINT DELEETE$;
2670 GOTO 2450
```

Program 2. Apple Version

```
1100 REM  **INITIALIZATION**
1110 BELL$ = CHR$ (7)
```

```
1120 LEFTARROW$ = CHR$ (8)
1130 REETURN$ = CHR$ (13)
1140 ESC$ = CHR$ (27)
1150 DEELEET$ = LEFTARROW$ + " " + LEFTARROW$
1160 MAXLNGTH = 5
1200 REM  **MAIN PROGRAM**
1210 HOME : VTAB 3
1220 PRINT "HERE IS AN EXAMPLE OF HOW GET STATE
    MENTS"
1230 PRINT "CAN BE USED TO MIMIC AN INPUT STATE
    MENT."
1240 PRINT "USE ESCAPE TO EXIT THE PROGRAM."
1250 PRINT : PRINT
1260 PRINT "ENTER A FIVE DIGIT NUMBER: ";
1270 GOSUB 1400: REM INPUT SUBROUTINE
1280 NUMBER = VAL (INPUUT$)
1290 PRINT : PRINT
1300 PRINT "YOU HAVE ENTERED THE NUMBER ";NUMBE
    R;"."
1310 PRINT
1320 PRINT "PRESS ANY KEY TO CONTINUE."
1330 IF PEEK (49152) < 128 THEN 1330
1340 GET G$
1350 IF G$ = ESC$ THEN PRINT : PRINT "DONE.": E
    ND
1360 GOTO 1210
1400 REM  **INPUT SUBROUTINE**
1410 POKE 49168,0: REM CLEARS KEYBOARD STROBE
1420 INPUUT$ = ""
1430 IF PEEK (49152) < 128 THEN 1430
1440 GET G$
1450 IF G$ = REETURN$ THEN RETURN
1460 IF G$ = ESC$ THEN PRINT : PRINT : PRINT "D
    ONE.": END
1470 IF G$ = LEFTARROW$ THEN 1600
1480 IF G$ < "0" OR G$ > "9" THEN PRINT BELL$;:
     GOTO 1430
1490 IF LEN (INPUUT$) > = MAXLNGTH THEN PRINT B
    ELL$;: GOTO 1430
1500 PRINT G$;
1510 INPUUT$ = INPUUT$ + G$
1520 GOTO 1430
1600 REM  **LEFT ARROW IS PRESSED**
1610 IF LEN (INPUUT$) = 0 THEN PRINT BELL$;: GO
    TO 1430
1620 IF LEN (INPUUT$) = 1 THEN PRINT DEELEET$;:
     GOTO 1420
1630 INPUUT$ = LEFT$ ( INPUUT$, LEN (INPUUT$) -
     1)
1640 PRINT DEELEET$;
1650 GOTO 1430
```

©

For the Texas Instruments 99/4A computer, this computer-assisted game could be entertainment at a party or a teaching tool. Requires less than 7000 bytes.

CHARADES

Stephen Davis
Dallas

Remember the "good old days" when you had friends over for a party and you all ended up playing games? These days, you probably have your guests lined up at your computer to take turns at Invaders or some other one-man game.

Here is a chance to use your computer to help you entertain the whole group with the old parlor game charades. Charades has always been popular as a party game because it allows many players to participate, it is lively and fast-moving, and it generally leads to a good time.

Fun And Educational

As well as being a wild and crazy adult game, charades can be great entertainment for the youngsters as well. By altering the data statements in the program, you can substitute a custom list of words or phrases on any number of subjects, making the program quite versatile as an educational tool.

In the original game, you needed someone to keep time and score, and you had to take time to think of phrases, write them on slips of paper, and draw them out of a hat. Now the computer can take care of all those chores for you. For those who have never played, and for those who are just a little rusty, here is a brief rundown of the rules:

Although as few as two can play, it is suggested that an even number of players of six or more participate because the group will be divided into two teams. Players from each team take turns pantomiming phrases to be guessed by the other members of their team. Phrases may be broken into words or syllables, but the player may not talk, write, or form words with his lips while he is pantomiming. He has two minutes to convey the phrase to his teammates, and the time he uses determines his score.

A player should begin by pantomiming the category of the phrase. Categories used in this program include Movies, Books, People, Songs, and Quotes and Clichés. Some of the traditional signals for these categories in charades are:

Movie – Hold one hand in front of your face and turn the other one in a circle, as if cranking an old-time movie camera.

Book – Put hands together, as if praying, then open them like a book.

Song – Hold arms out and open mouth, as if singing.

Person – Pat yourself on the head.

Quote – Hold hands out with two fingers out on each, as if putting quotes around something.

Holding up a certain number of fingers indicates the number of words in the phrase, which word you are acting, or the number of syllables in a word. Pinching your ear means that the word you are acting "sounds like" the one in your phrase. If you are creative, you will be good at this game.

This program displays the phrase for each player to study before he pantomimes it, so situate your TV screen so that your teammates cannot see it. However, turn up the volume because the program provides an audible "time's up" tone (just like the one on TV game shows that so rudely informs the contestant that, indeed, she did *not* win the washer and dryer).

The program listed here is written in TI BASIC for the Texas Instruments 99/4A Home Computer. It takes advantage of several special routines that the TI offers, including sound capabilities that not only provide audible prompts, but also make timing loops as accurate as possible. Most of the commands can be easily converted to other BASICs, but explanations of the various routines and their functions are documented below. This program includes 125 phrases, and, including data, consumes less than 7000 bytes.

If you get hooked on this game, you may want to substitute your own phrases for variety. Adding words that are of interest to your group (i.e., computers), foreign words, or even X-rated terms, presents all kinds of possibilities. Let your micro liven up your next party with this new slant on an old game.

Learning How The Program Works

Line 110 sets up arrays for 125 phrases (M$) and a counter to check for duplications (Z). Line 120 assures a different set of random numbers for each game, and the counter "Q" keeps track of how many phrases have been played. CALL CLEAR simply clears the screen. The PRINT statements (and long strings in DATA statements) have been composed for the 28-character screen display of the TI.

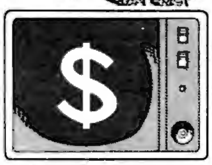
The colon in TI PRINT statements indicates a carriage return so that the command PRINT doesn't have to be repeated for each vertical space or new line of text. The subroutine at 1460 (referred

We've grown on you

Because Our Programs Deliver What They Promise.

With our sophisticated programs, you're never disappointed. *And*, our customer service is unparalleled in excellence.

Money Manager



By Andrew Bartorillo

In today's economy, we all need to monitor our expenses more closely. **Money Manager** will help you keep track of your income and expenses and give you an easy way to manage your budget. You define the categories according to your needs, including tax-deductible expenses . . . a great help at tax time. You can also reconcile your checkbook with the bank's balance. Full lineprinter capability allows you to print items by each category.

32K (Minimum) Disk: Atari; TRS-80 I & III; 64K: IBM. **$39.95**

plus $2.00 shipping and handling.

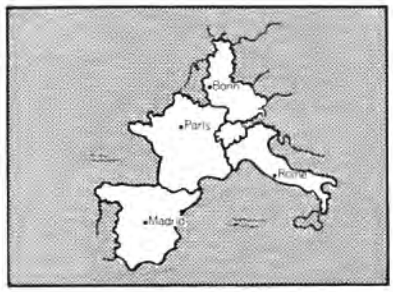
Everest Explorer

By William Godwin & Don Knowlton

Will you be one of the daring few to make it to the pinnacle of Mount Everest? Or will you succumb to the lack of oxygen, the unexpected violence of the storms, the incredibly rough terrain? You, as the leader of the Everest expedition, will have to choose the route, choose the timing, make sure your climbers are well-rested, set up a chain of camps and, if you reach the summit, get your followers back down to base camp. You'll have to manage money, climbers, Sherpas, tents, oxygen, food and fuel. Danger lurks at every step—can you get to the summit and return? "Save the game" feature on disk.

16K Tape or 32K Disk: Atari; TRS- 80 I & III; IBM. **$19.95**

plus $2.00 shipping and handling



Language Teacher

By Cindy & Andrew Bartorillo

Teachers, students, and those who just want to learn a new language—Acorn's Language Teacher gives you your choice of French or Spanish.
Use these programs at home or in the classroom. Students like this new way of learning and/or reviewing a previously learned language. Teachers can free themselves from tedious preparation of vocabulary lists and writing quizzes.
Each Language Teacher offers hundreds of word combinations, verb conjugations and phrases. Choose the drill topic and foreign language-to-English or vice versa. Option for multiple-choice answers plus retest on missed items. Gives running percentage of correct answers.
Available: French, Spanish I & II.

32K Disk: Atari; 64K: IBM. **$29.95**

plus $2.00 shipping and handling

Lost Colony

By David Feitelberg

You are the Economic Manager of the world's first space colony. The next support ship from Earth isn't due for another 15 years and you have instructions either to make things go better or get out of office in shame. You must assure the survival of this struggling space colony—it's all up to you. You'll be presented with the human, natural and industrial resources of the planet. You must allocate labor, explore new territories, decide on production quotas, determine pay scales and taxes for the most productivity —you're armed with maps and charts. 10 levels of difficulty. "Save the game" feature on disk.

16K Tape or 32K Disk: TRS-80 I & III; 40K: Atari; 48K: IBM. **$29.95**

plus $2.00 shipping and handling

Acorn Software Products, Inc.
634 North Carolina Avenue, S.E. Washington, D.C. 20003

We've Grown On You.

Our selection of programs for your Atari 400/800 expands with a new program monthly.

To Order By Phone CALL: (202) 544-4259

Dealer Inquires *Only*, Call Computer Software Distributors: (800) 424-3708

Please send these Acorn Programs: ☐ **Please send your catalog**

______ ______ ______

______ ______ ______

Charge to: **TOTAL: $**______

☐ VISA ☐ MASTERCARD ☐ CHECK ENCLOSED

CARD NO. ______ EXP. DATE ______ **U-9**

Name______

Address______

City______ **State**______ **Zip**______ to in line 180) plays the song "Charade." At 280, the GOSUB 1190 waits for the player to press a key before clearing the screen and moving on to the next routine. The loop at 530 reads the 125 phrases into the array (M$), and the loops at 320 and 340 assure that each team has alternating turns and that five rounds make a game.

To shorten or lengthen the game, change the number of rounds (five) in 320. The routine beginning at 950 first indicates whose turn it is, then it generates a random number (X) which, when used as a subscript to M$, will select which phrase will be played next. GOSUB 1380 checks to see if the phrase selected has been used so as to avoid duplication of phrases during the game. In other words, once it is "drawn out of the hat," it is discarded and can't be used again. (It is unlikely that you will play long enough to use all the phrases, but after about 100 have been used, you will naturally notice that it takes a bit longer for the computer to select an unused phrase.) At 1010 the program determines and prints the category of the phrase (there are five groups of 25 phrases); the phrase itself is printed at 1140. Again, the routine at 1190 is used to wait for a signal from the player to clear the screen and start the clock.

The routine at 1280 is the clock, which counts down the time (T) and thus the score. The first CALL SOUND statement in the loop (line 1300) plays an inaudible tone (40,000 hz) at -30db for 750 milliseconds (3/4 of a second), then line 1310 gives the clock a "tick" by sounding a short (20 millisecond) 220 hz tone at -10db.

CALL SOUND is used as a timing device because it can be more accurately adjusted than delay loops; however, a For-Next loop of, say 1 to 250 might be used instead at line 1300. If you hold down a key when your phrase has been guessed, the clock will stop, thanks to lines 1320 and 1330, and the last number displayed (T) becomes your score for that round. Each loop takes a total of one second. To give players more or less than the two minutes allowed here, change the number (120) in line 1280.

Lines 1350 and 1360 provide a loud "time's up" tone and reprint the phrase. After five rounds, a C-major three-note fanfare (at lines 1240-1260) announces the end of that game. The score of the winning team is displayed. If you have played 12 games (and by that time it should be well past your bedtime), lines 490 and 510 end the game before you run out of data.

If You Have A Talkative Micro

The whole idea of charades, of course, is to convey an idea without speaking. But if you have the Texas

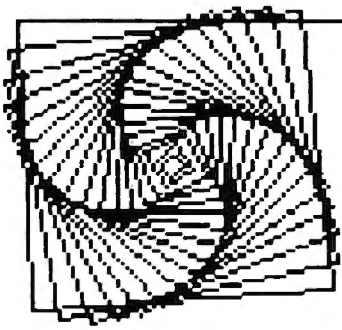
Instruments Speech Synthesizer unit and either the Speech Editor module or the TI Extended BASIC module, then it is okay to let your computer do the talking. If you wish, try adding these lines to the program:

```
185 CALL SAY("DO YOU WANT INSTRUCTIONS")
275 CALL SAY(P$)
335 CALL SAY("THIS IS.ROUND NUMBER")
336 CALL SAY(STR$(ROUND))
425 CALL SAY("NUMBER 2 #YOU WIN#")
465 CALL SAY("NUMBER 1 #YOU WIN#")
475 CALL SAY("DO YOU WANT TO PLAY AGAIN")
515 CALL SAY("GAMES OVER. GOODBYE")
965 CALL SAY("NUMBER")
966 CALL SAY(STR$(TEAM))
967 CALL SAY("IT IS YOUR TURN")
968 CALL SAY(P$)
969 CALL SAY("TO SEE YOUR WORDS")
1145 CALL SAY(P$)
1146 CALL SAY("TO START")
1165 CALL SAY(P$)
1166 CALL SAY("TO STOP. GO NOW")
1355 CALL SAY("STOP YOUR TIME IS UP")
```

```
100 REM     *CHARADES*     TI BASIC VERSION
   3/20/82
110 DIM M$(125),Z(125)
120 RANDOMIZE
130 P$="        PRESS ANY KEY"
140 Q=1
150 CALL CLEAR
160 PRINT TAB(8);"* CHARADES *"
170 PRINT ::::TAB(6);"COPYRIGHT(C)1982":TA
   B(7);"BY STEVE DAVEIS"::::::
180 GOSUB 1460
190 INPUT "WANT INSTRUCTIONS? (Y/N)":Y$
200 IF Y$<>"Y" THEN 290
210 PRINT :"DIVIDE GROUP INTO 2 TEAMS.":"YO
   U WILL PLAY 5 ROUNDS EACH.":"WHEN
   INSTRUCTED,1 PLAYER"
220 PRINT "FROM TEAM# DISPLAYED SHALL":"PRE
   SS A KEY TO REVEAL HIS":"PHRASE. HE
   SHOULD STUDY IT"
230 PRINT "BEFORE PRESSING A KEY TO":"START
    CLOCK. HE HAS 120 SEC.":"TO
   PANTOMIME THE CATEGORY"
240 PRINT "& PHRASE TO HIS TEAM.":"HE MAY N
   OT TALK OR WRITE.":"WHEN THE PHRASE
   IS GUESSED,"
250 PRINT "HOLD DOWN A KEY UNTIL CLOCK":"ST
   OPS. A TONE WILL SOUND":"WHEN TIME
   IS UP."
260 PRINT "THE LESS TIME YOU USE,":"THE HIG
   HER YOUR SCORE."
270 PRINT "CATEGORIES INCLUDE MOVIES,":"SON
   GS,BOOKS,PEOPLE,AND":"QUOTES & CLI
   CHES."
280 GOSUB 1190
290 GOSUB 530
300 SCOR(1)=0
310 SCOR(2)=0
320 FOR ROUND=1 TO 5
330 PRINT "ROUND #";ROUND:::
340 FOR TEAM=1 TO 2
350 GOSUB 950
360 SCOR(TEAM)=SCOR(TEAM)+T
370 NEXT TEAM
380 NEXT ROUND
390 GOSUB 1240
400 IF SCOR(1)>SCOR(2)THEN 460
410 IF SCOR(1)=SCOR(2)THEN 440
420 PRINT ::"CONGRATULATIONS ,TEAM #2!"::"Y
   OU WIN WITH A SCORE OF ":SCOR(2)
430 GOTO 470
440 PRINT ::"IT'S A TIE! THAT DOESN'T    HA
   PPEN OFTEN!"
450 GOTO 470
460 PRINT ::"CONGRATULATIONS, TEAM #1!"::"Y
   OU WIN WITH A SCORE OF":SCOR(1)
470 PRINT ::"WANT TO PLAY AGAIN? (Y/N)"
480 INPUT Y$
490 IF Q>=120 THEN 510
500 IF Y$="Y" THEN 300
510 PRINT "GAME OVER. OUT OF DATA":"TYPE RU
   N TO START AGAIN"
520 END
530 PRINT "INITIALIZING DATA,STAND BY"
540 FOR I=1 TO 125
550 READ M$(I)
560 NEXT I
570 CALL CLEAR
580 RETURN
590 REM *MOVIES*
600 DATA A MAN AND A WOMAN,MAN WITH THE GOL
   DEN ARM,SOME LIKE IT HOT,MARY POPPINS
610 DATA WHITE CHRISTMAS, MUTINY ON THE BOU
   NTY, ON THE WATERFRONT, YOUNG
   FRANKENSTEIN
620 DATA AGONY AND THE ECSTASY, THE WIZARD
   OF OZ, YOU ONLY LIVE TWICE, THE
   LITTLE FOXES
630 DATA DIAL M FOR MURDER, NORTH BY NORTHW
   EST, PSYCHO, LADY SINGS THE BLUES
640 DATA MEET ME IN ST.LOUIS, THE GREAT ZIE
   GFELD,LAURA,THE EMPIRE STRIKES BACK
650 DATA WHERE THE BOYS ARE, DOCTOR ZHIVAGO
   , DOCTOR STRANGELOVE,2001 A SPACE
   ODYSSEY, THE TURNING POINT
660 REM *BOOKS*
670 DATA VALLEY OF THE DOLLS, THE CARPETBAG
   GERS, GONE WITH THE WIND, EVERYTHING
   YOU WANTED TO KNOW ABOUT SEX
680 DATA CATCHER IN THE RYE, THE BIBLE, MAG
   NIFICENT OBSESSION, OLIVER TWIST
690 DATA WOMEN IN LOVE, JANE EYRE, REBECCA,
    ALICE IN WONDERLAND
700 DATA THE HOBBIT, FUTURE SHOCK, GOODBYE
   MR. CHIPS, MOBY DICK
710 DATA HUCKLEBERRY FINN, WAR AND PEACE, L
   ITTLE WOMEN, GULLIVER'S TRAVELS
720 DATA BRAVE NEW WORLD, THE SCARLET LETTE
   R, TALE OF TWO CITIES, GIANT, LOLITA
730 REM *PEOPLE*
740 DATA MARILYN MONROE, MARIE ANTOINETTE,
   GROUCHO MARX, JOHN KENNEDY
750 DATA MARTIN LUTHER KING, SOPHIA LOREN,
   WALTER CRONKITE, SEAN CONNERY
760 DATA ELEANOR ROOSEVELT, JUDY GARLAND, E
   DGAR HOOVER, COLUMBUS
770 DATA GREER GARSON, RONALD REAGAN, LADY
   BIRD JOHNSON, NELSON EDDY
780 DATA JOHNNY CARSON, GEORGE WALLACE, CYD
    CHARISSE, GRETA GARBO
790 DATA DOLLY PARTON, JOAN CRAWFORD, BETTE
   DAVIS, PAT NIXON, GEORGE GERSHWIN
800 REM *QUOTES&CLICHES*
```

```
810 DATA A STITCH IN TIME SAVES NINE, DONT ~
    LOOK A GIFT HORSE IN THE MOUTH,CLEAN
    AS A WHISTLE, NEVER SAY DIE
820 DATA REMEMBER THE ALAMO, IGNORANCE IS B
    LISS, HASTE MAKES WASTE, CONTENTED
    AS A COW
830 DATA ALL THAT GLITTERS IS NOT GOLD, PUR
    R LIKE A KITTEN, I SHALL RETURN,
831 SHARP AS A TACK
840 DATA TO BE OR NOT TO BE, I'LL THINK ABO
    UT THAT TOMORROW, I WANT TO BE ALONE,
    THE BUCK STOPS HERE
850 DATA WE HAVE NOTHING TO FEAR BUT FEAR I
    TSELF, THAT'S ALL FOLKS, WHAT'S UP
    DOC, THERE'S NO PLACE LIKE HOME
860 DATA DONT COUNT YOU CHICKENS BEFORE THE
    Y HATCH, PARTING IS SUCH SWEET
    SORROW, HOLD YOUR HORSES
870 DATA IT'S ALWAYS DARKEST BEFORE THE DAW
    N, HINDSIGHT IS 20/20 VISION
880 REM *SONGS*
890 DATA SANTA CLAUS IS COMING TO TOWN, STA
    RDUST, MY FUNNY VALENTINE, FEELINGS
900 DATA MIDNIGHT BLUE, PEOPLE, CAMP TOWN R
    ACES, SOME ENCHANTED EVENING
910 DATA DO RE MI, I WANNA HOLD YOUR HAND, ~
    YESTERDAY, DOWNTOWN
920 DATA HOUSE OF THE RISING SUN, MY COUNTR
    Y TIS OF THEE, THE LADY IS A TRAMP,
    THE MAN I LOVE
930 DATA ST.LOUIS BLUES, AMERICAN PIE, STOR
    MY WEATHER, OVER THE RAINBOW
940 DATA YOU'VE GOT A FRIEND, MOON RIVER, I
     GOT PLENTY OF NOTHIN, TRY TO
    REMEMBER, YOU'LL NEVER KNOW
950 REM
960 PRINT "TEAM #";TEAM;" -IT'S YOUR TURN":
    :
970 GOSUB 1190
980 X=INT(RND*125)+1
990 GOSUB 1380
1000 REM
1010 IF X<=25 THEN 1070
1020 IF (X>=26)*(X<=50)THEN 1090
1030 IF (X>=51)*(X<=100)THEN 1130
1050 PRINT ::"(SONG)"::::
1060 GOTO 1140
1070 PRINT ::"(MOVIE)"::::
1080 GOTO 1140
1090 PRINT ::"(BOOK)"::::
1100 GOTO 1140
1110 PRINT ::"(PERSON)"::::
1120 GOTO 1140
1130 PRINT ::"(QUOTE&CLICHE)"::::
1140 PRINT M$(X):::::
1150 GOSUB 1190
1160 PRINT "(HOLD DOWN A KEY TO STOP)"
1170 GOSUB 1280
1180 RETURN
1190 PRINT :P$::
1200 CALL KEY(0,KEY,STATUS)
1210 IF STATUS=0 THEN 1200
1220 CALL CLELAR
1230 RETURN
1240 CALL SOUND(300,523,2,392,3,330,3)
1250 CALL SOUND(200,494,2,294,3,247,3)
1260 CALL SOUND(400,523,2,392,3,330,3)
1270 RETURN
1280 FOR T=120 TO 1 STEP -1
1290 PRINT T
1300 CALL SOUND(750,40000,30)
1310 CALL SOUND(20,220,10)
1320 CALL KEY(0,KEY,STATUS)
1330 IF STATUS=0 THEN 1340 ELE 1350
1340 NEXT T
1350 CALL SOUND(1100,220,0)
1360 PRINT:M$(X):::TAB(10);"* * *":::: 1370
     RETURN
1380 REM TEST FOR DUP
1390 FOR Y=1 TO Q
1400 IF X=Z(Y)THEN 980
1410 NEXT Y
1420 Z(Q)=X
1430 Q=Q+1
1440 RETURN
1450 REM TUNE
1460 DUR=250
1470 CALL SOUND(DUR,262,1)
1480 CALL SOUND(DUR,277,1)
1490 CALL SOUND(DUR,262,1)
1500 CALL SOUND(DUR*2,392,1)
1510 CALL SOUND(DUR,349,1)
1520 CALL SOUND(DUR*3,262,1)
1530 CALL SOUND(100,40000,30)
1540 CALL SOUND(DUR,262,1)
1550 CALL SOUND(DUR,277,1)
1560 CALL SOUND(DUR,262,1)
1570 CALL SOUND(DUR*2,233,1)
1580 CALL SOUNDD(DUR,208,1)
1590 CALL SOUND(DUR,262,1)
1600 CALL SOUND(DUR*3,196,1)
1610 RETURN
```

©



A Monthly Column

Friends Of The Turtle

David D. Thornburg
Associate Editor

Battle Of The UFL's

The development of User Friendly Languages (UFL's) is proceeding so quickly that any report is likely to be outdated by the time it appears. Nonetheless, there is enough interest in the UFL's for the Atari, TI, and Apple computers to warrant an overview of the best offerings for these machines. The user friendly languages of principal interest seem to be PILOT and LOGO. It so happens that Atari PILOT (and Apple SuperPILOT) incorporate turtle graphics. While turtle graphics (common to all LOGO's) is not essential for a language to be user-friendly, it helps.

Rather than detail all UFL's for each computer, I will restrict the analysis to Atari PILOT, TI LOGO and Apple LOGO. The differences between Apple LOGO and the Apple versions of LOGO produced by Terrapin and Krell are deserving of separate comment later. (I have just received Krell LOGO and will need to use it some more before writing about it.)

What makes the following comparison interesting is the tremendous difference in price and features of the three chosen language systems. The table summarizes all three configurations. I have listed the bare minimum configuration needed to make the language work. If you want to save your programs, the cost of a recorder must be added to the Atari and TI systems. Since all three systems require a separate display, I have left that item out of the cost analysis.

The entries in this table reflect questions readers have been sending to Friends of the Turtle.

The Atari PILOT system is the least expensive. This results from the low cost of the computer and from the fact that Atari PILOT can be used with a minimum amount of RAM. The increased memory requirement of TI LOGO results in a profoundly increased cost for that system – a cost difference we would not have if we were comparing BASIC's. Since Apple LOGO requires both 64K of RAM and a disk drive, it is the most expensive of the systems. However, Apple LOGO is by far the most powerful of the languages under consideration.

The turtle graphics implementations are ex-

Table. System Comparison

Feature	Atari PILOT	TI LOGO	Apple LOGO
Minimum System	Atari 400 Pilot cartridge	TI 99/4A 32K memory exp. LOGO cartridge	Apple II or II+ language card floppy disk LOGO disk
List Price	$429	$980	$2625
Visible Turtle	No	Yes	Yes
Turtle Graphics Resolution	160 X 80	256 X 192 (may "run out of ink")	280 X 240 (vertical scale is changeable)
Number of Simultaneous Colors	4	16	6
Total Color Range	16 hues X 8 luminances	16	6
Character Font Editor	No	Yes	No
Multiple Dynamic Turtles	No	32	No
Real Number Arithmetic	No	No	Yes
Unlimited Tail-end Recursion	Yes	No	Yes
Recursion Depth Before Crash	8	end of memory	end of memory
Access to Joysticks, etc.	Yes	No	Yes
Full Stroke Keyboard	No	Yes	Yes
TV Sound Generator	Yes	Yes (in LOGO II)	No
Direct Memory Access	Yes	No	Yes

cellent in all three systems. Atari PILOT is the only one that does not have a visible turtle, but this can be remedied somewhat with the Visiturt program I published a few months ago (**COMPUTE!**, April 1982, #23). The Atari system has the lowest resolution, but has the greatest color accuracy and range of the three languages. A major annoyance with the TI system is the "out of ink" error that arises when trying to create complex pictures. Since TI creates high resolution graphics by dynamic character definition (a topic for a later column), it is not as versatile as a true memory mapped display. Multiple velocity turtles (turtles that have speeds as well as positions and orientations) are only available on TI LOGO. Up to 32 such animated characters can be created with any of 26 shapes formed in a 16 X 16 dot matrix. While the Atari hardware allows for such animated characters (called players), PILOT users must gain access to these through machine language instructions. Of the three systems, only TI allows the user to interactively modify or define the shapes of characters and velocity turtles.

If you are content with integer arithmetic, any of the systems will do. If you must have access to decimal fractions, only Apple LOGO will meet your needs. Interestingly enough, the restriction to integer arithmetic can result in minor graphics problems (drawing a regular seven-sided polygon, for example) in Atari PILOT and TI LOGO, although these problems can be easily overcome by careful programming.

Recursion is of two types. A simple jump to the beginning of a procedure is called tail-end recursion. Recursion involving the use of a procedure that ultimately returns to the calling procedure is more difficult since the computer must keep track of the sequence and names of all calling procedures. As a result, recursion can use up all free memory just by keeping track of this information. Atari PILOT allows unlimited tail-end recursion (as does Apple LOGO), but allows only eight nested procedure calls.

TI LOGO differs from both Atari PILOT and Apple LOGO in that the user is not provided with access to joysticks nor to the direct reading and alteration of memory. Both Atari PILOT and Apple LOGO have the equivalent of BASIC PEEK and POKE commands to allow the examination and alteration of the contents of arbitrary memory locations.

The keyboard quality is highest for the Apple II, although TI's decision to use a conventional keyboard makes that machine easy to use as well. My experience is that the Atari 400 membrane keyboard is acceptable to children, but is annoying to adults accustomed to typewriters. Since the Atari 800 has a fine full-stroke keyboard, this option is available to those willing to pay the higher price.

In summary, each system has strong features and drawbacks. You are certain to like some aspects of each system. For the price, the Atari PILOT system is beyond comparison. On the other hand, Apple LOGO is a powerhouse of a language, and its features are well worth its price. The ease with which animated sprites can be created and used in TI LOGO makes this system a natural choice for anyone interested in animation.

All three manufacturers are in this business for the long haul, so your selection should be based purely on needs and budget.

Apple LOGO And The Silentype Printer

Those of you who use the Silentype printer with your Apple computer have probably wondered how to get copies of the displays of turtle graphics created by LOGO procedures. The easiest way I have found is to initialize the printer before loading LOGO. When you initialize a file diskette, it contains a program named HELLO. Normally (for LOGO) there will be no statements in this program. However, if you were to boot this disk first rather than start with the LOGO disk, the HELLO program would be automatically run. Since your Apple already has one dialect of BASIC in ROM (either integer or Applesoft), then you could use a BASIC HELLO program to initialize the Silentype printer.

The Silentype manual shows the numerous ways in which the printer's graphic features can be set up. The default mode lets the printer print bidirectionally. One set of dots is printed as the head moves from left to right and the second set is drawn as it moves from right to left. While this significantly improves the printing speed, the slack in the Silentype mechanism causes this mode to produce unacceptable vertical misalignment when printing high resolution graphics. The unidirectional printing mode does not have this problem.

Second, the Silentype normally prints images just as they appear on the screen. If you have a few white lines on a black background, that is how the printer will print the picture. Normally one expects the reverse of this for line drawings – the background should be white and the lines should be dark. As a result, the printer needs to have its color fields reversed.

Both of these changes are made in the BASIC program shown below. This program assumes that the Silentype printer interface is located in slot #1 and that the disk drive is located in slot #6.

```
10 D$="":REM D$ CONTAINS CTRL-D

20 PRINT D$;"PR #1"
```

```
30 PRINT
40 POKE -12529,255
50 POKE -12524,0
60 PRINT D$;"PR #0"
70 PRINT "GRAPHICS PRINTER INITIALIZED"
80 PRINT "INSERT LOGO DISK AND PRESS RETURN"
90 INPUT A$
100 PRINT D$;"PR #6"
110 END
```

Once this program has been saved in the HELLO file, your printer will be automatically initialized. To set up the printer, you must first start the computer with the file diskette that has this HELLO program. Once the display instructs you to insert the LOGO disk, you should do that and press RETURN. Now you will have LOGO in the computer and also have a properly initialized printer.

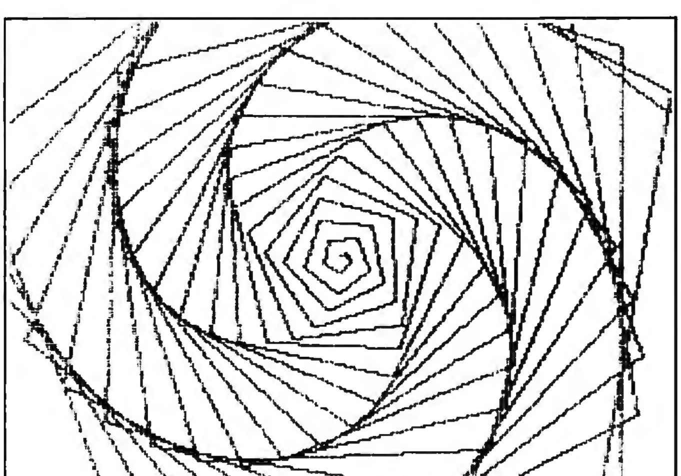
To print a high resolution screen image from LOGO you can use the procedure:

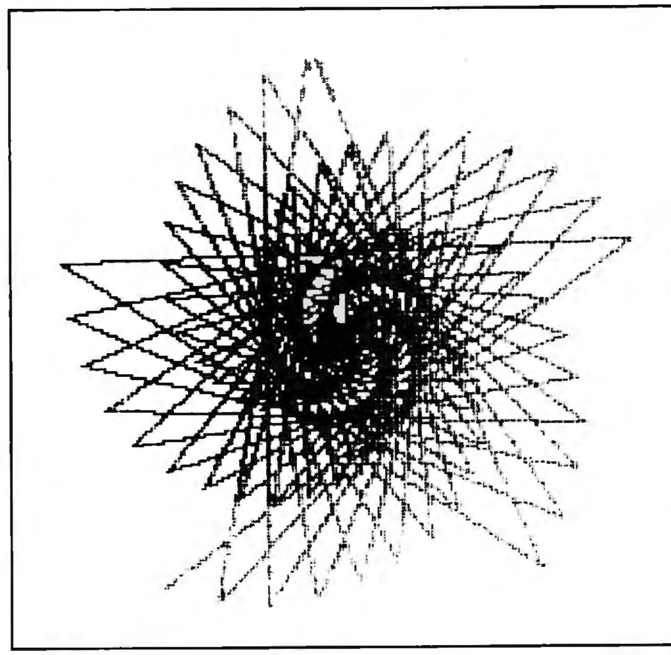
```
TO PICT
.PRINTER 1
PRINT CHAR 17
.PRINTER 0
END
```

From then on, any time you enter PICT the current graphics screen will be copied onto the printer.

If you try printing an image of a square or a circle, you may notice that the image is squashed vertically. This results from a difference in the aspect ratio of the printer and your TV display. To print pictures with a perfect aspect ratio, you must enter

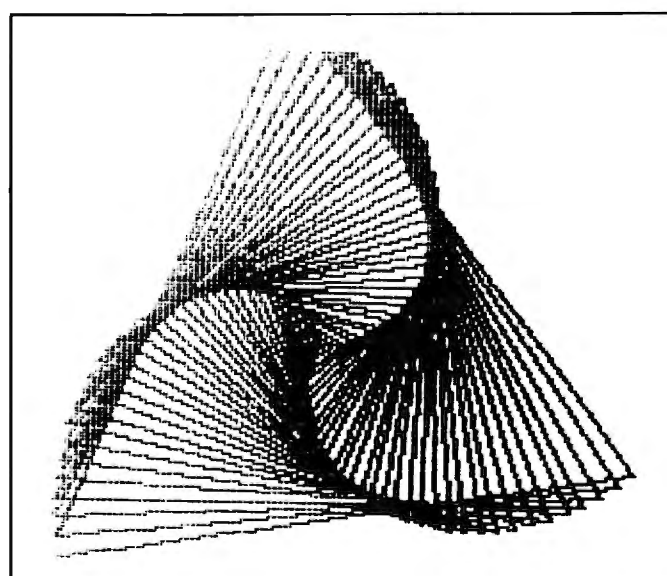
```
SETSCRUNCH 1
```





before drawing the figure you want to print. Once the aspect ratio has been changed to this value (from its default value of 0.8), all your pictures will come out perfectly. The accompanying figures show some of the results.

The Silentype printer is an excellent tool for capturing your LOGO graphic images. It is time you put it to work!



Teachers sometimes need to adjust a mark distribution. Reasons can include an exam that was too difficult or perhaps two sections of a course taught by different teachers who have very different standards. This useful set of programs calculates accurate adjustments. It is for Atari and Microsoft (Apple, PET, OSI, etc.) BASICs.

Student Mark Adjustment

R D Wink
Peterborough, Ontario

Adjusting marks by simply adding a constant can cause problems at the extremes of the distribution (a mark of 106% could become possible, for example). This program applies a quadratic regression to the three points (0,0); (X1,Y1) and (M,M) where M is the maximum mark, X1 is the actual mean or median mark, and Y1 is the required mean or median mark.

The output in Programs 1 and 3 is written in single-column form, but it can easily be arranged into several columns depending upon the dimensions of your computer screen.

Since the quadratic curve has a turning point which must be avoided in this application, the change in median must be relatively small to avoid ridiculous results. For example, if the original median turns out to be 50%, the new median must not be below 25% or above 75%. In practice, this is not a serious limitation. You will know if you have exceeded the range of the program because the new marks will rise above the value of M at the high end of the distribution.

Program 1. Atari Version

```
100 PRINT "{CLEAR}THIS PROGRAM WILL COMP
    UTE A MARK":PRINT " CONVERSION TABLE
    "
110 PRINT "{2 DOWN}PLEASE INPUT THE MAXI
    MUM POSSIBLE MARK";:INPUT M
120 PRINT "{DOWN}PLEASE INPUT THE MEDIAN
     (OR AVERAGE)":PRINT "OF YOUR SET OF
     MARKS";:INPUT X2
130 PRINT "{DOWN}PLEASE INPUT THE MEDIAN
     THAT YOU":PRINT "WOULD LIKE THE CLA
    SS TO HAVE";:INPUT Y2
135 DIM A$(1)
140 GOSUB 370
150 X=0
160 PRINT "{CLEAR}";:R=0:S=5:C=0
170 REM INITIALIZE TAB VARIABLES
180 GOSUB 320
190 REM PRINT HEADINGS
200 POSITION 0,2
210 FOR K=1 TO 20
220 GOSUB 290
230 POKE 85,R+2:PRINT X;:POKE 85,S+2:PRI
    NT P
240 X=X+1
250 NEXT K
260 C=C+1:R=R+10:S=S+10
270 IF C=3 THEN 350
280 GOTO 200
290 P=INT(Y2*X*(X-M)/(X2*(X2-M))+X*(X-X2
    )/(M-X2)+0.5)
300 IF X>M THEN POSITION 2,22:GOSUB 370:
    END
310 RETURN
320 PRINT "OLD  NEW  OLD  NEW  OLD  NEW"
330 PRINT "MARK MARK MARK MARK MARK MARK
    "
340 RETURN
350 GOSUB 370
360 GOTO 160
370 PRINT "PRESS RETURN TO CONTINUE";:IN
    PUT A$
380 RETURN
```

Program 2. Atari Version

```
5 REM MARK ADJUSTMENT PROGRAM
15 REM
20 REM INPUT SECTION 20-45
25 PRINT "MAXIMUM POSSIBLE MARK";:INPUT
   M
30 PRINT "ACTUAL MEDIAN";:INPUT X1
40 PRINT "DESIRED MEDIAN";:INPUT Y1
45 REM
50 REM COMPUTE AND PRINT CONVERSION TABL
   E
55 PRINT "ORIGINAL MARK{13 SPACES}FINAL
   MARK"
60 FOR X=0 TO M
65 Y=Y1*X*(X-M)/(X1*(X1-M))+X*(X-X1)/(M-
   X1)
70 POKE 85,9:PRINT X;:POKE 85,32:PRINT I
   NT(Y+0.5)
75 NEXT X
80 END
```

Program 3. Microsoft Version

```
100 PRINT "{CLEAR}THIS PROGRAM WILL COMPUTE
    A MARK      CONVERSION TABLE"
110 INPUT "{02 DOWN}PLEASE INPUT THE MAXIMUM P
    OSSIBLE MARK";M
120 INPUT"{DOWN}PLEASE INPUT THE MEDIAN (OR AV
    ERAGE)    OF YOUR SET OF MARKS";X2
130 INPUT"{DOWN}PLEASE INPUT THE MEDIAN THAT Y
    OU WOULD   LIKE THE CLASS TO HAVE";Y2

140 GOSUB 370
150 X=0
160 PRINT "{CLEAR}":R=0:S=5:C=0
170 REM INITIALISE TAB VARIABLES
180 GOSUB 320
190 REM PRINT HEADINGS
200 PRINT "{HOME}{03 DOWN}"
210 FOR K = 1 TO 20
220 GOSUB 290
230 PRINT TAB(R);X;TAB(S);P
240 X=X+1
250 NEXT K
260 C=C+1:R=R+10:S=S+10
270 IF C=3 GOTO 350
280 GOTO 200
290 P=INT(Y2*X*(X-M)/(X2*(X2-M))+X*(X-X2)/(M-X
```

```
    2)+.5)
300 IF X>M THEN GOSUB 380:END
310 RETURN
320 PRINT "OLD  NEW  OLD  NEW  OLD  NEW"
330 PRINT "MARK MARK MARK MARK MARK MARK"
340 RETURN
350 GOSUB 370
360 GOTO 160
370 INPUT"{DOWN}PRESS RETURN TO CONTINUE   .{0
    3 LEFT}";A$
380 PRINT "{HOME}":FOR K= 1 TO 23:PRINT:NEXT:R
    ETURN
```

Program 4. Microsoft Version

```
20 REM INPUT SECTION 20-45
25 INPUT"MAXIMUM POSSIBLE MARK";M
30 INPUT"ACTUAL MEDIAN";X1
40 INPUT"DESIRED MEDIAN";Y1
45 REM
50 REM COMPUTE AND PRINT CONVERSION TABLE
55 PRINT"ORIGINAL MARK                FINAL MA
   RK"
60 FOR X=0 TO M
65 Y=Y1*X*(X-M)/(X1*(X1-M))+X*(X-X1)/(M-X1)
70 PRINT TAB(9);X;TAB(32);INT(Y+.5)
75 NEXT X
```

©

DYNACOMP

Quality software for:*

APPLE II Plus
OSBORNE-1
NORTH STAR***
ATARI
PET/CBM
ALTAIR****
NEC PC-8000
TRS-80 (Level II)**
SUPERBRAIN*****
CP/M Disks/Diskettes
(See Availability box MBASIC/CBASIC)

CARD GAMES

NEW

BRIDGE MASTER (Available for all computers) Price: $21.95 Diskette
If you liked DYNACOMP'S BRIDGE 2.0, you will absolutely love BRIDGE MASTER. BRIDGE MASTER is a comprehensive bridge program designed to provide hours of challenging competition. Bidding features include the Blackwood convention, Stayman convention, pre-emptive openings, and recognition of demand bids and jump-shift responses. After playing a specific hand, you may replay the same hand, with the option of switching cards with your computer opponents. This feature allows you to compare your bidding and playing skills to BRIDGE MASTER. Bonuses for game contracts and slams are awarded as in duplicate bridge. Doubled contracts are scored based upon a computer assigned vulnerability. A score card is displayed at the conclusion of each hand. The score card displays a summary of total hands played, total points scored, number of contracts made and set, and % bids made. BRIDGE MASTER is clearly the best computer bridge program available.
DYNACOMP'S previous BRIDGE 2.0 customers may upgrade to BRIDGEMASTER for a nominal charge of $5.00 plus postage and handling (see ordering information box). Original cassette/diskette must be returned.

BACCARAT (Atari only) Price: $18.95 Cassette/$22.95 Diskette
This is the European card game which is the favorite of the Monte Carlo jet set. Imagine yourself at the gaming table with 007 to your left and Goldfinger to your right. Learn and play BACCARAT at your leisure on the Atari. Contains full high resolution color graphics and matching sound. Runs in 16K. Requires one joystick.

GIN RUMMY (Apple diskette only) Price: $22.95 Diskette
This is the best micro computer implementation of GIN RUMMY existing. The computer plays exceptionally well, and the HIRES graphics are superb. What else can be said?

POKER PARTY (Available for all computers) Price: $19.95 Cassette/$23.95 Diskette
POKER PARTY is a draw poker simulation based on the book, POKER, by Oswald Jacoby. This is the most comprehensive version available for microcomputers. The party consists of yourself and six other (computer) players. Each of these players (you will get to know them) has a different personality in the form of a varying propensity to bluff or fold under pressure. Practice with POKER PARTY before going to that expensive game tonight! Apple cassette and diskette versions require a 32K (or larger) Apple II.

GO FISH (Available for all computers) Price: $14.95 Cassette $18.95 Diskette
GO FISH is a classic children's card game. The opponent is a friendly computer with user inputs that are simple enough for small children to easily master. The Apple and Atari versions employ high resolution graphics for the display of hands. A must for children! Runs in 16K.

NEW

BLACKJACK COACH (32K TRS-80 only) Price: $29.95 Cassette $33.95 Diskette
BLACKJACK COACH teaches and evaluates professional playing methods. This program will coach you using the Basic and the Complete Card Counting Methods. The BLACKJACK COACH can be used in automatic, unattended play to test the playing and betting strategies you select. Extensive summary reports pinpoint the strengths and weaknesses of various methods of play. All the standard player choices are included: Insurance, splitting pairs, double downs and surrender (optional). A line printer may be used to collect data. If you risk money at the tables, increase your skills with the BLACKJACK COACH!

THOUGHT PROVOKERS

MANAGEMENT SIMULATOR (Available for all computers) Price: $25.95 Cassette $29.95 Diskette
This program is both an excellent teaching tool as well as a stimulating intellectual game. Based upon similar games played at graduate business schools, each player or team controls a company which manufactures three products. Each player attempts to outperform his competitors by setting selling prices, production volumes, marketing and design expenditures etc. The most successful firm is the one with the highest stock price when the simulation ends.

FLIGHT SIMULATOR (Available for all computers) Price: $19.95 Cassette $23.95 Diskette
A realistic and extensive mathematical simulation of take-off, flight and landing. The program utilizes aerodynamic equations and the characteristics of a real airfoil. You can practice instrument approaches and navigation using radials and compass headings. The more advanced flyer can also perform loops, half-rolls and similar aerobatic maneuvers. Although this program does not employ graphics, it is exciting and very addictive. See the software review in COMPUTRONICS. Runs in 16K Atari.

VALDEZ (Available for all computers) Price: $17.95 Cassette $21.95 Diskette
VALDEZ is a computer simulation of supertanker navigation in the Prince William Sound, Valdez Narrows region of Alaska. Included in this simulation is a realistic and extensive 256 x 256 element map, portions of which may be viewed using the ship's alphanumeric radar display. The motion of the ship itself is accurately modelled mathematically. The simulation also contains a model for the tidal patterns in the region, as well as other traffic (outgoing tankers and drifting icebergs). Chart your course from the Gulf of Alaska to Valdez Harbor! See the software reviews in 80 Software Critique, Personal Computing and Creative Computing.

BACKGAMMON 2.0 (Available for all computers) Price: $19.95 Cassette $23.95 Diskette
This program tests your backgammon skills and will also improve your game. A human can compete against a computer or against another human. The computer can even play against itself! Either the human or the computer can double or generate dice rolls. Board positions can be created or saved for replay. BACKGAMMON 2.0 plays in accordance with the official rules of backgammon and is sure to provide many fascinating sessions of backgammon play.

NEW

FROG MASTER (Atari only) Price: $17.95 Cassette $21.95 Diskette
The Atari APEX first prize winner. FROG MASTER contains exciting arcade features in addition to being a highly educational program. It is a fast-moving high-concentration game for 1-4 players. You score by making touchdowns on the opponents' goal line – if his goalie doesn't get there first. But your players (tadpoles and frogs) must be trained. This is accomplished by giving them a reward at just the right moment when they do something right. This takes precise timing and judgement. Your critters must penetrate barriers and avoid evil line backs if they are to score. Many will fall by the wayside, but some will get through. As they learn you can look inside their heads to see how they think. As you reward them, they reward you (the "thought processes" simulated demonstrate the basic type of animal learning - operant conditioning - widely studied in high school and college courses). As you teach them they teach you how learning takes place! Great graphics! Runs in 16K. Requires two joysticks.

FOREST FIRE! (Atari only) Price: $14.95 Cassette/$18.95 Diskette
Using excellent graphics and sound effects, this simulation puts you in the middle of a forest fire. Your job is to direct operations to put out the fire while compensating for changes in wind, weather and terrain. Not protecting valuable structures can result in startling penalities. Life-like variables are provided to make FOREST FIRE! very suspenseful and challenging. No two games have the same setting and there are 3 levels of difficulty.

CRANSTON MANOR ADVENTURE (North Star, SuperBrain and CP/M only) Price: $19.95 Diskette
At last! A comprehensive Adventure game for North Star and CP/M systems. CRANSTON MANOR ADVENTURE takes you into mysterious CRANSTON MANOR where you attempt to gather fabulous treasures. Lurking in the manor are wild animals and robots who will not give up the treasures without a fight. The number of rooms is greater and the associated descriptions are much more elaborate than the current popular series of Adventure programs, making this game the top in its class. Play can be stopped at any time and the status stored on diskette.

SPACE EVACUATION! (Available for all computers) Price: $15.95 Cassette $19.95 Diskette
Can you colonize the galaxy and evacuate the Earth before the sun explodes? Your computer becomes the ship's computer as you explore the universe to relocate millions of people. This simulation is particularly interesting as it combines many of the exciting elements of classic space games with the mystery challenge of ADVENTURE.

MONARCH (Atari only) Price: $14.95 Cassette $18.95 Diskette
MONARCH is a fascinating economic simulation requiring you to survive an 8-year term as your nation's leader. You determine the amount of acreage devoted to industrial and agricultural use, how much food to distribute to the populace and how much should be spent on pollution control. You will find that all decisions involve a compromise and that it is not easy to make everyone happy. Runs in 16K Atari.

NEW

RUBIK'S CUBE SOLVER (Available for all computers) Price: $14.95 Cassette $18.95 Diskette/$21.45 Disk
Solving the Rubik's cube puzzle is an exercise in algorithmic logic, and is a "natural" for computer calculation. The RUBIK'S CUBE SOLVER permits you to input the starting state of the 24 facing elements of the cube. It then solves the problem one step at a time, with each step shown as a unfolded view of the cube. Can you solve the cube in fewer steps. In any case, it sure beats disassembling the cube or peeling off and replacing the colors! Requires 16K.

AVAILABILITY

DYNACOMP software is supplied with complete documentation containing clear explanations and examples. Unless otherwise specified, all programs will run within 16K program memory space (ATARI requires 24K). Except where noted programs are available on ATARI, PET, TRS-80 (Level II), NEC and Apple (Applesoft) cassette and diskette as well as North Star single density (double density compatible) diskette. Additionally, most programs can be obtained on standard (IBM 3740 single density double density compatible format) 8" CP/M floppy disks for systems running under MBASIC or CBASIC (for example Altos, Xerox 820, Heath Zenith and many others). 5¼" CP/M diskettes are available for the North Star, SuperBrain and Osborne computer systems.

*ATARI, PET, CBM, NORTH STAR, CP/M, IBM, OSBORNE, SUPERBRAIN, NEC PC-8000 and XEROX are registered trademarks and/or trademarks.
**Except where noted, all TRS-80 Model I software is available (on cassette only) for the TRS-80 Model III. Exceptions: VALDEZ, CRIBBAGE, GRAFIX, CHESSMASTER. TRS-80 diskettes are not supplied with either DOS or BASIC.
***For most North Star disk-based systems
****For Altair systems having Microsoft BASIC
*****For SUPERBRAIN systems running under MBASIC or CBASIC (state which)

DYNACOMP OFFERS THE FOLLOWING

- Widest variety
- Guaranteed quality
- Fastest delivery
- Friendly service
- Free catalog*
- Toll free order phone

*$2.00 U.S. funds for foreign mailing

AND MORE...

STARBASE 3.2 (Available for all computers) Price: $13.95 Cassette/$17.95 Diskette
This is the classic space simulation, but with several new features. For example, the Croylins now shoot at the Invincible without warning while also attacking starbases in other quadrants. The Croylins also attack with both light and heavy cruisers and move when shot at! The situation is hectic when the Invincible is besieged by three heavy cruisers and a starbase S.O.S. is received! The Croylins get even! See the software reviews in A.N.A.L.O.G., 80 Software Critique and Game Merchandising.

LIL' MEN FROM MARS (Atari only) Price: $19.95 Cassette/$23.95 Diskette
Defend yourself! The little men from Mars are out to get you if you don't get them first. This is a hilarious high resolution animated graphics (arcade) game which exercises much of the Atari's power. Requires one joystick.

NEW

ALVIN (Atari only) Price: $17.95 Cassette $21.95 Diskette
ALVIN is a great arcade game. You are commanding a highly maneuverable ship seeking to destroy several enemy cities. You are attempting to bomb these cities while at the same time trying to avoid their defensive fire ("MISSILE COMMAND" in reverse?). Also, your radar has been damaged so that you can only see downwards. This would normally not be much of a problem except that you also have to contend with high-flying enemy aircraft. As long as you are above these aircraft you have the advantage and are safe. However, high level bombing takes considerable skill. Therefore to achieve your goal the best strategy is to swoop down for a bombing run while the enemy craft is out of range and quickly retreat to the skies. A fun game. Requires 16K.

ESCAPE FROM VOLANTIUM (Atari only) Price: $15.95 Cassette/$19.95 Diskette
Bring the action and excitement of an arcade into your home with ESCAPE FROM VOLANTIUM! To escape you must maneuver your space ship around obstacles and laser blast the guardian (without being eaten). If he is killed with a direct shot (not just a leg lopped off), a door opens to the outside. However, the door does not stay open indefinitely. If you fail to escape in time, the door closes and a new guardian appears. Sometimes you can smash through the door by repeatedly chipping away at it. Other times it is impervious. At the higher levels of play more obstacles and guardians appear, adding to the excitement. Uses high resolution graphics and sound. Runs in 16K.

ALPHA FIGHTER (Atari only) Price: $13.95 Cassette/$17.95 Diskette
Two excellent graphics and action programs in one! ALPHA FIGHTER requires you to destroy the alien starships passing through your sector of the galaxy. ALPHA BASE is in the path of an alien UFO invasion, let five UFO's get by and the game ends. Both games require the joystick and get progressively more difficult the higher you score! ALPHA FIGHTER will run on 16K systems.

THE RINGS OF THE EMPIRE (Atari only) Price: $14.95 Cassette $18.95 Diskette
The empire has developed a new battle station protected by rotating rings of energy. Each time you blast through the rings and destroy the station, the empire develops a new station with more protective rings. This exciting game runs on 16K systems, employs extensive graphics and sound and can be played by one or two players.

INTRUDER ALERT (Atari only) Price: $15.95 Cassette/$19.95 Diskette
This is a fast paced graphics game which places you in the middle of the "Dreadstar" having just stolen its plans. The droids have been alerted and are directed to destroy you at all costs. You must find and enter your ship to escape with the plans. Five levels of difficulty are provided. INTRUDER ALERT requires a joystick and will run on 16K systems.

MIDWAY (Atari 32K only) Price: $14.95 Cassette $18.95 Diskette
MIDWAY is an exciting extension of the game of Battleship. It mixes the challenges of strategy and chance. Your opponent can be another human or the computer. Color graphics and sound are both included. Runs in 16K.

NEW

GOLF PRO (Atari only) Price: $17.95 Cassette $21.95 Diskette
Both realism and beautiful graphics are joined together in GOLF PRO to produce the best golf simulation available. To really appreciate this game, you should have a color TV so that you can see the green of the fairway, the blue of the water hazards, and the white sand of the traps. You tee off with a wood, use your wedge in the sand trap, and putt on the green just as would be done on the course. Show off the Atari to your friends with GOLF PRO. Requires 16K and one joystick.

GAMES PACK I (Available for all computers) Price: $14.95 Cassette $18.95 Diskette
GAMES PACK I contains the classic computer games of BLACKJACK, LUNAR LANDER, CRAPS, HORSERACE, SWITCH and more. These games have been combined into one large program for ease in loading. They are individually accessed by a convenient menu. This collection is worth the price just for the DYNACOMP version of BLACKJACK.

GAMES PACK II (Available for all computers) Price: $14.95 Cassette $18.95 Diskette
GAMES PACK II includes the games CRAZY LIGHTS, JOTTO, ACEY-DUCEY, LIFE, WUMPUS and others. As with GAMES PACK I, all the games are loaded as one program and are called from a menu. You will particularly enjoy DYNACOMP's version of CRAZY LIGHTS.
Why pay $9.95 or more per program when you can buy a DYNACOMP collection for just $14.95?

MOON PROBE (Available for all computers) Price: $12.95 Cassette/$16.95 Diskette
This is an extremely challenging "lunar lander" program. The user must drop from orbit to land at a predetermined target on the moon's surface. You control the thrust and orientation of your craft plus direct the rate of descent and approach angle. Runs in 16K Atari.

SPACE TRAP (Atari only, 16K) Price: $14.95 Cassette $18.95 Diskette
This galactic "shoot 'em up" arcade game places you near a black hole. You control your spacecraft using the joystick and attempt to blast as many of the alien ships as possible before the black hole closes about you.

SUPER SUB CHASE (Atari only) Price: $19.95 Cassette $23.95 Diskette
SUPER SUB CHASE simulates a search and destroy mission. Set your course and keep an eye on the sonar readings as you hunt for the hidden submarine. Set the depth charge explosion depth and watch them sink towards the sub. This is an addictive game which takes advantage of the Atari's graphics and sound capabilities. One or two players. Joystick(s) required.

TWO PLAYER GAMES

NEW

TWO PLAYER GAMES (Available for all computers: 32K disk/diskette only)
DYNACOMP has acquired the distribution rights to the best eight of Kitron's war games. These two-player games were originally written for the North Star computer, but have since been converted to play on all of the computers currently supported by DYNACOMP. Because our licensing and development costs were so low, DYNACOMP offers these programs two to a disk(ette) for only $19.95 diskette, $23.95 disk. If you like war games, then this is a bargain you can not pass up.

Set #1 PANZER and BLITZKRIEG
PANZER
Date: 23 Nov. 1943 Place: Several miles west of Kiev, Russia. The Russians have just liberated Kiev and are moving quickly to reach the German forces which are preparing for a last desperate attempt to halt the Russian advance.
BLITZKRIEG
Date: Spring 1940 Place: Northern France. The German blitzkrieg in the east was complete. Germany had turned its attention to the west: France. The German forces has penetrated the Ardennes and Meuse. The heroism of Dunkirk, the defense of the Aisne-Somme position, and the final collapse of the French armies in the south has all passed. And, now, the drive on Paris.

Set #2 STARSHIP TROOPERS and INVASION OF THE MUD PEOPLE
STARSHIP TROOPERS
Date: Fortieth Century Place: Arachnid planet of Sheol. The first all-out battle on the planet Sheol which will match equal forces of Terran and alien units. The outcome will set the course of the conflict, for the planet of Sheol is a key position in the solar war.
INVASION OF THE MUD PEOPLE
A Peruvian army battalion has been dispatched to a remote village area to investigate the destruction of many local dwellings and the disappearance of most of the villagers. Eye-witnesses have reported strange creatures appearing from scores of slimy mud holes which have oddly begun forming across the terrain.

Set #3: FALL OF THE THIRD REICH and ARMORCAR
FALL OF THE THIRD REICH
Date: March, 1945 Place: Remagen, Germany. The allies under General Eisenhower had reached the Rhine. The Germans had failed in destroying the Ludendorff railroad bridge, allowing several allied divisions to cross before it finally collapsed on March 17 ... and so, the allies began their drive on Berlin.
ARMORCAR
Date: 2 Feb. 1944 Place: Minsk, Russia. A German front-line unit is hard pressed for radio equipment and medical supplies. A relief convoy of armored cars must reach them through partisan-infested territory.

Set #4: MOUNT SURIBACHI and MIDDLE EARTH
MOUNT SURIBACHI
Date: 16 Feb. 1945 Place: Iwo Jima. The Japanese opened fire from Mount Suribachi as the marines landed on the porkchop-shaped island. Gunfire from the hill could cover the entire island, thus it was a critical objective if the Americans were to capture and utilize the all-important airfield. Mount Suribachi proved to be one of the most strongly defended positions in the Japanese theatre of war.
MIDDLE EARTH
Date: 1997 Place: MIDDLE EARTH. Through a maze of tunnels, crevices, and rocky passageways discovered leading from an inactive volcano in South America, a team of United Nations' researchers have undertaken a mission to an uncharted frontier, the center of the Earth. After a perilous journey spanning a period of several months, the mission has arrived at the Earth's core, a land of flames, steam, oceans, and unforeseen vegetation. And then the creatures of MIDDLE EARTH appeared ... unmatched by the most frightening horror stories created by man.

MISCELLANEOUS

CRYSTALS (Atari only) Price: $14.95 Cassette/$18.95 Diskette
A unique algorithm randomly produces fascinating graphics displays accompanied with tones which vary as the patterns are built. No two patterns are the same, and the combined effect of the sound and graphics are mesmerizing. CRYSTALS has been used in local stores to demonstrate the sound and color features of the Atari. Runs in 16K.

NORTH STAR SOFTWARE EXCHANGE (NSSE) LIBRARY
DYNACOMP now distributes the 23 volume NSSE library. These diskettes each contain many programs and offer an outstanding value for the purchase price. They should be part of every North Star user's collection. Call or write DYNACOMP for details regarding the contents of the NSSE collection.
Price: $9.95 each, $8.95 each (4 or more)
The complete collection may be purchased for $159.95

NEW

5¼" DISKETTES (soft sectored ten sectored) Price: $39.95 20 Diskettes
As you might imagine, DYNACOMP purchases diskettes in large quantities and at wholesale prices. We want to pass the savings along to you!

A Monthly Column

The World Inside The Computer

Fred D'Ignazio is a computer enthusiast and author of several books on computers for young people. He is presently working on two major projects: he is writing a series of books on how to create graphics-and-sound adventure games. He is also working on a computer mystery-and-adventure series for young people.
As the father of two young children, Fred has become concerned with introducing the computer to children as a wonderful tool rather than as a forbidding electronic device. His column appears monthly in **COMPUTE!**

The Talking Head

Fred D'Ignazio
Associate Editor

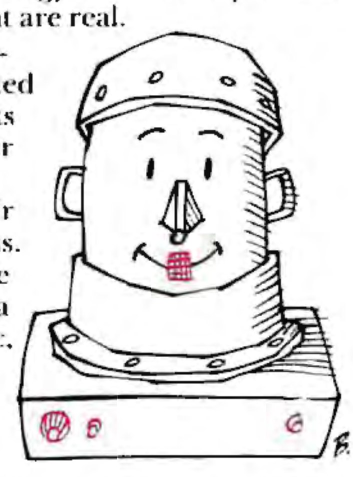
In her book, *Machines That Think*, Pamela McCorduck described the ancient popularity of talking heads. Wise men built the heads, then consulted them for useful advice. For example, a medieval pope, Sylvester II, supposedly built a talking head that answered only when spoken to. It was, in a sense, an early computer: to all questions, it gave only two answers – "yes" or "no"; yet, like modern computers, it was credited with having great wisdom and the ability to foretell the future.

A host of brilliant and famous men kept brazen (brass) heads as advisors, pets, and oracles in their homes. Albertus Magnus, for example, had a head in the form of a "a lovely woman who could speak." The head provided much sage advice, but occasionally its answers were flippant and mischievous. According to legend, the head so offended Albertus Magnus's pupil, Thomas Aquinas, that he kidnapped the head and burned it.

Conjuring Up A Talking Head

The first talking heads were products of alchemy and magic. They belong alongside all the other creatures of fantasy, myth, and legend.

Today, a thousand years after the first heads appeared, modern technology has made it possible to build talking heads that are real.



The heads are computer-controlled robots, mounted on robot bodies. Scientists are building them in their labs. Youthful hobbyists are building them in their workshops and bedrooms.

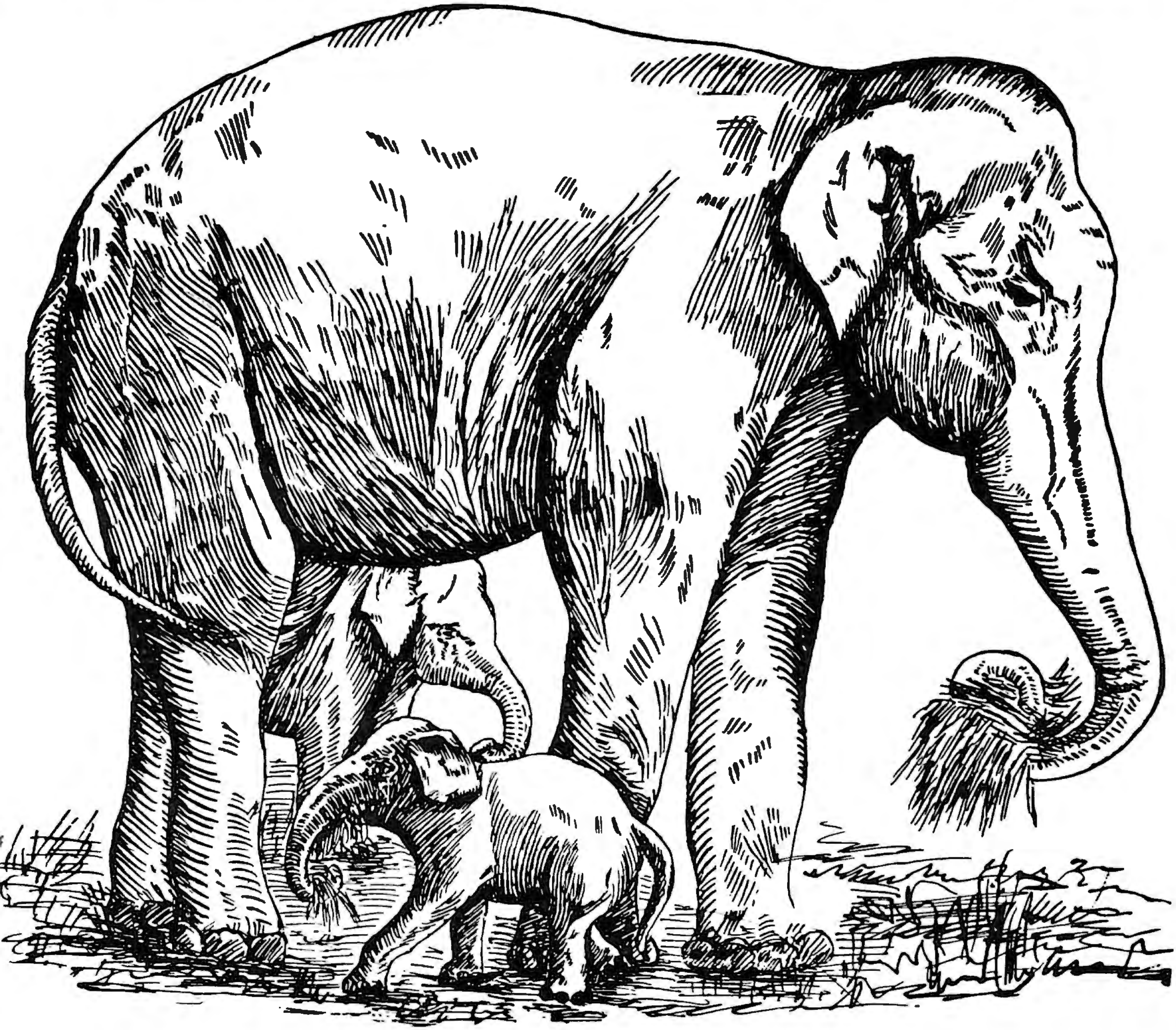
But if you don't have the time or skill to build a head out of metal, plastic, and servomotors, don't despair. You can "conjure" up a talking head on a program and call it forth from the world inside your computer. The head you create may not have the wisdom of the ages or be able to foresee the future, but it can become a great friend for your child.

Mirror, Mirror, On The Wall

In *Snow White*, the evil queen had a magical mirror on the wall. The mirror had a face, a voice, and a puckish, irreverent personality.

Your TV picture screen can be like the queen's mirror. When your child turns on the computer and runs the "talking head" program, the mirror will darken. Then, magically, a sleeping face will

Micro-Ed is EXPANDING!



Send for free educational catalogs ❊

❊ Specify: Pet • Vic • TI • Apple • TRS-80 color computer
• The New Commodore 64

you can telephone us at
612-926-2292

Micro-Ed, Inc.
PO Box 24156 Minneapolis, MN 55424

appear. Its eyes will blink open. It will wink at the child, talk to the child, and answer his or her questions in a musical voice.

The Face Of A Friend

Following this column is the "Talking Head" program for Atari, PET, and Apple computers. The talking head program is just the first step toward a much more elaborate "computer friend" program. I described this program in my last column (**COMPUTE!**, August 1982, #27). Next month we will give the friend the ability to play with your child. The following month, we will give your child the ability to teach the friend and shape its character.

The "Talking Head" Program

Lines 40 to 130: Program initialization.

Lines 500 to 550: The "Friend Master."

Consists entirely of GOSUBs to subroutines. You need to think of the interaction between your child and the friend in terms of episodes or frames. Each time the friend talks, that's a frame. Each time your child responds, that's another frame. Each GOSUB in this section handles a single frame. You can add new frames by adding new GOSUBs at this master level.

Lines 1000 to 1110: "Friend Wake-Up."

In this frame, the friend's face appears. A bell rings, the friend wakes up, blinks its eyes open, and winks at the child.

Lines 2000 to 3110: "The Friend Talks."

This is a general-purpose talking subroutine. It enables the friend to read in sequences of DATA statement messages and print them on the screen, in large letters, to the right of the friend's face. The friend can handle words up to nine letters long. It prints words, one per line, on up to five lines per screen. If the friend has a message of more than five words, it will take more than a single screen. That's okay. Each of the friend's messages can be up to nine screens (45 words) long.

Lines 3200 to 3270: Friend accepts child's name.

Lines 4000 to 4880: Sound effects – including the wake-up bell and the friend's voice.

Lines 5000 to 5470: Drawing the friend's face – including the basic face (ski cap, ears, nose, and chin).

Additional routines to animate mouth (5200-5280) and eyes (5300-5470).

Lines 5500 to 5550: Clear message window for new message.

Lines 6000 to 6022: Friend's messages to the child.

Each new message should begin on line 6000 + some multiple of 10 (for example, 6010, 6020, 6030, etc.).

Each new message begins with the number of message screens in the message (a number from 1 to 9).

Each list of words to appear on a single screen ends with a "-1."

The friend's name (line 6011) is given as "GED." To give the friend a new name, just replace the old name with one of your choosing.

After the friend has asked and received the child's name (the first GOSUB 2010 and GOSUB 3210), you can add the child's name to any message by placing the token character "*" in the message list (e.g., 6022 TO,SEE,YOU,*,-1).

Warning And Acknowledgment

Remember, this program is just the beginning. It makes a good talking head. But, as yet, the head is not good at answering. Since the head cannot carry on a conversation with the child, it is not yet a computer friend.

Next month we'll give the head the ability to carry on a conversation with your child and play games. Then it will start being a real friend.

I would like to thank Bruce Mitchell for some valuable programming assistance. Also, thanks to Richard M. Kruse for the doorbell sound.

Program 1. Atari Version

```
40 GRAPHICS 2+16
50 FOR P=1 TO 800:NEXT P
100 REM *** DIMENSION VARIABLES
110 DIM M$(9):REM * MESSAGE
120 N=1:REM * MESSAGE POINTER
130 DIM NAME$(9):REM * CHILD'S NAME
500 REM *** FRIEND MASTER
510 GOSUB 1010:REM * FRIEND WAKE-UP
520 GOSUB 2010:REM * FRIEND TALK
530 GOSUB 3210:REM * STORE CHILD'S NAME
540 GOSUB 2010:REM * FRIEND TALK
550 END
1000 REM *** FRIEND WAKE-UP
1010 GOSUB 5010:REM * DRAW FACE
1020 GOSUB 5410:REM * DRAW SLEEP EYES
1030 GOSUB 5210:REM * DRAW CLOSED MOUTH
1035 FOR P=1 TO 800:NEXT P
1040 GOSUB 4010:REM * WAKE-UP BELL
1050 GOSUB 5460:REM * DRAW OPEN EYES
1060 FOR P=1 TO 600:NEXT P
1070 GOSUB 5320:REM * WINK EYE
1080 FOR P=1 TO 100:NEXT P
1085 M=0:GOSUB 4820:REM * WINK NOISE
1090 GOSUB 5460:REM * DRAW OPEN EYES
1100 FOR P=1 TO 800:NEXT P
1110 RETURN
2000 REM *** FRIEND TALK
2005 RESTORE 6000+N*10:REM * SELECT MESS
     AGE
2006 N=N+1:REM * SET POINTER TO NEXT SET
      OF FRIEND MESSAGES
2010 READ SNUM:REM * SNUM = NUMBER OF SC
     REENS IN CURRENT SET OF FRIEND MESS
     AGES
2015 FOR K=1 TO SNUM
2020 GOSUB 3010:REM * FRIEND TALK--1 SCR
```

```
     EEN
2033 FOR P=1 TO 800:NEXT P
2035 GOSUB 5510:REM * CLEAR MESSAGE WIND
     OW
2040 NEXT K
2050 RETURN
3000 REM *** FRIEND TALKING--1 SCREEN
3010 PY=2:REM * MESSAGE VERTICAL (Y) STA
     RT LOCATION
3020 PY=2:REM * MESSAGE VERTICAL (Y) STA
     RT LOCATION
3030 PX=14:REM * HORIZONTAL (X) CENTER O
     F MESSAGE ON SCREEN
3040 READ M$
3050 IF M$="-1" THEN RETURN
3051 IF M$="*" THEN M$=NAME$
3055 GOSUB 5260:REM * OPEN MOUTH
3060 POSITION INT(PX-(LEN(M$)/2)+0.5),PY
     :REM * CENTER LINE
3070 PRINT #6;M$
3075 GOSUB 4810:REM * FRIEND SOUND
3080 FOR P=1 TO 10:NEXT P:REM * KEEP MOU
     TH OPEN
3090 GOSUB 5210:REM * CLOSE MOUTH
3095 FOR P=1 TO 50:NEXT P:REM * KEEP MOU
     TH CLOSED
3100 PY=PY+2
3110 GOTO 3040
3200 REM *** FRIEND ASKS CHILD'S NAME
3210 OPEN #1,4,0,"K:"
3215 POSITION 11,4
3217 FOR I=1 TO 9
3220 GET #1,A
3230 IF A=155 THEN 3265
3240 PRINT #6;CHR$(A);
3250 NAME$(LEN(NAME$)+1)=CHR$(A)
3260 NEXT I
3265 FOR P=1 TO 75:NEXT P
3267 GOSUB 5510:REM * CLEAR MESSAGE WIND
     OW
3270 CLOSE #1:RETURN
4000 REM *** WAKE-UP BELL
4010 BEL=105:TIM=7.5:GOSUB 4040
4020 BEL=132:TIM=8.5:GOSUB 4040
4030 SOUND 0,0,0,0:RETURN
4040 VLM=15:INC=0.79+TIM/50
4050 SOUND 0,BEL,10,VLM
4060 VLM=VLM*INC
4070 IF VLM>1 THEN 4050
4080 RETURN
4625 FOR P=1 TO 15:NEXT P
4770 RETURN
4800 REM *** FRIEND VOICE
4810 M=INT(RND(1)*51)+15
4820 FOR A=M+25 TO M STEP -8
4830 SOUND 0,A,10,10
4840 FOR T=1 TO 10
4850 NEXT T
4860 NEXT A
4875 SOUND 0,0,0,0
4880 RETURN
5000 REM *** FRIEND'S FACE
5010 GRAPHICS 2+16
5040 POSITION 2,1:PRINT #6;"{3 SPACES}*"

5050 POSITION 2,2:PRINT #6;"  / \"
5060 POSITION 2,3:PRINT #6;" ====="
5070 POSITION 2,4:PRINT #6;"/{5 SPACES}\
     "

5090 POSITION 1,6:PRINT #6;"<:   ^   :>"
5100 POSITION 2,9:PRINT #6;"\_____/"

5110 RETURN
5200 REM *** CLOSE MOUTH
5210 POSITION 2,7:PRINT #6;":{5 SPACES}:
     "
5220 POSITION 2,8:PRINT #6;":  ---  :"
5230 RETURN
5250 REM *** OPEN MOUTH
5260 POSITION 2,7:PRINT #6;":  ___  :"
5270 POSITION 2,8:PRINT #6;":  \_/  :"
5280 RETURN
5300 REM *** LEFT EYE WINK
5320 POSITION 2,5:PRINT #6;": O - :"
5330 FOR P=1 TO 150:NEXT P
5340 RETURN
5400 REM *** EYES ASLEEP
5410 POSITION 2,5:PRINT #6;": - - :"
5440 RETURN
5450 REM *** EYES AWAKE
5460 POSITION 2,5:PRINT #6;": O O :"
5470 RETURN
5500 REM *** CLEAR MESSAGE WINDOW
5510 FOR Y=2 TO 8 STEP 2
5520 POSITION 10,Y
5530 PRINT #6;"{9 SPACES}"
5540 NEXT Y
5550 RETURN
6000 REM *** MESSAGES
6010 DATA 3
6011 DATA HI, I'M, GED,-1
6012 DATA YOU,TURNED,ME,ON,-1
6013 DATA WHO'S,OUT,THERE?,-1
6020 DATA 2
6021 DATA I'M,SO,HAPPY,-1
6022 DATA TO,SEE,YOU,*,-1
```

Program 2. PET/CBM Version with suggested changes for Apple II+

```
40 PRINT"{CLEAR}":REM HOME FOR APPLE
50 FOR P=1 TO 800:NEXT P
100 REM *** DIMENSION VARIABLES
120 N=1:REM * MESSAGE POINTER
355 GOSUB 5260:REM * OPEN MOUTH
500 REM *** FRIEND MASTER
510 GOSUB 1010:REM * FRIEND WAKE-UP
520 GOSUB 2010:REM * FRIEND TALK
530 GOSUB 3210:REM * STORE CHILD'S NAME
540 GOSUB 2010:REM * FRIEND TALK
550 PRINT:PRINT:PRINT:PRINT:PRINT:END
1000 REM *** FRIEND WAKE-UP
1010 GOSUB 5010:REM * DRAW FACE
1020 GOSUB 5410:REM * DRAW SLEEPING EYES
1030 GOSUB 5210:REM * DRAW CLOSED MOUTH
1035 FOR P=1 TO 800:NEXT P
1040 GOSUB 4000:REM * WAKE-UP BELL
1050 GOSUB 5460:REM * DRAW OPEN EYES
1060 FOR P=1 TO 600:NEXT P
1070 GOSUB 5320:REM * WINK EYE
1080 FOR P=1 TO 100:NEXT P
1085 M=0:GOSUB 4820:REM * WINK NOISE
1090 GOSUB 5460:REM *DRAW OPEN EYES
1100 FOR P=1 TO 800:NEXT P
1110 RETURN
2000 REM *** FRIEND TALK
2005 REM * SELECT MESSAGE
2006 N=N+1:REM * SET POINTER TO NEXT SET OF FRI
     END MESSAGES
2010 READ SNUM:REM * SNUM = NUMBER OF SCREENS I
     N CURRENT SET OF FRIEND MESSAGES
2015 FOR K=1 TO SNUM
2020 GOSUB 3010:REM * FRIEND TALK--1 SCREEN
2033 FOR P=1 TO 1000:NEXT P
2035 GOSUB 5510:REM * CLEAR MESSAGE WINDOW
2040 NEXT K
2050 RETURN
3000 REM *** FRIEND TALKING--1 SCREEN
3010 PY=3:REM * MESSAGE VERTICAL (Y) START LOCA
     TION
```

```
3030 PX=14:REM * HORIZONTAL NEW(X) CENTER OF ME
     SSAGE ON SCREEN
3040 READ M$
3050 IF M$="-1" THEN RETURN
3051 IF M$="*" THEN M$=N$
3052 REM FOLLOWING WOULD BE "VTAB PY:HTAB PX-LE
     N(M$)/2+.5" ON APPLE
3060 POKE 216,PY:PRINT"{UP}";TAB(PX-(LEN(M$)/2)
     +0.5);:REM * CENTER LINE
3070 PRINT M$:GOSUB 5250
3075 GOSUB 4810:REM * FRIEND SOUND
3080 FOR P=1 TO 50:NEXT P:REM * KEEP MOUTH OPEN

3090 GOSUB 5200:REM * CLOSE MOUTH
3095 FOR P=1 TO 100:NEXT P:REM * KEEP MOUTH CLO
     SED
3100 PY=PY+2
3110 GOTO 3040
3200 REM *** FRIEND ASKS CHILD'S NAME
3210 REM
3215 REM USE "VTAB 3:HTAB 10:INPUT N$" HERE FOR
      APPLE
3220 PRINT"{HOME}{03 DOWN}";TAB(10);:INPUT N$
3265 FOR P=1 TO 75:NEXT P
3267 GOSUB 5510:REM * CLEAR MESSAGE WINDOW
3270 RETURN

4000 REM *** WAKE-UP BELL
4005 REM DELETE LINES 4010-4070 FOR APPLE
4010 BEL=105:TM=30:GOSUB 4040
4020 BEL=132:TM=20:GOSUB 4040:RETURN
4040 VLM=15:INC=0.79+TIM/50
4045 POKE59467,16:POKE 59466,51
4050 POKE 59464,BEL
4060 FOR W=1 TO TM*10:NEXT
4070 POKE 59467,0
4080 RETURN
4625 FOR P=1 TO 15:NEXT P
4760 NEXT A
4770 RETURN
4800 REM *** FRIEND VOICE
4801 REM DELETE LINES 4810-4880 FOR APPLE AND C
     HANGE LINES 4810 AND 4820
4802 REM TO "RETURN"
4810 M=INT(RND(1)*51)+15
4820 POKE 59467,16:POKE 59466,51
4825 FOR A=M+25 TO M STEP -8
4830 POKE 59464,A
4840 FOR T=1 TO 10
4850 NEXT T,A
4875 POKE 59467,0
4880 RETURN
5000 REM *** FRIEND'S FACE
5010 PRINT"{CLEAR}":REM USE "HOME" FOR APPLE
5040 PRINT "     *"
5050 PRINT "    / \"
5060 PRINT "   ====="
5070 PRINT " /       \"
5090 PRINT:PRINT "<:   ^   :>"
5100 PRINT::PRINT:PRINT " \$$$$$/":REM "$" IS U
     NDERLINE
5110 PRINT
5200 REM *** CLOSE MOUTH
5205 REM USE "VTAB 7" ON APPLE
5210 PRINT"{HOME}{07 DOWN} :       :"
5215 REM USE "VTAB 8" ON APPLE
5220 PRINT"{HOME}{08 DOWN} : --- :"
5230 RETURN
5250 REM *** OPEN MOUTH
5255 REM USE "VTAB 7" ON APPLE
5260 PRINT"{HOME}{07 DOWN} : $$$ :":REM "$" IS
     UNDERLINE
5265 REM USE "VTAB 8" ON APPLE
5270 PRINT"{HOME}{08 DOWN} : \$/ :"
5280 RETURN
5300 REM *** LEFT EYE WINK
5305 REM USE "VTAB 5" ON APPLE
5320 PRINT"{HOME}{05 DOWN} : O - :"
5330 FOR P=1 TO 150:NEXT P
5340 RETURN
5400 REM *** EYES ASLEEP
5405 REM USE "VTAB 5" ON APPLE
5410 PRINT "{HOME}{05 DOWN} : - - :"
5420 RETURN
5450 REM *** EYES AWAKE
5455 REM USE "VTAB 5" ON APPLE
5460 PRINT"{HOME}{05 DOWN} : O O :"
5470 RETURN
5500 REM *** CLEAR MESSAGE WINDOW
5505 REM USE "VTAB 1" ON APPLE INSTEAD OF "{
     HOME}"
5510 PRINT"{HOME}":FOR I=1 TO 10
5530 PRINTTAB(10);"                    "
5540 NEXT I
5550 RETURN
6000 REM *** MESSAGES
6010 DATA 3
6011 DATA HI, I'M, GEB,-1
6012 DATA YOU,TURNED,ME,ON,-1
6013 DATA WHO'S,OUT,THERE?,-1
6020 DATA 2
6021 DATA I'M,SO,HAPPY,-1
6022 DATA TO,SEE,YOU,*,-1
```

©

For Atari and Apple, this is the first of a three-part series on PILOT. This month, an Apple version (Microsoft BASIC) – and next month an Atari BASIC PILOT will be published in Part II. The Apple version requires Applesoft, 32K memory, and one disk drive.

Part I

Turtle PILOT

Alan W. Poole
Loomis, CA

How would you like a powerful new language for your computer that combines PILOT, turtle graphics, and all commands and functions? The programs at the end of this article create a version of PILOT which contains all of these features. Best of all, unlike most languages available for the Apple, this language isn't going to cost you a fortune. This version of PILOT, which I have named Turtle PILOT, has been patterned after Atari PILOT. Most Atari PILOT programs can be converted to Turtle PILOT without very many changes.

At the end of this article are two program listings. The Turtle PILOT Editor is used for typing PILOT programs. The Turtle PILOT Translator writes a program in Applesoft which is equivalent to a PILOT program typed with the Editor. Both programs require an Apple with Applesoft, 32K, and one disk drive. Also included is an example showing everything typed and printed on the screen while entering and translating a program, along with a sample RUN after it was translated.

Introduction To PILOT

PILOT is a simple language and is very easy to learn, especially if you already understand BASIC. In this article I will assume that the reader is familiar with Applesoft BASIC. To get an idea of how PILOT works, consider the following program and explanations for each line.

```
1 *QUIZ
2 T:HOW MUCH IS 8+4?
3 A:
4 M:12,TWELVE
5 TY:THAT'S CORRECT.
6 TN:NO, TRY AGAIN.
7 JN:*QUIZ.
```

The first line is a label, which is used to identify sections and modules (modules are subroutines) of a program. Labels always begin with an asterisk. In the second line the T is the instruction name for Type. Everything following the colon will be displayed on the screen. The third line Accepts a response from the user. Line four uses the Match instruction. Each item following the colon is compared with the last response. The Y in line five is the Yes conditioner. The Yes conditioner causes the instruction to be executed only if the last Match succeeded. The N in line six is the No conditioner. A line with a No conditioner will be executed only if the last Match failed. The last line causes a Jump back to the first line if the question was answered incorrectly. Notice that a label is used instead of a line number. The line numbers are not actually part of the program, but are used only to make editing easier.

Parts Of An Instruction

The order of the parts of an instruction is important. Below is a description of each of the elements in an instruction. Although an instruction does not have to contain all of the optional elements, the elements that are included must be in the order they are given below.

1. Instruction Name – The instruction name is a single letter and always comes first. It is required with every line other than a label, since labels are not considered instructions.

2. Conditioner – The conditioner is either a Y or an N. The conditioner is optional and may be used with any instruction.

3. Expression – An instruction with an expression is executed only when the expression is true. The expression must be placed between parentheses. Expressions are optional and may be used with any instruction. Below are some examples of instructions that include expressions.

```
T(S>100):VERY GOOD.
J(L=SQR(N)):*START
AY(X=0 AND MID$(I$,M,1)=STR$(J)):
```

4. Colon – Every instruction must have a colon.

5. Object – The last part of an instruction is the object. Everything following the colon is called the object of the instruction. An object is optional with some instructions.

The Turtle PILOT Instructions

There are 11 instructions in Turtle PILOT, not including the turtle graphics commands. Below is an explanation of each instruction. At the end of each explanation are a few samples of the instruction.

T: Type. The Type instruction will print everything in the object on the screen. The Type instruction has an advantage over BASIC's PRINT command. With the Type instruction, words at the end of a line will not usually be divided between two lines of the screen. A string variable can be Typed by placing the name of the string variable in the object preceded and followed by a pound sign (#). An ampersand (&) placed at the end of the object will cause the next printed character to continue on the same line. The ampersand will not be displayed on the screen. A Type instruction without an object will print a blank line.

```
T:
T:HELLO,$NAME$ &
TY:YOUR TOTAL SCORE IS #S#.
```

A: Accept. The Accept instruction inputs a response from the user. The Accept instruction is very similar to BASIC's INPUT command, with the advantage that any character can be typed without an error occurring. An object is not required, but the input can be assigned to a variable by placing the variable name in the object.

```
A:
A:NAME$
A(M(K)=0):X
```

M: Match. The Match instruction compares all of the items in the object to the last response. The items in the object are separated by commas and may include string variable names. If any of the items match with the last response, the Y conditioner is set. Otherwise, the N conditioner is set. The Match instruction does not compare the item with just the start of the last response. It searches for the word through the entire response. For instance, GO would Match with GOING, INGOT, and LINGO. You may put up to 25 items in the object.

```
M:HI
M:A,E,I,O,U,Y,A$,B$
MN(Z<20):END,STOP
```

J: Jump. The Jump instruction causes a branch to the line with the label that matches the label in the object of the Jump instruction. This instruction resembles BASIC's GOTO command, except a label is used instead of a line number.

```
J:*START
JY:*PART TWO
```

U: Use. The Use instruction is for Using modules (subroutines) in a PILOT program. It is similar to the Jump instruction, but the computer remembers the line from which it came. Program execution will continue at the line with a label that matches the label in the object until an End instruction is encountered. The End instruction will cause a return to the line following the Use instruction. The Use instruction is like BASIC's GOSUB command, with labels used instead of line numbers.

```
U:*PRINT
UY:*FIRST
```

E: End. The End instruction will terminate the program unless a Use instruction has been executed. If a Use instruction has been executed, program execution will continue at the line following the Use instruction. No object is used with an End instruction. The End instruction is similar to BASIC's RETURN and END commands.

```
E:
E(N=T):
```

R: Remark. The Remark instruction is not executed and is used only for program documentation.

```
R:THIS IS A REMARK
```

C: Compute. The Compute instruction may be used for numeric calculations or string manipulation.

```
C:N=N+1
C:S(K)=SIN(A*10)
CY:A$="ABC"
C(T=1):Z$(N,1)=X$+RIGHT$(I$,3)
```

B: Basic. The object of the Basic instruction may contain any Applesoft commands.

```
B:HOME
B:GET K$:PRINT K$;
BY:HPLOT 10,Y TO X,50
BN(V>3):COLOR=2: HLIN 10,20 AT Y
```

S: Sound. The object of the Sound instruction should contain a number from 1-31 for the pitch of the note, a comma, and a number from 1-255 for the duration. The notes range from C below middle C for a pitch value of 1 to F# above C above middle C for a value of 31. These values are the same as the pitch values used in Atari PILOT.

```
S:10,200
S:30,D
SY:P,SQR(L*2)
```

G: Graphics. The Graphics instruction precedes all turtle graphics commands, which will be explained next month in Part II.

Variables In Turtle PILOT

You may use any variable in a Turtle PILOT program that you could use in an Applesoft program, except variables beginning with Q. The reason for not using Q variables will be explained next month in Part II of this article. You may also use any of Applesoft's mathematical and string functions.

Using The Editor

The Editor has 11 commands to help you in typing Turtle PILOT programs. The Editor has two modes. First there is the command mode, indicated by a prompt. Any of the Editor's 11 commands may be typed in the command mode. The second mode is the program mode, indicated by a line number that is automatically printed on the left side of the screen. The program mode may be entered through several of the Editor commands. PILOT programs are typed in the program mode. To return to the command mode from the program mode, press RETURN without typing anything. When you have finished using the Editor, press the ESC key. Accidentally pressing the ESC key can be corrected by pressing RETURN immediately. Following is a description of each of the Editor commands.

The ADD command is used for entering the program mode to start or continue a program. It will ADD lines to the end of the program. The ADD command may also be followed by a line number. The lines ADDed to the program will then start at the line number specified. All the lines of the program currently in memory from that line to the end of the program will be erased.

The LIST command will LIST the program in memory. To LIST a single line, type the line number following the LIST command. A range of lines can be LISTed by typing the first and last line numbers separated by a comma. All the lines of a module may be LISTed by specifying the label of the module. Pressing RETURN will abort a LISTing.

The EDIT command is used to change line(s) of a program. It may be followed by a single line number or a range of line numbers with the first and last line numbers separated by a comma. The line to be EDITed will appear on the screen with the cursor at the beginning of the line.

The INSERT command is used for adding a line in the interior of a program. The INSERT command must be followed by a line number. No lines of the program will be deleted, but the numbers of the lines after the INSERTed line will be raised by one.

The DELete command is used for erasing line(s) from a program. A single line, a range of lines, or a label of a module may be specified. The numbers of the lines following the DELeted lines will be lowered.

The NEW command erases the program in memory.

The LOAD command will read a PILOT program from the disk and append it to the program in memory. The NEW command must be used first if the program in memory is not wanted. The name of the program is specified by following the LOAD command by that name.

The SAVE command will store the program in memory on the disk under the name specified. A program must be SAVEd before it can be translated.

The MEM command prints the number of free bytes available.

The CATalog command will print a catalog of the files on the disk. Turtle PILOT programs will appear as text files with ".P" at the end of their names.

The PR# command changes output to the specified slot, allowing a printer to be used with the Editor.

Using The Translator

The Translator is used to translate your PILOT programs into Applesoft programs. When you RUN the Translator, it will ask you to type the name of the program to be translated. Make sure the program to be translated has been SAVEd on disk. After you have typed the name, the translating will automatically be done. The translated Applesoft program will be sent to a text file on the disk that can be EXECuted later to load it into memory. The computer will tell you what to type to load the program into memory. Once it is in memory, you can RUN, SAVE, LOAD, and LIST it just like a normal Applesoft program.

Errors

Major syntax errors will be caught by the Editor, and the Editor will have you retype the line. Most other errors will be detected by the Translator while the program is being translated. If an error occurs during a translation, the computer will print "ERROR IN PILOT LINE NO." followed by the number of the line in which the error occurred. The computer will then automatically RUN the Editor. You should LOAD the program you were translating, correct the error, SAVE the program, and RUN the Translator again. Since this is a time consuming process, you should look the program over carefully before trying to translate. If an error occurs while a translated program is RUNning, divide the line number by ten to calculate the corresponding PILOT line number.

Typing The Programs

A typing error in the Translator program could produce disastrous results. To make it easier to find mistakes, the following line should be included when the Translator is first typed.

```
15 POKE 216,0: GOTO 30
```

This will cause the translated program to be sent to the screen instead of the disk, and error messages will be printed normally. When all mistakes have been corrected, delete line 15.

The turtle PILOT Editor must be SAVEd under the name EDITOR, and the Turtle PILOT Translator must be SAVEd under the name TRANSLATOR. Make sure you SAVE the Translator immediately after typing it and before you RUN it, since it erases itself when it is finished.

This is only the first in a series of three articles about Turtle PILOT. In Part II we'll cover turtle graphics, which is probably the best feature of Turtle PILOT, and we'll take a look at some other features of the language. In Part III we'll convert an Atari PILOT program to Turtle PILOT. If you are already familiar with turtle graphics, experiment with the commands listed in line 10230 of the Translator program and see what you can discover.

The author has offered to make a copy of the programs for you. Send a blank disk (specify DOS 3.2 or 3.3), a stamped, self-addressed mailer, and a $3.00 copying fee to:

Alan Poole
4728 King Road
Loomis, CA 95650

Program 1.

```
1 REM  TURTLE PILOT TRANSLATOR
2 REM  BY ALLAN POOLE
10 GOSUB 10000
17 REM
18 REM *** OPEN EXEC FILE ***
19 REM
20 N$ = LEFT$ (N$,LEN (N$) - 1) + "EXEC": PRI
   NT D$"OPEN"N$: PRINT D$"DELETE"N$
21 PRINT D$"OPEN"N$: PRINT D$"WRITE"N$
30 POKE 33,30: LIST 50000 - 60000: POKE 33,40

40 PRINT "5 GOSUB 50000"
47 REM
48 REM *** MAIN LOOP ***
49 REM
50 FOR LN = 1 TO NL:LN$ = ""
60 LN$ = LN$ + STR$ (LN * 10)
70 I = 0: FOR L = 1 TO 12: IF LEFT$ (P$(LN),1
   ) = I$(L) THEN I = L
80 NEXT : IF I = 0 THEN 15000
90 GOSUB 200: GOSUB 300: GOSUB 400
100 ON I GOSUB 500,1000,1500,2000,2500,3000,35
    00,4000,4500,5000,6000,6500
110 PRINT LN$: NEXT
117 REM
118 REM *** END OF PROGRAM ***
119 REM
120 PRINTLN*10;"END":PRINT "?";CHR$(34);"YOUR ~
    TRANSLATED PROGRAM IS IN MEMORY";CHR$
    (34)
121 PRINT D$"CLOSE"
130 PRINT : PRINT "TO LOAD YOUR TRANSLATED PRO
    GRAM INTO":
131 PRINT "MEMORY,TYPE "CHR$ (34);"EXEC "N$; C
    HR$ (34)
140 NEW
197 REM
198 REM *** SPLIT PILOT LINE AT COLON ***
199 REM
200 FOR L = 1 TO LEN (P$(LN)): IF MID$ (P$(LN)
    ,L,1) = ":" THEN T = L:L = 300
210 NEXT
215 IF LEFT$ (P$(LN),1) = "*" THEN L$ = "*":R$
     = P$(LN): RETURN
220 L$ = LEFT$ (P$(LN),T - 1): IF T = LEN (P$(
    LN)) THEN R$ = "": RETURN
230 R$ = RIGHT$ (P$(LN), LEN (P$(LN)) - T)
240 T$ = L$: GOSUB 11000:L$ = T$
250 IF LEFT$ (L$,1) = "G" THEN T$ = R$: GOSUB ~
    11000:R$ = T$
260 RETURN
297 REM
298 REM *** FIND CONDITIONER ***
299 REM
300 C = 0: IF LEN (L$) < 2 THEN RETURN
310 IF MID$ (L$,2,1) = "Y" THEN LN$ = LN$ + "I
    F QC=1 THEN ":C = 1
320 IF MID$ (L$,2,1) = "N" THEN LN$ = LN$ + "I
    F QC=0 THEN ":C = 2
330 RETURN
397 REM
398 REM *** FIND EXPRESSION ***
399 REM
400 EX$ = "": IF RIGHT$ (L$,1) < > ")" THEN RE
    TURN
410 T = 0: FOR L = 1 TO LEN (L$) - 1: IF MID$ ~
    (L$,L,1) = "(" THEN T = L:L =300
420 NEXT :EX$ = MID$ (L$,T + 1, LEN (L$) - T -
     1):LN$ = LN$ + "IF"+EX$ + "THEN"
430 RETURN
497 REM
498 REM *** T: INSTRUCTION ***
499 REM
500 LN$ = LN$ + "QT$=" + CHR$ (34): IF R$ = ""
     THEN LN$ = LN$ + CHR$ (34) + "
501 GOSUB 51000": RETURN
510 FOR L = 1 TO LEN (R$):T$ = MID$ (R$,L,1)
520 IF T$ = "$" THEN 600
530 IF T$ = "#" THEN 700
540 LN$ = LN$ + T$
550 NEXT :LN$ = LN$ + CHR$ (34) + ":GOSUB 5100
    0": RETURN
600 IF L > LEN (R$) - 2 THEN 540

610 T = 0: FOR L1 = L + 2 TO LEN (R$): IF MID$
    (R$,L1,1) = "$" THEN T=L1:L1=300
620 NEXT : IF T = 0 THEN 540
630 LN$ = LN$ + CHR$ (34) + "+" + MID$(R$,L + ~
    1,T - L) + "+" + CHR$ (34)
631 L = T: GOTO 550
700 IF L > LEN (R$) - 2 THEN 540
710 T = 0: FOR L1 = L + 2 TO LEN (R$): IF MID$
    (R$,L1,1)="#"THEN T=L1:L1 = 300
720 NEXT : IF T = 0  THEN 540
730 LN$=LN$+CHR$(34)+"+STR$(" + MID$ (R$,L + 1
    ,T - L - 1) + ")" + "+" +CHR$(34)
731 L = T: GOTO 550
997 REM
998 REM *** A: INSTRUCTION ***
999 REM
1000 LN$ = LN$ + "GOSUB 52000"          1010 IF
     R$ = "" THEN RETURN
1020 IF RIGHT$ (R$,1) = "$" THEN LN$ = LN$ + ":
     " + R$ + "=QI$": RETURN
1030 LN$ = LN$ + ":" + R$ + "=VAL(QI$)": RETURN

1497 REM
1498 REM *** M: INSTRUCTION ***
1499 REM
1500 FOR L = 1 TO 25:M$(L) = "": NEXT : IF R$ =
     "" THEN 15000
1510 T=1:FOR L=1 TO LEN(R$):IF MID$(R$,L,1)< > ~
     ","THEN M$(T)=M$(T)+MID$(R$,L,1)
1511 GOTO 1530
```

```
1520 T = T + 1
1530 NEXT
1540 FOR L = 1 TO T
1541 IF RIGHT$(M$(L),1)="$"THEN LN$=LN$+"Q$(" +
     STR$ (L) + ")=" + M$(L)+":"
1542 GOTO 1560
1550 LN$ = LN$ + "Q$(" + STR$ (L) + ")=" + CHR$
     (34) + M$(L) + CHR$ (34) + ":"
1560 NEXT :LN$ = LN$ + "GOSUB 53000": RETURN
1997 REM
1998 REM *** J: INSTRUCTION ***
1999 REM
2000 IF R$ = "" THEN 2100
2010 IF LEFT$ (R$,1) < > "*" THEN R$ = "*" + R$

2020 T = 0: FOR L = 1 TO NL: IF P$(L) = R$ THEN
     T = L:L = 2500
2030 NEXT : IF T = 0 THEN 15000
2040 LN$ = LN$ + "GOTO" + STR$ (T * 10): RETURN

2100 T = 0: FOR L = LN TO 1 STEP - 1: IF LEFT$ ~
     (P$(L),1) = "A" THEN T = L:L = 0
2110 NEXT : IF T = 0 THEN 15000
2120 GOTO 2040
2497 REM
2498 REM *** U: INSTRUCTION ***
2499 REM
2500 LN$ = LN$ + "QU=QU+1": IF LEFT$ (R$,1) < >
     "*" THEN R$ = "*" + R$
2510 T = 0: FOR L = 1 TO NL: IF P$(L) = R$ THEN
     T = L:L = 2500
2520 NEXT : IF T = 0 THEN 15000
2530 LN$ = LN$ + ":GOSUB" + STR$ (T * 10): RETU
     RN
2997 REM
2998 REM *** E: INSTRUCTION ***
2999 REM
3000 LN$ = LN$ + "IF QU=0 THEN END"
3010 PRINT LN * 10 + 5;
3020 IF C=1 THEN PRINT "IF QC=1 THEN";  3030 IF
     C=2 THEN PRINT "IF QC=0 THEN";
3040 IF EX$ < > "" THEN PRINT "IF";EX$;"THEN";
3050 PRINT "QU=QU-1:RETURN": RETURN
3497 REM
3498 REM *** C: INSTRUCTION ***
3499 REM
3500 LN$ = LN$ + R$: RETURN
3997 REM
3998 REM *** R: INSTRUCTION ***
3999 REM
4000 RETURN
4497 REM
4498 REM *** S: INSTRUCTION ***
4499 REM
4500 T = 0: FOR L = 1 TO LEN (R$): IF MID$ (R$,
     L,1) = "," THEN T = L:L = 255

4510 NEXT :LN$ = LN$ + "POKE 768,Q$(" + LEFT$ (
     R$,T - 1) + ")
4511 POKE 769," + RIGHT$ (R$, LN (R$) - T) + ":
     CALL 770": RETURN
4997 REM
4998 REM *** G: INSTRUCTION ***
4999 REM
5000 IF R$ = "" THEN LN$ = LN$ + "POKE -16304,0
     :POKE -16297,0": RETURN
5009 REM     FIND LOOPS
5010 F = 0:IF VAL(R$) > 0 THEN LN$=LN$+ "FOR Q1
     =1 TO"+STR$( VAL (R$)) + ":":F=1
5011 R$ = LEFT$ (R$, LEN (R$) - 1)
5012 R$ = RIGHT$ (R$, LEN (R$) - LEN (STR$ ( VA
     L (R$))) - 1)
5019 REM     FIND INDIVIDUAL COMMANDS
5020 FOR L = 1 TO 6:GL$(L) = "": NEXT
5030 T = 1: FOR L = 1 TO LEN (R$)
5040 IF MID$ (R$,L,1) < > ";" THEN GL$(T) =GL$(
     T) + MID$ (R$,L,1): GOTO 5060
5050 T = T + 1
5060 NEXT
5069 REM     TRANSLATE EACH COMMAND
5070 FOR L = 1 TO T
5080 GC = 0:FOR L1=1 TO 11:IF LEFT$ (GL$(L), LE
     N (G$(L1))) < > G$(L1) THEN 5110
5090 GC = L1:L1 = 11: IF GL$(L) = G$(GC) THEN G
     L$(L) = "": GOTO 5110
5100 GL$(L) = RIGHT$ (GL$(L), LEN (GL$(L)) - LE
     N (G$(GC)))
5110 NEXT
5120 IF GC = 0 THEN 15000
5130 ON GC GOSUB 5200,5250,5300,5350,5400,5450,
     5500,5550,5600,5650,5700
5140 LN$ = LN$ + ":": NEXT : IF F = 1 THEN LN$ ~
     = LN$ + "NEXT"
5150 RETURN
5199 REM     CLEAR COMMAND
5200 LN$ = LN$ + "HGR": RETURN
5249 REM     TURNTO COMMAND
5250 LN$ = LN$ + "QA=90-" + GL$(L): RETURN
5299 REM     TURN COMMAND
5300 LN$ = LN$ + "QT=" + GL$(L) + ":GOSUB 54000
     ": RETURN
5349 REM     DRAW COMMAND
5350 LN$ = LN$ + "QL=" + GL$(L) + ":GOSUB 55000
     ": RETURN
5399 REM     PEN COMMAND
5400 T$ = GL$(L): IF T$ = "UP" THEN LN$ = LN$ +
     "QP=1": RETURN
5402 IF T$ = "DOWN" THEN LN$ = LN$ + "QP=0":RET
     URN
5405 LN$ = LN$ + "HCOLOR=": IF T$ = "ERASE" THE
     N LN$ = LN$ + "QB"
5410 IF T$ = "BLACK" THEN LN$ = LN$ + "0"
5415 IF T$ = "GREEN" THEN LN$ = LN$ + "1"
5420 IF T$ = "VIOLET" THEN LN$ = LN$ + "2"
5425 IF T$ = "WHITE" THEN LN$ = LN$ + "3"
5430 IF T$ = "BLACK2" THEN LN$ = LN$ + "4"
5435 IF T$ = "RED" THEN LN$ = LN$ + "5"
5440 IF T$ = "BLUE" THEN LN$ = LN$ + "6"
5445 IF T$ = WHITE2" THEN LN$ = LN$ + "7"
5448 RETURN
5449 REM      SCREEN COMMAND
5450 GOSUB 5400:LN$ = LN$ + ":HPLOT 0,0:CALL 62
     454:QB=" + RIGHT$(LN$,1): RETURN
5499 REM     GOTO COMMAND
5500 T = 0: FOR L1 = 1 TO LEN (GL$(L)): IF MID$
     (GL$(L),L1,1) = "," THEN T = L1
5501 L1 = 255
5510 NEXT :LN$ = LN$ + "QX=" + LEFT$ (GL$(L),T ~
     - 1) + "
5511 QY=" + RIGHT$ (GL$(L), LEN (GL$(L)) - T): ~
     RETURN
5549 REM      FULL COMMAND
5550 LN$ = LN$ + "POKE -16302,0": RETURN
5599 REM      MIX COMMAND
5600 LN$ = LN$ + "POKE-16301,0": RETURN
5649 REM      QUIT COMMAND
5650 LN$ = LN$ + "TEXT": RETURN
5699 REM      GO COMMAND
5700 LN$ = LN$ + "QP=1:QL=" + GL$(L) + ":GOSUB ~
     55000:QP=0": RETURN
5997 REM
5998 REM *** B: COMMAND ***

5999 REM
6000 LN$ = LN$ + R$: RETURN
6497 REM
6498 REM *** LABEL ***
6499 REM
6500 LN$ = LN$ + "REM" + R$: RETURN
9997 REM
9998 REM *** INITIALIZE ***
9999 REM
```

```
10000 TEXT : HOME
10010 HTAB 6: INVERSE : PRINT "                            "
10020 HTAB 6: PRINT " ";: HTAB 34: PRINT " "
10030 HTAB 6: PRINT " ";: HTAB 9: NORMAL : PRINT "TURTLE PILOT TRANSLATOR";
10031 HTAB 34: INVERSE : PRINT " "
10040 HTAB 6: PRINT " ";: HTAB 34: PRINT " "
10050 HTAB 6: PRINT " ";: HTAB 14: NORMAL : PRINT "BY ALAN POOLE";
10051 HTAB 34: INVERSE : PRINT " "
10060 HTAB 6: PRINT " ";: HTAB 34: PRINT " "
10070 HTAB 6: PRINT "                            ": NORMAL
10080 DIM P$(2500),I$(12),G$(11),GL$(6),M$(25):D$ = CHR$ (4)
10090 PRINT : INPUT "WHAT IS THE NAME OF THE PROGRAM? ";N$: IF N$ =""THEN 10090
10100 IF RIGHT$ (N$,2) < > ".P" THEN N$ = N$ + ".P"
10105 PRINT : PRINT "PLEASE WAIT..."
10110 ONERR GOTO 16000
10120 PRINT D$"VERIFY"N$: PRINT D$"OPEN"N$: PRINT D$"READ"N$
10130 INPUT NL
10140 FOR L = 1 TO NL:I$ = ""
10150 GET K$: IF ASC (K$) = 13 THEN P$(L) = I$: GOTO 10170
10160 I$ = I$ + K$: GOTO 10150
10170 NEXT : PRINT : PRINT D$"CLOSE"
10180 FOR L = 1 TO 12: READ I$(L): NEXT
10190 FOR L = 1 TO 11: READ G$(L): NEXT
10200 ONERR GOTO 15000
10210 RETURN
10220 DATA T,A,M,J,U,E,C,R,S,G,B,*
10230 DATA CLEAR,TURNTO,TURN,DRAW,PEN,SCREEN,GOTO,FULL,MIX,QUIT,GO
10997 REM
10998 REM *** REMOVE SPACES FROM T$ ***
10999 REM
11000 IF T$ = "" THEN RETURN
11010 T1$ = "": FOR L = 1 TO LEN (T$)
11020 IF MID$ (T$,L,1) < > " " THEN T1$ = T1$ + MID$ (T$,L,1)
11030 NEXT :T$ = T1$: RETURN
14997 REM
14998 REM *** ERROR ROUTINES ***
14999 REM
15000 PRINT D$"CLOSE": PRINT "ERROR IN PILOT LINE NO. ";LN; CHR$ (7)
15001 PRINT D$"RUN EDITOR"
16000 PRINT D$"CLOSE": PRINT "UNABLE TO LOAD"; CHR$ (7) : RUN
49996 REM
49997 REM THE FOLLOWING LINES ARE NOT PART OF THE TRANSLATOR, BUT ARE
49998 REM INCLUDED IN EVERY TRANSLATED PROGRAM.
49999 REM
50000 DIM Q$(25),QS(31)
50010 HCOLOR= 3:QX = 0:QY = 0:QC = - 1:QR = 40:QA = 90:QQ = 3.1415927 / 180
50020 QS(1) = 192:QS(2) = 180: QS(3) = 171:QS(4) = 161:QS(5) = 153:QS(6) = 144
50021 QS(7) = 136:QS(8) = 129:QS(9) = 122:QS(10) = 115:QS(11) = 108:QS(12) =102
50022 QS(13) = 96:QS(14) = 91:QS(15) = 86
50025 QS(16) = 81:QS(17) = 76:QS(18) = 72:QS(19) = 68:QS(20) = 64:QS(21) = 60
50026 QS(22) = 57:QS(23) = 54:QS(24) = 50:QS(25) = 47:QS(26) = 45:QS(27) = 42
50027 QS(28) = 40:QS(29) = 37:QS(30) = 35:QS(31) = 33
50030 POKE 770,173: POKE 771,48: POKE 772,192: POKE 773,136: POKE 774,208
50031 POKE 775,5: POKE 776,206: POKE 777,1: POKE 778,3: POKE 779,240
50032 POKE 780,9: POKE 781,202: POKE 782,208: POKE 783,245: POKE 784,174
50040 POKE 785,0: POKE 786,3: POKE 787,76: POKE 788,2: POKE 789,3: POKE 790,96
50041 POKE 791,0: POKE 792,0
50050 RETURN
51000 IF QT$ = "" THEN PRINT : RETURN
51005 QT = 0: IF RIGHT$ (QT$,1) = "&" THEN QT$ = LEFT$(QT$, LEN (QT$) - 1):QT=1
51010 FOR Q1 = 1 TO LEN (QT$)
51011 IF MID$ (QT$,Q1,1) = " " AND PEEK (36) > QR - 9 THEN GOSUB 51100
51020 PRINT MID$ (QT$;Q1,1);: NEXT : IF QT = 0 THEN PRINT
51030 RETURN
51100 QF = 0: FOR Q2 = Q1 + 1 TO Q1 + QR - PEEK (36) - 1
51101 IF Q2 > = LEN (QT$) THEN Q2 = 1000:QF = 1: GOTO 51120
51110 IF MID$ (QT$,Q2,1) = " " THEN Q2 = 1000:QF = 1
51120 NEXT : IF QF = :0 THEN PRINT :Q1 = Q1 + 1
51130 RETURN
52000 QI$ = ""
52010 GET QK$:QT = ASC (QK$): PRINT QK$;
52020 IF QT = 13 THEN RETURN
52030 IF QT = 8 THEN 52100
52040 IF QT = 21 THEN 52200
52050 IF QT = 24 THEN PRINT CHR$ (92): GOTO 52000
52060 QI$ = QI$ + QK$: IF LEN (QI$) > 245 THEN PRINT CHR$ (7);
52070 IF LEN (QI$) > 250 THEN PRINT CHR$ (92): GOTO 52000
52080 GOTO 52010
52100 IF QI$ = "" THEN PRINT : GOTO 52010
52110 IF LEN (QI$) = 1 THEN 52000
52120 QI$ = LEFT$ (QI$, LEN (QI$) -1): GOTO 52010
52200 QI$ = QI$ + CHR$ (PEEK ( PEEK (40) + PEEK (41) * 256 + PEEK (36)) - 128)
52210 POKE 36, PEEK (36) + 1: IF PEEK (36) > QR - 1 THEN PRINT
52220 GOTO 52010
53000 QM = 0:QC = 0: FOR Q1 = 1 TO 25
53001 IF LEN (QI$) < LN (Q$(Q1)) OR Q$ (Q1) = "" THEN 53030
53010 FOR Q2 = 1 TO LEN (QI$) - LEN (Q$(Q1)) + 1
53011 IF Q$(Q1) = MID$ (QI$,Q2, LEN (Q$(Q1))) THEN QC = 1:QM = Q1:Q1=25:Q2 =300
53020 NEXT
53030 NEXT : FOR Q1 = 1 TO 25:Q$(Q1) = "": NEXT : RETURN
54000 QA = QA - QT: IF QA > 360 THEN QA = QA - 360
54010 IF QA < 0 THEN QA = QA + 360
54020 RETURN
55000 IF QP = 1 THEN 55005
55003 HPLOT QX + 139.0005, - QY + 80.0005
55005 QX = QX + QL * COS (QA * QQ):QY = QY + QL * SIN (QA * QQ)
55010 IF QX < - 139 THEN QX = - 139
55020 IF QX > 140 THEN QX = 140
55030 IF QY < - 111 THEN QY = - 111
55040 IF QY > 80 THEN QY = 80
55050 IF QP = 1 THEN RETURN
55060 HPLOT TO QX + 139.0005, - QY + 80.0005: RETURN
```

Program 2.

```
1 REM TURTLE PILOT EDITOR
2 REM BY ALAN POOLE
```

```
10 GOSUB 20000
17 REM
18 REM *** MAIN LOOP ***
19 REM
20 PRINT : PRINT "<";: GOSUB 100: GOSUB 12000
   : IF I$ = "" THEN 20
25 IF LEFT$ (I$,3) = "CAT" THEN PRINT D$"CATA
    LOG": GOTO 20
28 IF LEFT$ (I$,3) = "PR#" THEN PRINT D$"PR#"
    VAL ( RIGHT$ (I$,1)): GOTO 20
30 GOSUB 500: IF C = 0 THEN PRINT SE$: GOTO 2
   0
40 ON C GOSUB 1000,2000,3000,4000,5000,6000,7
   000,8000,9000
50 GOTO 20
97 REM
98 REM *** INPUT ***
99 REM
100 I$ = ""

110 GET K$:K = ASC (K$): PRINT K$;
120 IF K = 13 THEN RETURN : REM RETURN KEY
130 IF K = 8 THEN 200: REM BACKSPACE
140 IF K = 21 THEN 300: REM RETYPE
150 IF K = 24 THEN PRINT CHR$ (92): GOTO 100: ~
    REM CTRL-X
160 IF K = 27 THEN 15000: RM ESC
170 I$ = I$ + K$: IF LEN (I$) > 195 THEN PRINT
     BELL$;
180 IF LEN (I$) > 200 THEN PRINT CHR$ (92): GO
    TO 100
190 GOTO 110
197 REM
198 REM *** BACKSPACE ***
200 IF I$ = "" THEN PRINT : GOTO 110
210 IF LEN (I$) = 1 THEN 100
220 I$ = LEFT$ (I$, LEN (I$) - 1): GOTO 110
297 REM
298 REM *** RETYPE ***
299 REM
300 I$ = I$ + CHR$ ( PEEK ( PEEK (40) + PEEK (
    41) * 256 + PEEK (36)) - 128)
310 POKE 36, PEEK (36) + 1: IF PEEK (36) > 39 ~
    THEN PRINT
320 GOTO 110
497 REM
498 REM *** DETRMINE NO. OF EDITOR COMMAND ***

499 REM
500 C = 0: FOR L = 1 TO 9: IF C$(L) = LEFT$ (I
    $, LEN (C$(L))) THEN C = L
510 NEXT : IF C = 0 THEN RETURN
519 REM MAKE I$ THE PART RIGHT OF COMMAND
520 IF I$ = C$(C) THEN I$ = "": RETURN
530 I$ = RIGHT$ (I$, LEN (I$) - LEN (C$(C))): ~
    GOSUB 12000: RETURN
997 REM
998 REM *** ADD ***
999 REM
1000 IF I$ = "" THEN 1030
1010 IF VAL (I$) > LN OR VAL (I$) < 1 THEN PRIN
     T RE$: RETURN
1020 LN = VAL (I$)
1030 IF LN > 2499 THEN PRINT "TOO MANY LINES";B
     ELL$: RETURN
1040 LT = LN + 1: GOSUB 10000:LN = LT:P$(LN) = ~
     I$: GOTO 1030
1997 REM
1998 REM *** LIST ***
1999 REM
2000 IF LN = 0 THEN PRINT : RETURN
2010 IF LEFT$ (I$,1) = "*" THEN GOSUB 13000: GO
     TO 2050
2020 IF I$ = "" THEN FL = 1:LL = LN: GOTO 2050
2030 GOSUB 11000
2040 IF FL < 1 OR FL > LN OR LL < 1 OR LL > LN ~
     THEN PRINT RE$: RETURN
2050 FOR L = FL TO LL: HTAB 5 - LEN (STR$ (L)):
      PRINT L;" ";
2051 IF LEFT$ (P$(L),1) = "*" THEN 2057
2055 PRINT " ";: IF LEFT$ (P$(L),1) < > "R" THE
     N PRINT " ":
2057 PRINT P$(L)
2060 IF PEEK ( - 16384) = 141 THEN L = 2500
2070 NEXT : RETURN
2997 REM
2998 REM *** EDIT ***
2999 REM
3000 IF I$ = "" THEN PRINT SE$: RETURN
3010 GOSUB 11000
3020 IF FL < 1 OR FL > LN OR LL < 1 OR LL > LN ~
     THEN PRINT RE$: RETURN
3030 HOME : FOR L1 = FL TO LL: VTAB 18: HTAB 5 ~
     - LEN ( STR$ (L1))
3031 PRINT L1;" ";P$(L1);: VTAB 18: HTAB 6: GOS
     UB 100: GOSUB 10010
3040 VTAB 24: FOR L2 = 1 TO LEN (P$(L1)) / 40 +
      2: PRINT : NEXT :P$(L1) = I$
3050 NEXT : RETURN
3997 REM
3998 REM *** INSERT ***
3999 REM
4000 LT = VAL (I$): IF LT < 1 OR LT> LN THEN PR
     INT RE$: RETURN
4010 IF LN > 2499 THEN PRINT "TOO MANY LINES";B
     ELL$: RETURN
4020 GOSUB 10000
4030 FOR L = LN + 1 TO LT STEP - 1: P$(L) = P$(
     L - 1):·NEXT
4040 P$(LT) = I$: LN = LN + 1: RETURN
4997 REM
4998 REM *** DELETE ***
4999 REM
5000 IF I$ = "" THEN PRINT SE$: RETURN
5010 IF LEFT$ (I$,1) = "*" THEN GOSUB 13000: GO
     TO 5030
5020 GOSUB 11000: IF FL < 1 OR FL > LN OR LL<1 ~
     OR LL> LN THEN PRINT RE$: RETURN
5030 FOR L = FL TO LN - (LL - FL + 1):P$(L) = P
     $(L + (LL - FL + 1)): NEXT
5031 LN = LN - (LL - FL + 1)
5040 RETURN
5997 REM
5998 REM *** NEW ***
5999 REM
6000 FOR L = 1 TO LN:P$(L) = "": NEXT :LN = 0: ~
     RETURN
6997 REM
6998 REM *** LOAD ***
6999 REM
7000 N$ = I$: IF N$ = "" THEN PRINT SE$: RETURN

7010 ONERR GOTO 21000
7020 IF RIGHT$ (N$,2) < > ".P" THEN N$ = N$ + "
     .P"
7030 PRINT D$"VERIFY"N$: PRINT D$"OPEN"N$: PRIN
     T D$"READ"N$
7040 INPUT T
7050 FOR L = 1 TO T
7052 GET T$: IF T$ = CHR$ (13) THEN 7058
7055 P$(LN + L) = P$(LN + L) + T$: GOTO 7052
7058 NEXT : PRINT
7060 PRINT D$"CLOSE"
7070 POKE 216,0: REM RESET ONERR
7080 LN = LN + T: RETURN
7997 REM
7998 REM *** SAVE ***
7999 REM
8000 IF I$ = "" THEN PRINT SE$: RETURN
8003 ONERR GOTO 22000
8005 IF RIGHT$ (I$,2) < > ".P" THEN I$ = I$ + "
     .P"
8010 PRINT D$"OPEN"I$: PRINT D$"WRITE"I$
8030 PRINT LN
8040 FOR L = 1 TO LN: PRINT P$(L): NEXT
8050 PRINT D$"CLOSE"I$
8060 POKE 216,0: RETURN
```

```
8997 REM
8998 REM *** MEM ***
8999 REM
9000 PRINT "THERE ARE "; FRE (0) - 3500;" BYTES
     LEFT.": RETURN
9997 REM
9998 REM *** INPUT PROGRAM LINE ***
9999 REM
10000 HTAB 5 - LEN ( STR$ (LT)): PRINT LT;" ";:
     GOSUB 100: GOSUB 12000
10010 IF I$ = "" THEN POP : RETURN
10020 F = 0:P = 0: FOR L = 1 TO LEN (I$):T$ = MI
     D$ (I$,L,1): IF T$ = ":"THENF=1
10030 IF T$ = "(" THEN P = P + 1
10040 IF T$ = ")" THEN P = P - 1
10050 NEXT:IF (F = 0 AND LEFT$ (I$,1) < > "*")OR
      P< > 0THEN PRINT SE$:GOTO10000
10060 F = 0: FOR L = 1 TO 12: IF IN$(L) = LEFT$
     (I$,1) THEN F = 1:L = 12
10070 NEXT : IF F = 0 THEN PRINT SE$: GOTO 10000

10080 RETURN
10997 REM
10998 REM *** FIND TWO NOS. DIVIDED BY COMMA ***

10999 REM
11000 FL = VAL (I$):LL = 0: FOR L = 1 TO LEN (I$
     ) - 1
11001 IF MID$ (I$,L,1) = "," THEN I$ = RIGHT$ (I
     $, LEN (I$) - L):LL = VAL (I$)
11010 NEXT : IF LL = 0 THEN LL = FL
11997 REM
11998 REM *** REMOVE LEADING SPACES ***
11999 REM
12000 IF I$ = "" THEN RETURN
12010 IF LEFT$ (I$,1) < > " " THEN RETURN
12020 IF I$ = " " THEN I$ = "": RETURN
12030 I$ = RIGHT$ (I$, LEN (I$) - 1): GOTO 12000

12997 REM
12998 REM *** FIND FIRST AND LAST LINES OF A MOD
     ULE ***
12999 REM
13000 FOR L = 1 TO LN: IF P$(L) = I$ THEN FL = L
     : GOTO 13020
13010 NEXT : PRINT "LABEL NOT FOUND";BELL$: POP
     : RETURN
13020 LL = 0: FOR L = L TO LN: IF LEFT$ (P$(L),1
     ) = "E" THEN LL = L:L = 2500
13030 NEXT : IF LL = 0 THEN LL = LN
13040 RETURN
14997 REM
14998 REM *** END PROGRAM ***
14999 REM
15000 PRINT : PRINT "DO YOU WANT TO SAVE THE PRO
     GRAM? (Y/N)";: GET I$
15001 IF I$ = CHR$ (13) THEN PRINT : GOTO 100
15010 IF I$ < > "Y" AND I$ < > "N" THEN 15000
15015 PRINT I$
15020 IF I$ = "Y" THEN INPUT "WHAT IS THE NAME O
     F THE PROGRAM? ";I$: GOSUB 8000
15030 PRINT : PRINT "DO YOU WANT TO TRANSLATE A"
     : PRINT "PROGRAM?(Y/N)";:GETI$
15031 IF I$ < > "Y" AND $ < > "N" THEN 15030
15040 PRINT I$: IF I$ = "Y" THEN PRINT CHR$ (4)"
     RUN TRANSLATR"
15050 HOME : END
19997 REM
19998 REM *** INITIALIZE ***
19999 REM
20000 TEXT : HOME
20010 HTAB 8: INVERSE : PRINT "
           "
20020 HTAB 8: PRINT " ";: HTAB 32: PRINT " "
20030 HTAB 8: PRINT " ";: HTAB 11: NORMAL : PRIN
     T "TURTLE PILOT EDITOR";
20031 HTAB 32: INVERSE : PRINT " "
20040 HTAB 8: PRINT " ";: HTAB 32: PRINT " "
20050 HTAB 8: PRINT " ";: HTAB 13: NORMAL : PRIN
     T "BY ALAN POOLE";: HTAB 32
20051 INVERSE : PRINT " "
20060 HTAB 8: PRINT " ";: HTAB 32: PRINT " "
20070 HTAB 8: PRINT "                        ":
     NORMAL
20080 DIM P$(2500),C$(9),IN$(12)
20090 BELL$ = CHR$ (7):D$ = CHR$ (4):SE$ = "SYNT
     AX ERROR" + BELL$
20091 RE$ = "LINE NO. OUT OF RANGE" + BELL$
20100 FOR L = 1 TO 9: READ C$(L): NEXT
20110 FOR L = 1 TO 12: READ IN$(L): NEXT
20120 RETURN
20130 DATA ADD,LIST,EDIT,INSERT,DEL,NEW,LOAD,SAV
     E,MEM
20140 DATA T,A,M,J,U,E,C,R,S,G,B,*
20997 REM
20998 REM *** ERROR ROUTINES ***
20999 REM
21000 POKE 216,0: PRINT D$"CLOSE": PRINT "UNABLE
      TO LOAD";BELL$: GOTO 20
22000 POKE 216,0: PRINT D$"CLOSE": PRINT "DOS ER
     ROR";BELL$: GOTO 20
```

©

A method for achieving statistical accuracy on microcomputer systems. These programs are written in a version of BASIC so general that they will work as is on Atari, PET, or nearly any other version of BASIC.

The Statistics Page

Accurate Statistical Calculations

Dr. Allen H. Wolach
Department of Psychology
Illinois Institute of Technology

Most statistical programs use formulas that are called computational formulas. Consider the computational formula $\Sigma x_i^2 - N\bar{X}^2$ which is used to calculate the sum of squares term for many statistics. The portion of the formula that is designated Σx_i^2 is obtained by squaring individual scores, and then summing the squared scores. The portion of the formula that is designated $N\bar{X}^2$ is obtained by multiplying the number of scores (N) by the square of the mean of the scores ($\bar{X}^2$).

If Σx_i^2 is equal to 11178323747 and $N\bar{X}^2$ is equal to 11178304256.25, the sum of squares is 11178323747 minus 11178304256.25 or 19490.75. The sum of squares of 19490.75 is well within the nine digit capacity of most versions of BASIC in microcomputer systems. However, the intermediate calculations exceed the nine digit capacity of most microcomputer systems. If the microcomputer resorts to exponential notation, Σx_i^2 becomes 111783237 x 10^2 and $N\bar{X}^2$ becomes 111783042 x 10^2. When the sum of squares is calculated $\Sigma x_i^2 - N\bar{X}^2$ becomes 19500. Note that the actual value for the sum of squares is 19490.75, not 19500.

Difference formulas can be used instead of computational formulas The difference formula for the sum of squares is $\Sigma(x_i - \bar{X})^2$. The difference formula requires subtracting the mean of all scores from an individual score and then squaring this difference. The squared differences for each of the scores are then summed. Difference formulas never produce intermediate results that are larger than the final value for the sum of squares.

The computational formula for the sum of squares will produce the same results as the difference formula provided that computations for the computational formula are carried to enough digits to accommodate intermediate calculations. On the other hand, the difference formula requires using the mean with each difference that is computed. In order to obtain the mean, one has two partially unsatisfactory options. All of the scores can be saved in memory. Then the mean can be calculated for use in each difference. Finally, the scores can be recalled individually from memory, and the mean can be subtracted from each score.

If the data for a given analysis consist of a relatively large number of scores, one can easily exceed the memory capability of a microcomputer system. A second option is to enter all of the scores twice. A mean can be calculated the first time the scores are entered. This mean can be used for forming the differences the second time the scores are entered. Although this procedure does not require saving the scores in memory, it doubles the number of scores that must be entered.

A procedure that is intermediate to the computational formula and difference formula procedures can be used. This approach can be called the provisional mean procedure. The procedure requires entering the data only one time and does not require saving individual scores.

In addition, the provisional mean procedure does not produce the large numbers that are generated by the computational formula. The provisional mean procedure involves calculating a difference for each score. This difference is not the difference between the score and the mean. It is the difference between the score and a provisional mean. The provisional mean changes as the successive scores are entered. Program 1 shows the use of the provisional mean procedure to calculate the mean, sum of squares, variance, and standard deviation for a set of data. A difference is calculated in statement 60. The mean and sum of squares are calculated in statements 70 and 80, respectively.

Program 1.

```
10 PRINT "ENTER NUMBER OF SCORES";
20 INPUT N
30 FOR I = 1 TO N
40 PRINT "ENTER SCORE";I
50 INPUT X
60 LET D = X - M
70 LET M = M + D/I
80 LET S = S + D*(X - M)
90 NEXT I
100 PRINT "MEAN = ";M
110 PRINT "SUM OF THE SQUARES = ";S
120 PRINT "VARIANCE = ";S/(N -1)
```

```
130 PRINT "STANDARD DEVIATION = ";SQR(S/(N ~
   -1 ))
140 END
```

Suppose that one wanted to calculate a Pearson product moment correlation coefficient. This correlation coefficient requires a sum of products term. The sum of products term can be calculated using modified provisional mean formulas. Program 2 is a BASIC program for calculating the Pearson product moment correlation coefficient.

Program 2.

```
10 PRINT "ENTER NUMBER OF PAIRS OF SCORES ~
   ";
20 INPUT N
30 FOR I = 1 TO N
40 PRINT "ENTER SCORES IN PAIR ";I
50 INPUT X, Y
60 LET D1 = X - M1
70 LET M1 = M1 + D1/I
80 LET S1 = S1 + D1*(X - M1)
90 LET D2 = Y - M2
100 LET M2 = M2 + D2/I
110 LET S2 = S2 + D2*(Y - 2)
120 LET D3 = SQR(ABS(D1))*SQR(ABS(D2))
130 LET D4 = (X - M1)*(Y - 2)
140 IF D4 < 0 THEN 170
150 LET G = 1
160 GOTO 180
170 LET G = - 1
180 LET S3 = S3 + G*D3*SQR(ABS(D4))
190 NEXT I
200 LET R = S3/(N*(SQR(S1/N))*SQR(S2/N))
210 PRINT "CORRELATION COEFFICIENT = ";R
220 END
```

Note that the mean of the X scores is calculated in statement 70, and the sum of squares for the X scores is calculated in statement 80. The mean for the Y scores is calculated in statement 100, and the sum of squares for the Y scores is calculated in statement 110. The sum of products term is calculated in statement 180.

Once sum of square and sum of products terms are mastered, accurate t-test and analysis of variance programs can be written for microcomputer systems. ©

A Monthly Feature

Learning With Computers

Glenn Kleiman
Teaching Tools: Microcomputer Services
Palo Alto, CA

As another school year begins, the "old-timers" among users of personal computers can look back and remember the early days, fondly recalling their 8K PET computers with small keyboards and built-in cassette recorders, TRS-80 Model I computers with 4K of RAM memory, and early Apple II computers with cassette recorders and Integer BASIC. Those who were among the first to use personal computers in schools may be celebrating their fifth year of computers in their classrooms.

Personal computers in education have come a long way in five years. Floppy disk drives, dot matrix printers and color monitors are now common in many schools. Hard disks, network systems, letter quality printers and modems are found in some. Improved versions of BASIC have been developed and LOGO, PILOT, Pascal, FORTRAN, FORTH and other languages are available for some personal computers. Even the manuals have improved tremendously. How many of you recall the first manuals (in some cases, better called pamphlets) that provided all the available information about the early personal computers?

Most importantly, significant advances have been made in educational software. Gone are the days when any program that did not crash was considered acceptable. Educational software is now expected to be user friendly, to make good use of the interactive, graphics and sound capabilities of the computer, and to follow principles of effective pedagogy. Educators are beginning to expect not just single programs, but courseware packages – sets of programs combined with written aids for teachers and students.

In addition to computers' becoming a widespread tool for teaching and learning, computer science is joining chemistry, biology and physics as a standard part of the curriculum in many schools.

It has been a remarkable five years since the first completely assembled personal computers became available. Exciting advances continue to be made – video disks interfaced to computers, computer speech synthesis and speech recognition, computer aids for handicapped individuals, more powerful computer systems at lower prices, advances in computer graphics and music, color printers, more extensive information and communication systems, new languages and more.

Of course, it hasn't all been smooth sailing. All of us who ventured into the world of computers have experienced disappointing software, inadequate documentation, incompatible components, service and supply problems, and other difficulties. Some educators have run into difficulties using computers and, unable to get the support they needed from colleagues, computer dealers or others, became frustrated and gave up. There are classrooms in which a computer sits unused in the closet. But there are many more classrooms in which teachers and students are eager to learn about and use computers.

A Practical Guide To Computers In Education

A book designed to help teachers get started using computers has recently been published. It is called *Practical Guide to Computers in Education* and is written by Peter Colburn, Peter Kelman, Nancy Roberts, Thomas F. F. Snyder, Daniel H. Watt and Cheryl Weiner (Addison-Wesley, $9.95). A practical guide is much needed, and this one was produced by a group of experienced educators, computer users and writers.

The *Practical Guide to Computers in Education* contains eight chapters plus sections describing the different brands of computers, a list of suggested readings, an extensive list of resources for educators and a short glossary. It is a very practical book, containing descriptions of experiences of many educators who are already using computers.

Chapter 1 is entitled "The Computer Goes to School." It raises the important question: "Will computers transform the schools?" The authors present both positive and negative arguments. The positive arguments point to the diverse capabilities of computers and their currently successful use in schools. The negative arguments focus on the problems that some have experienced with computers and on the fact that many educators do not have sufficient training to make good use of computers. Those who take the negative position typically view computers as the latest educational fad, likely to soon go the way of the New Math.

The authors of *Practical Guide to Computers in Education* try to present a balanced perspective. They don't believe computers can replace teachers or make schools obsolete, but they are certain that the widespread use of computers in society will affect schools. They have mixed feelings about

Recreational Computing Back Issues

Recreational Computing was the first and only personal computing magazine when it started in 1972 (it was called the *PCC Newspaper* back then). Bob Albrecht, David Thornburg, Isaac Asimov, Don Inman, Ramon Zamora, Robert Jastrow, Mac Oglesby, Adam Osborne – the list of authors reads like a Who's Who of microcomputing. These and many other authors contributed some of the finest articles about computers and now-classic games to the pages of *Recreational Computing*.

Last fall, *Recreational Computing* was merged into **COMPUTE!** and we are now offering available back issues. Whatever your interest, you'll find something here – from Spanish BASIC to Computers in Sports Medicine, from Future Fantasy Games to Robot Pets.

September 1974 A Practical, Low-cost Home/School Microprocessor System, The Computer Illiteracy Problem, Eight Games In BASIC

March 1975 Build Your Own BASIC, The Computer In Art, Biorhythms

March/April 1976 A TTY Game Games With The Pocket Calculator Dodgem Square, Tiny BASIC To Go

July 1976 BASIC Music, Tiny Trek For Altair, 16 Bit Computer Kit, Musical Numbers Guessing Game, Programmer's Toolbox

September/October 1976 Computer Games In The Classroom, Planets Game, Dungeons And Dragons, Hats Game, Pythagoras And Rational Music

November/December 1976 Story, Snake, Packl, Frogs Games, Make Believe Computers, The First West Coast Computer Faire, Subroutines, The First Computer

January/February 1977 Robot Pets, Computers And Space, Tiny Languages, Teaching Using Conversational Programming, High School Computers, Reverse, Tiny PILOT Mastermind

March/April 1977 Z-80 PILOT 6502 Assembly Programming, Tiny BASIC For Beginners, Math Drills & Games, Community Information Systems, Mine, Sales Simulation, Native American Board Games

July/August 1977 Do-it-yourself CAI, Pet Robots New Capabilities, PILOT, CAI In BASIC Programming The HP-25, Capture, Inverse Reverse, 8080 Matrix Subroutines, Women And Computers

September/October 1977 The $595 PET, More Tiny Languages, Computer Networks, The Bead Game, Biofeedback And Microcomputers Part 1, Home Energy Management, Sandpile Game, A BASIC PILOT

January/February 1978 Pascal Vs BASIC, COMAL Structured BASIC, Video Disks, Magic Lamps for Educators?, A Computer Revolution?, Pounce, The Mechanics of Robots, TRS-80 A Status Report

March/April 1978 Epic Computer Games, Micros for the Handicapped, Buckets Game, Prayer Wheel Program, Computer Contagion, Measuring Time, Frog Race, The IBM 370 Model 69

July/August 1978 Computer Whiz Kids Public Access To Computers, Man-made Minds, Post-human Intelligence, A Modern-day Medicine Show, Live Wire Design, ASCII Graphix, Baseball, Concentration, Gambler's Paradox

September/October 1978 Kingdom Game, Computers and Museums Sorcerer of Exidy, Snooping With Your PET, APL, Decimals in Tiny BASIC, Apple Math, TRS-80 Level II A Grown-up Field Evaluation

November/December 1978 APL Games, The Return of the Dragons, Animated Games for TRS-80, Runequest, All In The Mind, The L-5 Society, Phantnum, Some Guidelines for Microcomputer Chess, Dataman

January/February 1979 A Jules Verne Fantasy, Artificial Intelligence, The Apple Corps is With Us, TRS-80 Personal Software Vending Machine Gets "Brain," Apple II I/O, The Memory Game, REINO Spanish Kingdom

March/April 1979 Calculator Comics, "Lord of the Rings," Chess Reconsidered, Database, Beastiary, Color Your Own Graphics, Universe, Easy POKEing with Applesoft BASIC, Air Raid, TRS-80 3-D Plots, Slot, Apple Rose

May/June 1979 PILOT for Apple II The Game of Life Gold Handicapping Hunt, BASIC vs. Pascal, Inspector Clew-so, Flash for SOL, Faster Jumble, Concept Sans Computer, A Beginner's Guide To FRP

July/August 1979 Summer Fun, Fooling Around With Your PET, Cryptarithms, Baseball, Newett Awl's Goat, Zork, A Computerized Fantasy Simulation Game, What Light on Yonder Panel Flashes, The Dedicated Word Processor, The FORTE Music Programming Language

September/October 1979 TRS-80 Outside Connection, The Architecture of Multi-Player Games, The Sounds of Texas Instruments, Dynamic Color Graphics on the New Atari, An Apple PILOT, Gandalf, Spanish BASIC, Designing Animal Games, APL Mastermind

November/December 1979 SHOGI Games For You To Program, Atari Sounds, Texas Instrument Graphics and Animation, Interrupt, Match Me, Calendar, Making Music on the PET, Tower of Hanoi, Bingo, Animal Games

January/February 1980 Computing and Holistic Health, TI Graphics and Animation Part 2, Games To Program, New Directions in Numerical Computing, An Extended BASIC "IF" Facility, Beating Computer Anxiety, Capture for PET 8080 Tic Tac Toe, Chainwalk, Programming Problems

March/April 1980 Special Games Issue: Recreation Apple II Hi-res Graphics, Delicious Functions, Galaxy II, Fairy Chess, Raging Robots, Program Instruction Builder, Data Retrieval An Introduction

May/June 1980 Introduction to Computer Music, CBBS Phone Numbers, 6502 Machine Language, The Electric Phone Book, Number Translation, Sea Search, Apple Animation, Twister Move Generator, DOZO, Shell Game, Home Video Displays, A Proposed Graphics Language

July/August 1980 Fantasy Games Issue: Write Your Own Computer Fantasy Simulation, Wizard's Castle, On Future Fantasy Games, Wonderful World of Eamon, In Defense of Hackers, Touch Panels and Interactive Graphics

September/October 1980 Probability Trees Big Business on the Micro, The Best of People's Computer Company, Computer Analysis of Athletics, Word Search, Computers in Sports Medicine, Wired, Revolution in Typography?, Textrapolation

November/December 1980 Computerized Voting, Computer-Using Educators, Hot-rod Computers, House of the Future, Yote, DOZO in Pascal, What is Truth? Sixth Order Magic Squares on a TRS-80

January/February 1981 The Education Revolution Computer Games in the Classroom, An Art-producing Turtle, Computer Literacy Resources, Musical Compositions Using Computers, Microcomputers in China, Twenty Questions, The Pirate's Life for Me, Computers and the Volcanic Fallout

March/April 1981 Space Exploration Frontiers for You and Your Micro, Voyage To Antares, A Spaceship Simulator, The Computer as Chess Ally, Star Trek – A Dialogue Approach Mark of Breeding (fiction), The Fifteen Puzzle

May/June 1981 Using Computers at Sesame Place Atari PILOT and Turtle Graphics, Computer Anatomy for Beginners, The Impact of Micros, Nevada-style 8-spot Keno, Sketch Pad, Sum of the Digits, TRS-80 Property Management Program, The Pocket Corner

July/August 1981 Which Computer Should You Buy?, Commodore's New Rainbow Machine, The Wired Nation Do We Want It?, Computers at the Junior Museum 3-D Tic Tac Toe for PET Number Crossword for all Computers

September/October 1981 43 Ways To Make Money With Your Micro, How To Start A Software Exchange, Who Are Computer Criminals?, Micros Behind Bars, Number Systems, Computer Knock-knock Jokes in BASIC and LISP, For Photographers Only, Fibonacci Nim, Roman Numeral Conversum Programum

Special *Recreational Computing* Back Issue Pricing:

Single Issue:	**$ 3.00**	**Any Fifteen Issues:**	**$25.00**
Any Five Issues:	**$10.00**	**Any Twenty Issues:**	**$30.00**
Any Ten Issues:	**$20.00**		

For Fastest Service, Call Toll Free **800-334-0868**
In NC Call **919-275-9809**

Or Send Order and Payment to **COMPUTE!** Publications, P.O. Box 5406, Greensboro, NC 27403.

In the US, please include $.20 per issue ordered for shipping and handling. Outside the US, please include $.30 per issue for surface mail. Orders must be prepaid in US funds or international money order. All orders subject to availability.

Krell's College Board SAT* Preparation Series

New for 82

A COMPREHENSIVE PREPARATION PACKAGE / 30 PROGRAMS / $299.95

1. Diagnostic analysis
2. Prescription of individual study plans
3. Coverage of all SAT* skills
4. Unlimited drill and practice
5. SAT* Exam Question simulator
6. All questions in SAT* format and at SAT* difficulty level
7. Instantaneous answers, explanations and scoring for problems
8. Worksheet generation and performance monitoring - (optional)
9. A complete record management system-(optional)
10. Systematic instruction in pertinent math, verbal & test taking skills - (optional)

Krell's unique logical design provides personalized instruction for each student according to individual needs.

APPLE, ATARI, COMMODORE, CBM/PET, CP/M, I.B.M., RADIO SHACK TRS-80.

LOGO $99

NO FRILLS PAK

1. Two copies of Krells LOGO for Apple II*
2. Utility Disk with M.I.T.'s valuable demo programs including Dynatrack
3. The official M.I.T technical manual LOGO FOR APPLE II by H. Abelson & L. Klotz

No Frills Turtle Price $99.00

FRILLS FOR LOGO

FRILLS FOR LOGO / Support Pak for: M.I.T. LOGO, KRELL & TERRAPIN INC.

1. Krell utility disk
2. Alice in LOGOLAND
3. LOGO for Apple II - by H. Abelson
4. Alice in Logoland Primer
5. Comprehensive wall chart
6. LOGO & Educational Computing Journal

$89.95

NO FRILLS LOGO AND ALL THE FRILLS COMBO FOR APPLE II* $149.95

ALSO AVAILABLE

Time Traveler / Odyssey in Time
Competency/Proficiency Series
Pythagoras and the Dragon
Isaac and F. G. Newton / Micro Deutsch
Super Star Baseball / Sword of Zedek
Krell Game Pak

CALL FOR DETAILS AND PRICES

VISA

KRELL® SOFTWARE CORP.

The state of the art in educational computing

1320 Stony Brook Road / Stony Brook NY 11790
Telephone 516-751-5139

Krell Software Corp. has no official ties with the College Entrance Examination Board or the Educational Testing Service Krell is, however, a supplier of products to the E T S

*Trademarks of Apple Comp Corp Tandy Corp Commodore Corp Digital Research Corp., I B M Atari Corp., Terrapin Inc

PROGRAMS AVAILABLE FOR THE TRS-80 APPLE II PET & ATARI

N.Y.S residents add sales tax

whether the effects will be good or bad. When feeling optimistic, they expect (among other effects) that computers will promote active, creative and individualized learning, and provide an antidote to many of the ill effects of television. When feeling pessimistic, they fear that the prevalence of computers in society may result in "more resistance to learning and more truancy from schools which cannot provide the immediate excitement of computer games."

Subculture Of Educational Computing

I share the view that computers have the potential for both positive and negative effects upon schools. However, I strongly disagree with the authors when they write: "Unfortunately, there appears to be little we, as educators, can do about such computer fallout in the schools except to prepare ourselves and our students for the possibility of it occurring." I believe there is much to be done by all of us who are concerned with education. We can make good use of the educational potential of computers and prepare students for a more computerized society. We can develop high quality software or help others do so. We can use computers to help in mainstreaming handicapped students. The authors of *Practical Guide to Computers in Education* seem to assume that educators must play a passive role in determining the future of schools.

The rest of Chapter 1 introduces some key terms and presents "a brief tour of the subculture of educational computing." The tour consists of descriptions of real teachers and students using computers in various ways. These vignettes provide valuable insights into the possible benefits and agonies of using computers. Similar vignettes appear in other chapters and comprise one of the strong features of this book.

Chapter 2 covers the myriad ways computers can be used in education. It begins with Computer Assisted Instruction, which is subdivided into drill and practice, tutorials, demonstrations, simulations and instructional games. It then goes on to Instruction/Learning Tools, subdivided into word processing, numerical analysis (e.g., VisiCalc), data processing (including accessing information from data bases such as the Source), instrument monitoring devices (for using computers in science labs), graphics and sound. The next section discusses Computer Managed Instruction – systems designed to measure and keep track of student performance and automatically present a sequence of lessons. Then the merits of teaching programming and computer literacy are discussed. I would have liked to see more information about graphics and programming, but overall this is a solid chapter with a great deal of useful information.

Chapter 3 is on "Bits and Bytes." It introduces many of the key terms and describes the parts of a computer system. However, it does not attempt to convey much understanding of how computers work and how they can be programmed to perform such a variety of functions.

Chapter 4 covers "Choosing Your Computer System." The authors take an approach with which I agree completely. They advise you to start by deciding how you want to use computers and checking into the availability of suitable software. Only then can you select a computer system that will meet your needs. They discuss five categories of computer systems: hand-held computers, cassette based computers, floppy disk based computers, hard disk based computers and resource-sharing networks. I noticed only one practical point that was neglected. Much software is available only on floppy disks and is protected so it cannot be copied. This often creates problems for people with hard disk systems – they may be unable to transfer desired software from a floppy disk to their hard disk.

Chapter 5, "Choosing Educational Software," starts out warning about the lack of quality software. Although I have seen my share of atrocious software, I feel that the authors are more negative than is warranted. The software available now is far superior to that available just a year or two ago, and I expect it will continue to improve.

Once past the negative introduction, this is an excellent chapter. It presents a detailed set of guidelines for evaluating software. The guidelines cover program content, pedagogy, program operation and student outcomes. More specific questions to consider are given under each category and examples of existing programs are described. I noticed only one gap worth filling. The authors discuss the important area of what type of feedback the computer provides to the students. However, they limit their discussion to right/wrong and reinforcing feedback, neglecting to discuss the value of remedial feedback – feedback which helps the student understand and correct his mistakes, not just know whether he was right or wrong.

Chapters 6 and 7, "Introducing Computers into the School" and "Integrating Computers into the School," will be very useful to many educators. Sections cover acquiring computers, funding, preparing teachers and administrators, locating computers in the school (e.g., spread among classrooms or in one central room), and providing the basic information needed in a computer facility. These chapters are based upon the experiences of many educators and should be read by all who are introducing computers into schools.

The final chapter is entitled "Issues and

Choices in Educational Computing." It first gives a brief history of computers in education, going back to the 1960's when talk of a forthcoming computer revolution in education was first heard. The authors then raise important questions for the current period of computers in education. These include: Where will good software come from? How will teachers get the needed training? Is the available hardware adequate? What kinds of social and educational changes might result from computers in the schools? The chapter ends with descriptions of four possible futures: computers turn out to be another educational fad; education becomes more centralized and systematic through the use of computers; schools become irrelevant; and educators become comfortable with computers and make good use of them.

The book also contains a large section on resources for educators. This 48-page list was compiled by Newton Key of Intentional Educations, and it will be updated in the 1983 *Classroom Computer News Magazine* Directory of Educational Computing Resources. Among the listings included are software directories and catalogs, associations, periodicals, resource centers, user groups and other sources of information. This is a valuable resource list. However, it will become dated quickly and annual updates are essential.

I recommend *Practical Guide to Computers in Education* to every educator who is starting to use, or considering using, computers. The chapters on evaluating software and introducing computers into schools, the real-life vignettes throughout the book, and the resource list and bibliography are especially valuable.

However, I have one general criticism of this book (in addition to the quibbles discussed above). It is, in many sections, negative in tone and short-sighted. I suspect the authors are over-reacting to those who claim that computers will cure all that ails education. Certainly, computers are not a panacea and a practical guide should be honest about the difficulties one may encounter. But I believe a book of this sort should also convey an understanding and appreciation of the potential of computers, and it should encourage educators to explore the ways computers can be used.

The authors limit themselves to describing existing software, without acknowledging recent improvements or looking ahead to what will be available soon. Little is said about computer graphics or music, two areas in which some excellent innovative software is already available. They mention the Source, but fail to convey the excitement of having enormous amounts of information readily accessible from your classroom or home. Nothing is said about the potential of video disks, computerized speech synthesis or speech recognition. They also neglect the ways computers will be able to aid handicapped students, such as automatic readers for the blind.

Perhaps the authors tried to limit the book to what is immediately practical for educators. However, with the rapid pace of change, what was not practical when the book was written may well be practical by the time many educators read it.

Software Catalogs

There is a lot of educational software available, but it is often difficult to find what you need and even more difficult to know whether it is worth purchasing. Several companies have put together educational software catalogs which should be very helpful. All the companies listed below have reviewed the software in their catalogs and, better yet, will allow you to return software within 30 days of purchase – a risk-free way to check whether a particular program meets your needs. All of these catalogs include Apple, TRS-80 and PET software for grades K-12, as well as books and computer supplies. Some also include software for Atari and TI computers.

K-12 Micromedia. P.O. Box 17, Valley Cottage, NY 10989, 914-358-2582.

Scholastic Microcomputer Instructional Materials. Scholastic, Inc. 904 Sylvan Ave., Englewood Cliffs, NJ 07632, 800-631-1586.

J. L. Hammett Microcomputer Catalog. Hammett Place, Braintree, MA 02184, 800-225-5467.

Opportunities for Learning Catalog. 8950 Lurline Ave., Chatsworth, CA 91311, 213-341-2535.

Periodicals on Computers in Education

There are now many periodicals on computers in education. Each of the following contains a variety of articles, announcements about new hardware and software, software and book reviews, and other information of interest to educators. Most of these have been publishing a short time and continue to change and develop. Judging from the past year, I recommend *Electronic Learning* and *Classroom Computer News* most highly, but all of the following have been useful.

Classroom Computer News. P.O. Box 266, Cambridge, MA 02138, 617-923-8595.

Electronic Learning. Scholastic, Inc. 902 Sylvan Ave., Englewood Cliffs, NJ 07632.

Educational Computer Magazine. P.O. Box 535, Cupertino, CA 95015.

Electronic Education. Suite 220, 1311 Executive Center Drive, Tallahassee, FL 32301.

The Computing Teacher. Dept. of Computer and Information Science. University of Oregon, Eugene, Oregon 97403. ©

Book Review:

Understanding Computer Science

Louis F. Sander
Pittsburgh

I picked up *Understanding Computer Science* while waiting in line at Radio Shack, and I couldn't put it down. Having brought it home and read it from cover to cover, I'm convinced that it's a little-known treasure that belongs on every computer owner's bookshelf. This book is an inexpensive overview of all the major areas of computing, written at a level which any interested person can understand, and the information it contains is perfectly packaged for home computerists. It is an ideal guide to self-paced individual learning of computer subjects.

The book was designed to build understanding step-by-step, and it succeeds. The first chapter reviews the development of computer hardware and software from the abacus to the microcomputer, and provides a non-technical background on the state of solid state technology. It familiarizes the reader with elementary terms and concepts, and charts the place of the personal computer in the world of the 1980's. Like all the subsequent chapters, this one concludes with a multiple-choice self-evaluation quiz. The questions are straightforward and there is an answer key in the back of the book.

The World Beneath The Keyboard

The second chapter, "Computer Architecture and Hardware," builds on the foundation laid by the first, providing a demystifying tour-de-force of busses, logic gates, binary numbers, and the other structures of the world beneath the keyboard. The secrets of tape and disk recording are also revealed here, and all of this is done with a minimum of jargon and a maximum of clarity. This chapter alone is worth the price of the book, but there's lots more to follow!

The "Programming" chapter starts with simple definitions, continues with examples of flowcharting and program design, and ends with a discussion of coding, translating, and debugging.

The "Languages" chapter gives an overview of BASIC – examples of some of the statements, a sample program, rules for evaluating expressions, etc. There's nothing remarkable here for the **COMPUTE!** reader who's past the beginner stage, but it's a good review of what BASIC is all about. What *is* remarkable is the rest of the chapter. It gives us similar, clear overviews of PASCAL, FORTRAN, COBOL, and PL/1. Did you ever wonder what these languages look like? Here's an excellent place to find out.

There are other chapters on Operating Systems, Resource Management (in large systems), Data Structures, Language Translators, and Systems Analysis. All are well-written, all are informative and useful. A view of the future of computers, a four-page glossary, and an excellent index complete the book.

Understanding Computer Science is a unique and valuable one-stop source of general information on computers. In a field where detail can overwhelm the beginner (and almost always does), this book offers a clear and understandable view of the big picture. In a field where even the expert can get lost in details, this $3 volume is an excellent overview.

Understanding Computer Science, by Roger S. Walker. Developed and published by Texas Instruments Learning Center. Distributed by Radio Shack (#62-1383). 267 pages, $2.95. ©

Review:

Flex File: A PET Data Base Manager

Donald C. Johnson
Renton, WA

Recent involvement with automatic test equipment has introduced me to a programming technique referred to as *application programming*. Application programming is very user oriented in that the operator initially defines his/her program needs by answering a series of computer screen prompts. Based upon the answers provided, the main program writes a second program that is tailored specifically to the user's needs. This affords a user the opportunity to design or truly create a custom program. *Flex-File* is such an application program.

Immediately upon receiving a program, I generally load the disk and run it. I tend to leave the instruction manual for assistance during periods of confusion, or as required reading during my next bout with the flu. However, *Flex-File* documentation, consisting of 36, 8½ x 11 pages of operational instructions, is actually a series of mini-lessons to be used with the sample inventory file provided on the disk.

Features

Filed data is handled via a keyed random access method. This allows for any and, if desired, all data fields to be designated as key fields.

The record size is limited to 250 characters; however, the number of records you can generate is limited only by the amount of storage space available on your disk system. Defined field lengths may be varied from record to record. Using a standard mailing list format, 1,000 records can be handled. Almost three times as much data can be handled if you are fortunate enough to own an 8050 drive.

All operations are menu driven. General commands such as add, delete, change, transfer, key (change), and exit perform as expected. An interesting feature incorporated in the file maintenance routine is the ability to scroll forward or backwards through the file.

Two available commands not normally found on many data base programs are BROWSE and USER. Exercising the BROWSE command provides a quick review or comparison of any two fields of data, i.e., the primary keyed field and the selected browse field. This function allows a cursory review of selected data fields without commanding total record recall. From a programmer's (or hacker's) standpoint, the program is very friendly. Menu space has been provided for a user's routine. There are no protection securities which prevent modifications. In fact, user modifications are encouraged. A special section of the documentation provides important information to allow program changes if desired. Additionally, a complete list of program variables is provided.

The author has provided two very powerful operations that are normally optional modules with other data bases. A full function math/calculating capability and a field-selectable, integrated mailing list routine enhance the flexibility of the system. Any column of numeric data may be mathematically processed with other columns. As a result, you can print averages, sub-totals, totals, ratios, etc. In fact, you can manipulate numeric data with +, -, x, /, % and log/trig functions.

Another handy feature of *Flex-File* is its random access to sequential file conversion ability. This allows the user to transfer in or out formatted data to other programs. For example, *Flex-File* compiled data may be transferred to a word processor for form letter generation, or inventory data can generate re-order forms using your word processor.

Considering the tremendous flexibility of *Flex-File*, the documentation could have been expanded to include other examples of interfacing with word processors, program listings, or suggestions for various professional applications. This is really a small criticism of such a powerful program though.

Michael Riley has developed an excellent, professional data base program that contains many features – too many to cover in a short review. Suffice it to say that you can spend much more for a high quality data base, but it's doubtful you will find the flexibility of an application program such as *Flex-File*.

Flex-File
AB Computers
252 Bethlehem Pike
Colmar, PA 18915
$80.00

©

Reviews:

Moonbase Io And Space Ace For Atari

Tom R. Halfhill
Features Editor

Moonbase Io is an arcade-style space game – actually a combination of three games – inspired by the recent flights of Voyagers 1 and 2 to Jupiter. The "Io" (pronounced "eye-oh") in the game's title refers to one of the four major moons of the solar system's largest planet. Since Jupiter was discovered to be a huge ball of inhospitable liquids and gases, unsuitable for landings, this game uses the moons Io, Europa, and Ganymede as moonbases for your spacecraft.

This choice of bases, however, has upset the local bug-eyed populace. (Can you blame them? They were probably never even consulted for the environmental impact statement.) Determined to send you back where you came from, or destroy you in the process, the aliens launch swarms of ships to battle your probe.

Audio Effects

Moonbase Io is a one-player game available on disk or cassette and requires 16K RAM and joystick. It also requires fast coordination and is suitable for all but the youngest children.

One extraordinary feature of this game is its use of the Atari's capability to synchronize the screen with a soundtrack on cassette tape. Both the cassette and disk versions include a soundtrack tape that is snapped into the Atari Program Recorder when the game starts. The PLAY button is left on, and the program starts and stops the tape at appropriate moments. The first time you play *Moonbase Io*, you get a long briefing on your mission from your superiors on Earth. Meanwhile, the screen displays a control room, a busy robot, and a window through which moving stars are visible. After receiving all your instructions (including a warning that the aliens are suspected to be unfriendly), you hear a countdown and blast-off, complete with sound effects and Cape Canaveral background chatter.

Because this initial briefing takes several minutes, there's a shortened version on Side B of the cassette which starts at the countdown for subsequent games. At various points during the game, during breaks between the three levels, the tape comes on again with additional messages. And if you make it past all the levels, you even get a congratulatory speech from the President of the Earth Federation.

This is the first time I've seen a soundtrack with a game program, although they are often used with Atari educational software. It's a good concept, and a laudable attempt to take advantage of all the machine's capabilities. However, some players, caught up in the intensity of *Moonbase Io*, may be frustrated by the procedure necessary to restart everything after a sudden explosion prematurely ends the game.

Besides the soundtrack, programmer John Konopa has used other Atari features to good advantage, too. The machine language program is fast and incorporates player/missile graphics, redefined characters, fine scrolling both vertically and horizontally, and well-executed sound effects from the Atari's four voices.

Moonbase Io is really three games in one, with two distinct phases. To secure each of the three moonbases, the player must pass a docking phase – impeded by flocks of aliens – and, if successful, transport to the surface of each moon for a phase which involves defending the base against more formations of fast-moving enemies. In addition, there are seven skill levels, from Novice to Galactic Wizard, with variations in scoring and availability of reserve ships. Some "hidden features" are also promised, although I never managed to survive long enough to experience them. Overall, *Moonbase Io* is a challenging arcade-style space game.

Space Ace

Like *Moonbase Io*, *Space Ace* is another fast-action, arcade-style, space warfare game for one player. It comes on tape or disk, requires 16K, a joystick, and the reaction instincts of a pro hockey goalie. It's suitable for all but very young children and is aimed at the video game addicts among us.

Space Ace starts you off at the bottom of the screen with a spaceship maneuverable in all eight directions with the joystick. The movement is smooth – player/missile graphics is obviously used here – and pressing the trigger button fires up to two shots at a time.

Programmer Greg Young makes good use of fine vertical scrolling to move a star field down the screen, creating the illusion of forward travel. This travel is not unobstructed, however, since your path is blocked by increasing numbers of asteroids (redefined characters). Blasting a path through the rocks isn't too hard, at least at the outset, but a

number of other twists are thrown in as well.

First, there are the "space vortexes." These are whirling player/missile shapes which look something like runaway blades from an old-style push lawnmower. The vortexes drop down the screen at you – they seem to know where your ship is – and cannot be destroyed, only evaded. Dodging them can be hazardous, though, since in your haste you often collide with stray asteroids.

Then there are the "Silurian space bombs," small objects which traverse the top of the screen until crossing above your ship. These things definitely know where you are. With a terrifying shriek, they plunge downward faster than you can see. Unless you dodge at just the right moment, or get off a lucky shot, you've had it.

As if these hazards weren't enough, occasionally a "space-mine field" stretching all the way across the screen descends upon your craft. You either blast a path through the mines or get trapped against the bottom of your TV.

And finally, at odd intervals, a "master Silurian warship" crosses the top of the screen. This warship unleashes a hail of missiles with deadly accuracy and can only be destroyed by hitting its central "atomic drive core" – not a large target.

Despite all this activity, *Space Ace* is very fast and smooth and makes extensive use of the Atari graphics and sound effects. There are multiple difficulty levels, and you can choose to receive a bonus ship every 1,000 points. I recommend this option for survival, especially since the scrolling asteroid clusters seem to speed up at 1,000-point intervals. Also, the "Training" level is fairly easy and suitable for children.

There is no pause between the destruction of one of your ships and its replacement with another. This happens so quickly that sometimes the same asteroid cluster which knocked you off the first time gets your new ship as well. However, it is possible to freeze all action on the screen at any time by pressing any key – if you can remember to do it in the heat of combat.

Fans of arcade-style games should find *Space Ace* both well-executed and challenging.

Space Ace
London Software
374 Wildwood Avenue
Piedmont, CA 94611
$29.95 Disk/Tape

Moonbase Io
Program Design, Inc.
11 Idar Court
Greenwich, CT 06830
$29.95 Disk/Tape

©

Review:

VIC Invaders

Harvey B. Herman
Associate Editor

With three VIC versions of the venerable computer game "invaders," how can an invader aficionado (and there are still lots of them around) make the proper choice? It seemed to me that it would be helpful to have a summary of the characteristics of each of these three programs. The prospective owner then would have more of a chance to make an informed decision.

I do have a problem. The invaders seem to get me before any of the advanced features come into play. Luckily, a friend of my son is a super player and was able to give the programs a good test.

Here is his summary of the relevant features of each program:

	VICVADERS	ALIEN BLITZ	VIC AVENGER
Medium	tape	tape	cartridge
Colors	various	blue/white	various
Skill Levels	1	10	1
Number of Aliens	40	40	50
Points per Alien	5	10-30	10-30
Mystery Ship Points	10-80	100	50-300
Base/Ship Size	large	small	large
Bases	3	3	3
Points for Bonus	1000	1000	1500
Continuous Fire	no	no	no
Instructions	no	no/sample game	yes/sample game
Adjustable Display	no	no	yes
Graphics	good	good	good
Price	$9.95	$24.95	$29.95
Vendor	Skyles Electric	Tensor Technology	Commodore

©

This month we conclude our overview of one of Commodore's new computers with game programming examples and the latest news on the software and peripherals soon to be available for the Commodore 64.

Sprite Graphics And Sound Synthesis On The Commodore 64

Tom R. Halfhill
Features Editor

Sometimes those shiny new shoes you buy wind up feeling like concrete blocks after a few hours' wear. Or those fancy stereo headphones turn into a vise by the end of one album.

My first few hours with a prototype Commodore 64, however, led me to believe it will be a comfortable computer. The manual is coherent, the keyboard is friendly, the full-screen editing is fun. Now that manufacturers are not merely selling computers to experienced users, but are also pushing them as home appliances to the wary masses, nothing less will do anymore. Someone with no previous experience on Commodores (or new to computing entirely) will quickly feel at home on the 64.

Not that the Commodore 64 is easy to master, or that its *User's Guide* is the ultimate reference book. Quite the contrary. My first real contact with the 64 at Commodore International's headquarters near Philadelphia reveals it to be a computer of formidable complexity – and flexibility – that will keep magazine writers and aspiring book authors busy for a long time explaining its inner workings. But while mastery may be elusive, familiarity is not. It's quite possible, at a first sitting, to be manipulating sprites and shaping envelopes.

Color Facilities

Sprites? Envelopes? Relax. The Commodore 64's advanced features take a little explaining, but are easy to grasp.

Let's start at the beginning. First, the 64 should be available in small quantities at Commodore dealers by the time you're reading this; Commodore predicts production of 15,000 to 20,000 units a month by late 1982. The $595 computer comes in the same compact plastic case and has the same keyboard as the $299 VIC-20. The 64 plugs into any color (or black-and-white) television via its built-in RF modulator, and upon power-up tells you that 38,911 bytes of Random Access Memory (RAM) are available (52K for pure machine language programming).

The screen is light blue surrounded by a darker blue border, and characters appear as dark blue. The characters are a bit hard to see against the background, and most people change them to white by holding down the control key (marked "CTRL") and typing "1". You can change the characters to any of 16 colors this way with the CTRL and numeric keys. And by inserting color control numbers (given in the manual) into two memory locations with the BASIC command POKE, you can also change the screen border and background to any of these 16 hues. It's the cheapest way of getting a green-screen you'll ever see.

The manual, which doubles as an elementary self-teaching guide for BASIC programming, steps the first-time user through these kinds of basic concepts with readable explanations and easy-to-follow examples. Early on, it shows how to load and save programs with disks and cassettes, so people who are not interested in programming are spared wading through swamps of murky text.

The computer has other features in line with this philosophy, such as a two-key "warm start" which clears everything except the program in RAM, and supposedly makes the computer uncrashable. (I say "supposedly" because I managed to rise to the challenge by inadvertently – and irretrievably – crashing the 64 within an hour, mystifying the Commodore people present. They suggested it might be a bug in the prototype, but not production, models. Oh well, nobody's perfect.)

Anyway, for the programmers among us, rest assured the 64 will be a hacker's machine too. Impressive screen displays can be created with 16 simultaneous colors and the high-resolution 320- by 200-dot graphics mode. The screen editing ranks with the best, and nearly all BASIC keywords and commands can be abbreviated merely by shifting the second letter as on other Commodore products.

Sprite Graphics

But the real power of the Commodore 64 lies in its more esoteric features: sprite animation and sound synthesis.

"Sprites" also are known as "MOBs" (for Movable Object Blocks) and "player-missiles" (in Atari parlance). Basically, a sprite is a graphics block corresponding to a block of memory which the programmer can sculpt into any shape and move around on the screen. This movement can be extremely smooth and fast, and is independent of other sprites or of objects drawn on the screen the usual ways.

Sprites can be made to pass in front of or behind other sprites and screen shapes, simulating three dimensions. The computer also detects collisions between sprites and other objects, so programmers can trigger explosions or other effects (sprites are usually used in games). Until now, this very powerful feature was available only on Atari and Texas Instruments computers.

Sprites are relatively easy to program. Within an hour after sitting down with the manual's 11-page chapter on sprites, an experienced BASIC programmer can be drawing and manipulating them without much trouble. And yes, BASIC can move at least a few sprites fast enough to make home-grown arcade-style games practical in BASIC.

Different computers implement sprites in different ways. The Commodore's sprites are blocks 24 pixels (screen dots) wide by 21 pixels tall. They can be any of 16 colors, and up to three colors can be combined in one sprite. Up to eight sprites can appear at once on a TV's horizontal scan line; much larger numbers are possible with a "raster-scan interrupt," an advanced technique the manual avoids. The sprites can be doubled or quadrupled in both dimensions with a POKE statement.

POKEs, in fact, are required for just about everything involving sprites on the 64. BASIC has no special keywords for these functions. However, Commodore is promising a plug-in cartridge soon which will add extended commands for sprites, graphics, and sound.

In the meantime, start POKEing numbers of memory locations into your brain until you can rattle them off like a baseball fan quoting batting averages. One very important location is the starting address of the special chip which controls sprites on the Commodore, 53248. Many other addresses are calculated from this location, so one approach is to store 53248 in a variable and handle everything else as "offsets."

To draw a sprite on the screen, it must first be "drawn" in memory. This is done by storing numbers corresponding to the image in DATA statements, and then POKEing them into memory with a FOR-NEXT loop (just like storing a machine language routine from BASIC). To determine these image numbers, you mark off a 24 by 21 grid on graph paper (to represent the 24- by 21-pixel sprite) and color the squares to draw your image. We won't cover the details here, but a colored square is an "on" bit, a blank square an "off" bit. The binary numbers that result are converted to decimal numbers and stored in the DATA statements.

A few more things are required to display and move sprites on the 64. A memory location offset from 53248 is POKEd to turn on the sprite, a pointer is set to tell the computer where to find the image data, and two more offsets from 53248 are POKEd to change the sprite's vertical and horizontal coordinates. These coordinates, by the way, extend beyond the visible screen. These steps are necessary for each sprite.

Here's a short example program, adapted from the *Commodore 64 User's Guide:*

```
10 SPRITEBASE=53248
20 POKE SPRITEBASE+21,4:REM Enable sprite #2.
30 POKE 2042,13 (Set sprite pointer to where sprite
   data should be read from; the address varies for
   each sprite.)
40 FOR X=0 TO 62:READ SHAPE:POKE 832+X,
   SHAPE:NEXT (This loop READs the 63 DATA
   bytes needed to draw a sprite.)
50 FOR LOOP=0 TO 200
60 POKE SPRITEBASE+4,LOOP (Update horizontal
   coordinates of sprite's screen location.)
70 POKE SPRITEBASE+5,LOOP (Update vertical
   coordinates.)
80 NEXT LOOP
90 GOTO 50
   This data draws the sprite:
100 DATA 0,127,0,1,255,192,3,255,224,3,231,224
110 DATA 7,217,240,7,223,240,7,217,240,3,231,224
120 DATA 3,255,224,3,255,224,2,255,160,1,127,64
130 DATA 1,62,64,0,15,128,0,156,128,0,73,0,0,73,0
140 DATA 0,62,0,0,62,0,0,62,0,0,28,0
```

This program creates a hot-air balloon which drifts diagonally down the screen from the upper-left corner, disappears at the bottom, and then starts again from the top. As you see, long variable names are possible, though only the first two letters are significant; it is not necessary to protect areas of memory to display sprites, unlike with other computers; by changing the pointer at line 30 with a quick POKE, it is possible to instantly redraw sprites with alternate shapes previously stored in memory (to fit a sprite's shape to the direction it's traveling, for example); smooth, rapid movement in any direction is possible with simple POKEs into the memory locations which store the horizontal and vertical coordinates.

You'll notice the program is not really very complicated. Remember, though, we moved just one, single-colored sprite with a simple loop; manipulating many multi-colored sprites with joysticks, "firing" shots (made of similar sprites), checking for collisions, and synchronizing sound effects would add considerably to the program's complexity – and slow it down. Fast-action games involving more than a couple of sprites probably will need some help from machine language subroutines. Still, the point is that the basics of sprite animation are quite accessible from BASIC.

Advanced Sound Synthesis

The same is true of the Commodore 64's amazing sound capabilities. When it comes to sound, no other personal computer on the market can hold a diode to the 64.

First, the 64 replaces the common, simple tone generator with a true three-voice sound synthesizer of musical instrument quality. Furthermore, the programmer has control over a great number of parameters: volume, frequency, waveforms, attack/decay, sustain/release, and filtering. This allows you to custom-design each note's "envelope," the shape of its sound wave, and its tone color.

As with the sprites, however, this requires various POKEs, at least until the extended-command cartridge becomes available. Here's an example, borrowed from the *User's Guide,* of how to create one note (a middle C):

10 POKE 54296,15 (Loudest volume setting.)
20 POKE 54277,190 (A number from 0 to 255 sets the attack/decay slope to define how fast a note rises to and falls from its peak volume.)
30 POKE 54278,248 (The opposite of attack/decay, this number sets the sustain/release slope to prolong the note at a certain volume before releasing it.)
40 POKE 54273,17:POKE 54272,37 (This creates a middle C. Two POKEs, for both high and low frequency, are required for each note because of the 16-bit frequency resolution.)
50 POKE 54276,17 (Choose one of four waveforms; in this case, "triangle.")
60 FOR DUR = 1 TO 250:NEXT (A timing loop sets the note's duration; here, a quarter note.)
70 POKE 54276,0:POKE 54277,0:POKE 54278,0 (Turn off the waveform control, attack/decay, and sustain/release settings.)

Whew! Can you imagine programming a symphony?

Actually, it's not so bad. Most of these parameters are set at the beginning of the program and left alone. Values for notes can be stored in DATA statements and summoned easily and quickly with a READ loop. And, as with sprite programming, important memory locations can be stored in variables to save typing.

Still, it may relieve you to know that Commodore plans to sell a plug-in piano-type keyboard for the 64. If Commodore doesn't, someone else will. With the appropriate software, this could make the 64 worth its cost as a musical instrument alone, aside from its other capabilities.

In an appendix of the *User's Guide* (which, incidentally, is crammed with invaluable charts, tables, and pin-maps for the 64) is a listing for a sound demo program that hints at the possibilities. The program converts a row of keyboard keys into a simple organ. Pressing the SPACE bar switches from solo to polyphonic sound. The four programmable function keys select among four octaves and, when shifted, among four waveforms. When the computer is plugged into a decent sound system (easily done), you'll swear you're in the same room with either an organ, electric piano, or harpsichord. It's that good.

And, needless to say, the same capabilities to create beautiful music also give you unprecedented control over sound effects. A gunshot sound can be built like this:

```
10 VOICE1=54296: WAVEFORM=54276:
   ATTACKDECAY=54277: HIFREQ=54273:
   LOFREQ=54272: NOISEWAVE=129
20 FOR LOUDNESS=15 TO 0 STEP -1
30 POKE VOICE1,LOUDNESS
40 POKE WAVEFORM,NOISEWAVE
50 POKE ATTACKDECAY,15
60 POKE HIFREQ,40:POKE LOFREQ,200
70 NEXT
80 POKE WAVEFORM,0:POKE ATTACKDECAY,0
```

Even echo effects are possible by juggling these settings.

Hardware And Software Support

Finally, for those who are interested in buying packaged software as well as (or instead of) writing their own, Commodore is promising a flood of support with the computer's introduction, or soon after. In a first for Commodore, not all of it is being produced in-house. Some hardware goodies are under development, too. All of it is supposed to be for sale by January 1983. Here's a rundown:

- We've already mentioned that most VIC-20 peripherals will work with the 64 with little or no modification. The VIC 1540 disk drive will require a ROM chip change. Already in production is a 1541 drive to replace the 1540 to be compatible with both computers.
- Besides the $109 plug-in VICmodem, another direct-connect, plug-in modem will be made for the 64. This will feature auto-dial and auto-answer and, like the VICmodem, will come with terminal software on tape and a subscription to CompuServe. It will cost less than $200. Two more sophisticated terminal programs will be available on cartridge and disk.
- A PET Emulator will be targeted especially toward schools. Commodore thinks a big selling point of the 64, besides its price, will be the claim that its emulator can run 90 percent of the programs now used by schools on their popular PETs.
- Programmers will be interested in the languages nearing completion for the 64: an assembler for machine language programming, LOGO, and even a BASIC compiler.
- A word processor modeled after *WordPro* is in prototype stages. To accommodate the 64's 40-column screen, the version I saw allowed horizontal scrolling of more than 100 columns. The price will be much less than *WordPro.* A less powerful, and less expensive, word processor for home use will be sold on cassette.
- A whole family of business software is on the way, all with the prefix "Easy" – *EasyCalc, EasyGraph,* and so on. Commodore claims these will be improvements on the current mass of electronic ledger-type programs.

• The plug-in CP/M cartridge promised at recent trade shows and a networking cartridge aimed at schools are also supposed to be well on their way from the drawing boards to the marketplace.

• A new joystick-type controller is in prototype stages and offers unusual possibilities. Sorry, but I had to swear secrecy on this one before they'd even show it to me. It's not a track-ball. But I can say it will work on the VIC-20, too, and probably even on the Ataris.

• Then, of course, there are the games. We've seen some interesting ones, but Commodore hints that even more fantastic stuff using sprites in simulated 3-D is under development. Someone also is working on a music composition utility.

All in all, the Commodore 64 should have some solid hardware and software support behind it much sooner than most new computers, since Commodore seems to be regarding the support as more than just an afterthought. This is a welcome change from the way things were done in the past (not just by Commodore), as even one Commodore official admits. "I think it's safe to say that Commodore has learned from everyone's past mistakes," he says, "and is in a position to capitalize on the lessons." ©

For Upgrade and 4.0 Commodore BASIC, this is a fast, effective sort. It will operate on DATA statements or arrays and rather than exchanging strings, it manipulates their pointers. Records can be of any length. It sorts 200 records in 11 seconds, 800 records in about three minutes. That's speedy. Just type in the BASIC program and it will locate and protect the machine language and adjust it to your ROM version. The program contains some example sorts to get you started.

PET Pointer Sort

David Lummis
Stoney Creek, Ontario

Most previously published sorting subroutines have been written in BASIC. This meant that if you had a large list to sort it would take quite a while to finish.

One day, after I had waited 15 minutes for the computer to finish sorting a list of some 300 names, I decided that I'd had enough.

A friend of mine had suggested writing a machine language program to do the job much faster. This program is the result.

A Bubble With A Twist

I have used a bubble sort with a twist. Instead of switching the data (strings) itself (since not all strings are the same length), I simply switch the pointers that point to where the string is sitting in memory.

In order for you to understand this better, it might be best to explain how variable arrays are stored in memory.

When you DIMension a variable in BASIC, the computer sets aside some room in memory for a table. This table includes all the characteristics of the variable dimensioned. These would include the variable name, whether it's a numeric or string variable, the size of the array, and a few other items.

In order to examine this table we must first find out where it is located in memory. Locations $2C and $2D in hexadecimal (44 and 45 decimal) hold the address where the tables for *all* dimensioned variables start. Locations $2E and $2F hold the address where the last table ends.

As an example, let's assume that location $2C contains the value #$90 in hex, and that location $2D contains the value #$04 in hex. This means that the first table would start at memory location $0490. Here is a typical table created by the statement DIM A$(2): [*type SYS 4 then, in the monitor, type M 0490 0498 to see the hex dump*].

```
0490 41 80 10 00 01 00 03 00
0498 00 00 03 FB 3F 02 F7 3F
```

Figure 1.

The first seven numbers in the table represent what is called the header of the table. The first two characters, 41 and 80, are the name of the variable for that table. In this case 41 hex is the ASCII value of the letter A, the first letter of our variable name. Since we had no second letter in the variable name, the computer placed an 80 where the second letter would normally have gone.

If we had said DIM AA$(2) then the header would have looked like this:

```
0490 41 C1 10 00 01 00 03 00
```

Instead of placing the ASCII value of our second A in the second spot in the header, it placed the ASCII value of a shifted A in its place. This shifted character allows the computer to distinguish between a string variable and a numeric variable. The tables we'll be looking at will deal only with string variables.

The third and fourth bytes of the header are the size of the table, stored in low/high order.

The fifth byte is the number of dimensions in the array. In other words, it would be an 01 in the case of a one dimensional array such as A$(2), an 02 in the case of a two dimensional array such as A$(3,6), and so on.

The sixth and seventh bytes are the number of elements in the array, stored in high/low order.

Following the header there are three bytes reserved for each element in the array. The first of the three bytes is the length of the string. The two bytes following the length is a pointer, stored in low/high order, that points to where the string is stored in memory.

Now let's look back at Figure 1, which can be interpreted as follows:

Bytes	Explanation
1-2	The variable name is A$.
3-4	The length of the table is $0010 bytes long.
5	Indicates that it is a one dimensional array.
6-7	There are $0003 elements in the table.
8-10	Byte 8 (a zero) shows there is no string.
11-13	These show that the second element in the table is a string of three characters starting at $3FFB in memory.
14-16	These show that the third element in the table is a string of two characters starting at $3FF7.

Now let's take a look at the strings themselves as they are stored in memory.

Since A$(0) was a null string, we'll start with A$(1). A$(1) is a string three characters long starting at $3FFB.

```
3FFB 41 42 43 9A 04 40 40 40
```

The first three characters make up the string ABC. The two bytes following each string are a pointer that points back to the table where the length of the string is. Note: only BASIC 4.0 has this pointer following each string.

For the Upgrade ROMs, the string A$(1) would look like this:

```
3FFB 41 42 43 53 45 40 40 40
```

Again, the first three bytes make up the string ABC, which is what A$(1) is equal to. The difference in the Upgrade ROMs is that there is no pointer after each string. The strings are simply stored one after another with nothing in between them. The 53 45 following the string ABC (in Upgrade BASIC) is simply another string.

The Sorting Method

The program compares two strings at a time as in a bubble sort. If the first string is greater than the second one, then the computer switches the pointers that point to where each string is stored in memory. It also has to switch the lengths of each string that it switches. If the program is RUN on a computer with 4.0 ROMs then the pointers after each string, that point back to the table where the length of the string is, will also be switched. When the program returns to BASIC, the array will be sorted in ascending order.

The program is a BASIC loader that places the machine language routine at the top of memory. Line 75 checks to see if it is running on a computer with Upgrade ROMs and, if it is, makes the appropriate corrections to the machine language program. The computer then protects the machine language from being written over.

The variable S is the starting location of the machine language program and is the SYS address which will sort a list. Before sorting a list, however, you must POKE the first two letters of the variable name into memory locations 679 and 680. POKE the ASCII value of the first letter into 679 and the ASCII value of the second *plus* 128. If there is no second letter in the name, simply POKE 680, 128.

For example, to POKE the variable name NA$ into memory you would enter:

```
POKE 679,ASC("N"): POKE 680,ASC("A") + 128
```

For the name N$ it would be:

```
POKE 679,ASC("N") : POKE 680,128
```

The second part of the program is an example of how the sort works in a program. It starts off by having you enter ten strings which the computer then sorts. The key line is line 540, which POKEs the variable name into memory before executing SYS S.

Using DATA Statements

Lines 600-680 show how strings read from DATA statements can be sorted. When data is READ into a string, the pointer in the array table points to the spot in the DATA statement where the string is located. In order to sort strings READ in from DATA statements, the strings must first be placed into upper memory. This is done by adding a null string (two quotes with nothing between them) to the string. Line 630 in the program demonstrates how this is done. If this is not done before sorting your list, then the computer may crash or the strings will contain unexpected characters.

Lines 60-410 are the important lines if you wish to include the sort in a program. They should be executed at the very beginning of a program, before any strings are given a value, in order to get the sort stored at the top of memory right away.

For the computer to sort a complete list of strings, there must not be any empty (null) strings among the ones to be sorted. When the computer sees an empty string within a group of strings containing characters, it assumes that it has reached the end of the list to be sorted and does not sort anything past this point. However, there may be as many null strings as you want *before* or *after* the strings you want sorted.

```
30 REM   THIS ROUTINE SORTS THE STRING
31 REM ARRAY OF YOUR CHOICE IN
32 REM ASCENDING ORDER (A-Z).
38 REM
39 REM   TO ACTIVATE THE ROUTINE IN
40 REM A PROGRAM, TYPE   'SYS S'
41 REM WHERE 'S' IS THE DECIMAL
42 REM STARTING ADDRESS OF THE ROUTINE
43 REM
44 REM   BEFORE ACTIVATING THE ROUTINE
45 REM BE SURE TO POKE THE VARIABLE
46 REM NAME INTO MEMORY LOCATIONS
47 REM 679 AND 680 !
48 REM POKE 679,X WHERE X IS THE ASCII
49 REM VALUE OF THE FIRST LETTER.
50 REM POKE 680,Y WHERE Y IS THE ASCII
51 REM VALUE OF THE SECOND LETTER+128
52 REM   **IMPORTANT***IF THERE IS NO
53 REM SECOND LETTER IN THE NAME THEN
54 REM POKE 680,128.
55 REM
60 REM CALCULATE STARTING(SYS) ADDRESS
63 S=PEEK(52)+PEEK(53)*256-324
64 PRINT"STARTING (SYS) ADDRESS IS"S
70 FORI=0TO324:READA:POKES+I,A:Q=Q+A:NEXT
71 IFQ<>41326THENPRINT"ERROR IN DATA STATEMEN
   TS":STOP
75 IFPEEK(50003)=1THENFORJ=S+229TOS+282:POKEJ
   ,234:NEXTJ
80 REM *PROTECT PROGRAM FROM STRINGS
81 M=INT((S)/256):N=S-M*256:POKE52,N:POKE53,M
   :POKE48,N:POKE49,M
100 DATA 165, 45, 197, 47, 208, 7, 165, 44, 19
   7, 46, 208
110 DATA 1, 96, 165, 44, 133, 1, 165, 45, 133,
   2
120 DATA 160, 0, 177, 1, 205, 167, 2, 208, 8,
```

Skyles Electric Works

Epson-PET/CBM Graphic ROM Pack

For PET/CBM Owners Who Want:

Complete Program Listing Printouts
Complete Screen Graphic Printouts
Graphic Printouts From Programs

on your Epson Printer

Order the Skyles Electric Works EPSON-PET GRAPHIC ROM Package. The ROM when installed in an EPSON MX80, MX80FT, or MX100 with Graftrax Plus printer will reproduce most of the PET/CBM graphics characters. Most importantly when using the accompanying high speed machine language program, the Epson-PET Graphic ROM pack gives a complete program listing with all screen controls shown (cursor, home, clear, etc.). This high speed machine language program for program listing, screen image printout (screen dumps) and BASIC program controlled printing (i.e. PRINT) automatically translates the PET-ASCII characters to the EPSON-GRAPHIC ROM characters. A BASIC sample program and PRINT subroutine, that may be incorporated into any existing BASIC program, completes this "complete solution" package.

EPSON MODEL	ROM MODEL
MX70	Not Available
MX80 (serial no. to 359999)	EPG80 (EPG82, 3 ROM Version)
MX80 (serial no. after 360000)	EPG81 (EPG83, 3 ROM Version)
MX80FT	EPG8F
MX80 Graftrax	Not Available
MX80 Graftrax Plus	EPG8G+
MX80FT Graftrax Plus	EPG8G+
MX100	Not Available
MX100 Graftrax Plus	EPG10G+

The Epson-PET Graphics ROM Pack has been designed to furnish you with PET/CBM graphics printing in the easiest way possible. This is done by furnishing a high speed machine language program that is "hidden" at the top of your PET/CBM memory.

The machine language program serves 3 major functions.

1: Translates PET-ASCII code to ASCII code for program listing.

2: Translates screen code to ASCII code for screen image printouts.

3: Translates PET-ASCII code strings to ASCII strings for normal program printout. This feature may also be used for making ASCII files for your disk or tape recorder.

INSTALLATION: **Installs into your Epson printer**

PRICE: **Epson-PET Graphics ROM Pack EPG80, EPG81, EPG8F.....$75.00**

Please specify your Epson printer model type and and serial number when ordering.

For all PET/CBM's BASIC 2.0/Revision 3, or BASIC 4.0

AVAILABILITY: Immediately from your LOCAL DEALER

or

VISA, MASTERCHARGE ORDERS CALL (800) 227-9998 (except California residents)
CALIFORNIA ORDERS PLEASE CALL (415) 965-1735

Skyles Electric Works
231E South Whisman Road
Mountain View, CA 94041
(415) 965-1735

```
     200
130 DATA 177, 1, 205, 168, 2, 240, 31, 160, 2,
     24
140 DATA 177, 1, 101, 1, 72 , 200, 177, 1, 101

150 DATA 2, 133, 2, 104, 133, 1, 165, 2, 197,
    47, 144, 215, 165, 1, 197
160 DATA 46, 144, 209, 96, 160, 5, 177, 1, 141
    , 160
170 DATA 2, 200, 177, 1, 141, 161, 2, 169, 1,
     141
180 DATA 162, 2, 169, 0, 141, 163, 2, 24, 165,
     1
190 DATA 105, 7, 133, 187, 165, 2, 105, 0, 133
    , 188
200 DATA 165, 187, 133, 177, 165, 188, 133, 17
    8, 24, 165
210 DATA 177, 105, 3, 133, 187, 165, 178, 105,
     0, 133
220 DATA 188, 160, 0, 177, 177, 208, 46, 24, 1
    73, 162
230 DATA 2, 105, 1, 141, 162, 2, 173, 163, 2,
    105
240 DATA 0, 141, 163, 2, 173, 163, 2, 205, 160
    , 2
250 DATA 144, 204, 173, 162, 2, 205, 161, 2, 1
    44, 196
260 DATA 173, 164, 2, 208, 1, 96, 169, 0, 141,
     164
270 DATA 2, 240, 160, 141, 165, 2, 177, 187, 2
    40, 236
280 DATA 141, 166, 2, 200, 177, 177, 133, 185,
     177, 187
290 DATA 133, 189, 200, 177, 177, 133, 186, 17
    7, 187, 133
300 DATA 190, 160, 0, 177, 185, 209, 189, 144,
     174, 240
310 DATA 2, 176, 11, 200, 204, 165, 2, 240, 16
    4, 204
320 DATA 166, 2, 208, 235, 160, 0, 24, 165, 18
    5, 109
330 DATA 165, 2, 133, 214, 165, 186, 105, 0, 1
    33, 215
340 DATA 24, 165, 189, 109, 166, 2, 133, 218,
    165, 190
350 DATA 105, 0, 133, 219, 177, 214, 72, 200,
    177, 214
360 DATA 72, 136, 177, 218, 145, 214, 200, 177
    , 218, 145
370 DATA 214, 104, 145, 218, 136, 104, 145, 21
    8, 160, 0
380 DATA 173, 166, 2, 145, 177, 173, 165, 2, 1
    45, 187
390 DATA 200, 165, 189, 145, 177, 165, 185, 14
    5, 187, 200
400 DATA 165, 190, 145, 177, 165, 186, 145, 18
    7, 169, 1
410 DATA 141, 164, 2, 169, 0, 240, 156, 95, 25
    5, 95

500 PRINT"{CLEAR}":FORI=1TO10
510 PRINT"NOW INPUTTING INTO VARIABLE A$("I")"
    ;:INPUTA$(I):NEXT
520 PRINT"{CLEAR}NOW STARTING SORT.
525 TI$="000000"
535 REM POKE VARIABLE NAME AND SORT LIST
540 POKE679,ASC("A"):POKE680,128:SYS S
550 PRINT"{02 DOWN}SORT FINISHED IN"TI"JIFFIES
    .{DOWN}
560 FORI=1TO10:PRINTA$(I):NEXT
570 PRINT"{02 DOWN}PRESS {REV}RETURN{OFF} TO C
    ONTINUE
580 GETA$:IFA$<>CHR$(13)THEN580
600 REM ***SORTING DATA READ IN FROM
601 REM *DATA STATEMENTS IS SLIGHTLY
602 REM *DIFFERENT FROM INPUTTING
603 REM *THE DATA.
604 DATAZ,D,S,F,H,R,C,H,U,U
605 PRINT"{CLEAR}NOW READING IN DATA.
610 FORI=1TO10:READNA$(I):NEXT
620 REM ***THIS IS THE DIFFERENCE***
630 FORI=1TO10:NA$(I)=NA$(I)+"":NEXT
640 PRINT"{03 DOWN}NOW STARTING SORT.
645 TI$="000000"
655 REM POKE VARIABLE NAME AND SORT LIST
660 POKE679,ASC("N"):POKE680,ASC("A")+128:SYS
    S
670 PRINT"{02 DOWN}SORT FINISHED IN"TI"JIFFIES
    .{DOWN}
680 FORI=1TO10:PRINTNA$(I):NEXT
```

©

Insight: Atari

Bill Wilkinson
Optimized Systems Software
Cupertino, CA

In addition to a continuation of the game development that I started last month, this month's column will delve into the argument of what makes BASIC run, including a chip that makes Atari BASIC run better. But first ...

FMS And Burst I/O, Yet Again

Well, July's column was supposed to fix the mistakes I made in the May column. And then, lo and behold, I blew it again in July. On page 172 of issue 26 of **COMPUTE!** there is a listing of changes to be made to FMS to help burst I/O work properly in update mode. The assembly language listing and the BUG changes were correct. Unfortunately, the POKEs from BASIC had one typo (my fault). The last POKE read

POKE 2773,13 ... WRONG!

should be

POKE 2773,31 ... RIGHT!

Speed And BASIC

Personally, I have never been sure that it is necessary for an interpreted language (e.g., BASIC) to be fast. Typically, I choose to use an interpreter for ease of use and speed of debugging, for writing quickie little programs, and for creating utilities that can run at any speed at any time.

But an increasing number of people are trying to use BASIC for writing serious software, including games, utilities, and business applications. Now I maintain that the speed of BASIC is irrelevant when it is being used for utilities (who cares how fast a disk fixer-upper runs?) or business applications (the program is usually waiting for keyboard, printer, or disk I/O anyway). But for writing games and a certain category of other programs (e.g., sorts), speed is important. But then why use BASIC? Because it's the easiest language to use? Because it can be made fast enough? Because it's the only language the author knows?

Actually, those (and many others) are all valid reasons to choose BASIC, as long as the author doesn't expect more than BASIC is capable of delivering. So what is BASIC capable of delivering? A lot of adequacy. After all, look at some of the very successful games that are written in BASIC (*Crush, Crumble, and Chomp* is the first one that comes to my mind). Or look at some games that should never have been written in BASIC and yet were (a lot come to mind, but I will refrain from naming any).

Certain other authors writing in another magazine have claimed that Atari BASIC is the slowest language ever created. My first impulse was to say, "Who cares? It is the easiest to use, and that's more important." But I simply couldn't take that statement lying down, as it were. After all, if Atari BASIC is such a snail, how come all these programs seem to work just fine?

So I armed myself with five different BASIC interpreters: Applesoft, Atari BASIC, Atari Microsoft BASIC, BASIC A+, and Cromemco's 32K Structured BASIC. Now OSS produced three of these five BASICs, so it might seem that I am prejudiced. Well...maybe a lot, but not too much. Some comments follow on what I decided to try to do.

I wanted to use a benchmark program that would, to some degree, show the fundamental speed of each BASIC. But I also wanted to see what impact such things as constants, variable names, and multi-statement lines would have. Luckily, at just about this same time, I happened upon a benchmark (as yet unpublished) which showed Atari faster than Applesoft in a very simple program. "Oh ho!" says I. "How can this be? Atari is the slowest machine ever, say certain voices."

Anyway, I began experimenting with a small benchmark program, allowing various changes so that I could see the impact on speed. The most fundamental program was simply:

```
10 < start a timer >
20 A=0 : B=12345.6
30 A=A+1.234567
40 IF A < B THEN 30
50 < print time used >
```

Obviously, the intent of this program is to cause a loop to execute 10,000 times. But what can be changed that will significantly affect the execution time without materially altering the program? Below I show all the versions of lines 20 and 30 that I tested. (Line 40 is not shown, but it followed line 20 in the naming of variables and otherwise remained unchanged.) The table also shows the times for the various languages, rounded to three significant figures.

In addition to the timings shown in Table 1, I also tried adding several variables to the programs. Adding 18 variables (in lines 11 and 12) added about five or six seconds to the Microsoft BASICs, about 1.5 seconds for Atari BASIC and BASIC

THE MOSAIC RAM SYSTEMS FOR ATARI*

CLEARLY THE BEST

MOSAIC 32K RAM

THE SCREEN CLARITY TEST

THE COMPETITION

YOU CAN SEE THE MOSAIC DIFFERENCE

WHAT THE EXPERTS HAD TO SAY

A.N.A.L.O.G. 400/800 MAGAZINE said in a 32K RAM board comparison test: "The Mosaic 32K RAM showed no sign of interference and gave the best screen clarity" and "Mosaic uses what we feel are the highest quality components which may improve reliability".

INTERFACE AGE said after replacing the Atari memory board with a Mosaic 32K RAM: "Once in place there is no noticeable change in screen clarity" and "in view of its excellent performance it should be a serious choice for those Atari owners intending to expand their memory."

Each Mosaic RAM board gives you more than just the best screen clarity but also the best in reliability, flexibility, and compatibility. The Mosaic RAM systems offer you the best in every way — these features prove it.

- Works in both the Atari 400*& 800*computers.
- 4 year guarantee.
- Complete instructions.
- Test cycled 24 hours for reliability.
- Gold edge connectors for better reliability.
- Quick no-solder installation.
- Full flexible memory configuration.
- Can be used with 8K, 16K and future products.
- Designed to take advantage of Atari 800's superior bus structure.
- Designed for inter-board communication in Atari 800.
- Allows Atari 800 to have 2 slots open for future expansion.
- Always the best components used for superior screen clarity and reliability.
- Low power design for safety and reliability.
- Available companion board ($5) to allow running The Mosaic RAM systems independent of other boards.

THE MOSAIC RAM SYSTEMS FOR THE SERIOUS ATARI OWNER.

THE MOSAIC 32K RAM

For the serious Atari owner. This is the 32K RAM board you've been hearing about. It has every feature you could want from a RAM board and more features than any other board offers. Each Mosaic 32K RAM comes with complete instructions so in a few minutes you will have expanded your Atari 800 to 48K RAM. The Mosaic 32K RAM works as well in the Atari 400, but we suggest the NEW Mosaic Expander.

THE MOSAIC EXPANDER

This is the most effective way you can expand to 32K RAM for your Atari 400 computer. And at almost the 16K price! The revolutionary Mosaic Expander is a memory board with 16K RAM in place and open slots to add 16K more from the Atari 16K board that comes with your Atari 400. Each board comes with complete instructions so in a few minutes you will have expanded to 32K RAM.

For your nearest Mosaic dealer call toll free

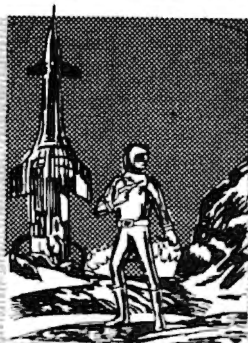
800-547-2807

*Trademark of Atari, Inc.

MOSAIC ELECTRONICS

P.O. Box 708 Oregon City, Oregon 97045
503/655-9574 Toll Free 800-547-2807

Great Autumn Offerings

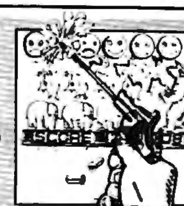


PROBE ONE
From Synergistic Software
High-res action game. You're the commander of the Terran space ship PROBE ONE during the war between the Terran Confederation and the Dreigan Hegemony. Only you can save the human race from extinction by the Dreigans...if you can find the matter transmitter and bring it to Terra. You'll need a quick mind and fast reflexes to fight off the guard robots as you search. Multiple skill levels. BASIC required; paddles or joystick.
40K Tape, ~~$34.95~~ *Save 15%!*
Now Thru Oct. 1 You Pay Just $29.71

CAVERNS OF MARS
By Greg Christensen from APX
An award-winning arcade game that will challenge your skill for many, many plays. You command the fighter Helicon VII on a mission to invade the Martian high command. The headquarters are located deep in the caverns, so you must descend through radar, rockets, laser gates, and space mines to reach your destination. Once there, you activate the fusion bomb and try to get out before it detonates. Good luck! Joystick required.
16K Tape, $34.95 *—WHILE THEY LAST!!*
16K Disk, $37.95

DE RE ATARI
From APX
Translated from Latin, the title of this book is "All About Atari" and it means what it says! Used in combination with Atari's Technical Reference Manual, advanced programmers will be able to learn to exploit the many hardware and operating system features that make the Atari 400/800 so tremendously versatile. Includes a useful discussion of the new GTIA chip. Once you know Atari BASIC and assembler, this book is a must.
Loose leaf (binder not supplied), $19.95

From DataSoft Inc.
Games, Graphics, Word Processing Programs



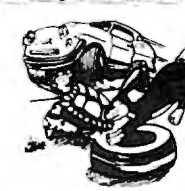
SHOOTING ARCADE
From DataSoft
Step right up, ladies and gentlemen, and try your luck! It's your favorite amusement park game right in your own home! Aim at the elephants, ducks, bunnies and more. Every time you hit the target you score. But aim carefully—don't waste bullets. Fire at a frowning face and your bullets turn to blanks; shoot the smiling face and win BIG! If you make it to the 10th round, there's a *surprise*. Requires joystick.
16K Disk, ~~$29.95~~ *Save 15%!*
Now Thru Oct. 1 You Pay Just $25.46

CLOWNS AND BALLOONS
From DataSoft by Frank Cohen
A huge bunch of balloons is stuck at the top of the circus tent. How do you get them down? By bouncing a clown on a trampoline, high enough to burst them, of course! You must break the balloons in order, one row at a time—if you miss any, the whole row fills up again. Aim your clowns where you want them by moving the trampoline to just the right position. 1 or 2 players; multi-skill levels; joystick or paddle required.
16K Tape or Disk, $29.95

CANYON CLIMBER
From DataSoft by Tim Ferris
You're at the bottom of the Grand Canyon, trying to scale the world's toughest wall. If that's not enough of a problem, there are three challenges to face along the way: angry sheep, threatening Indians and attacking birds. Action-packed arcade game you'll really enjoy. Requires joystick.
16K Tape or Disk, $29.95

SPELL WIZARD
From DataSoft
SPELL WIZARD is a proof-reading package, compatible with Atari's word processor and Text Wizard. It has a dictionary of 20,000 words; you can add up to 5,000 more. Insert your letter, term paper or the latest chapter of your novel, and SPELL WIZARD will proofread it for you. Your copy will be perfect.
32K Disk, $79.95

MICROPAINTER
From DataSoft by Bob Bishop
Create your own color graphics with this "electronic coloring book" for all ages and abilities. The special features include: draw mode, diamond fill, microscope mode for "close-up" viewing, inversing, rubberband line draw, and joystick-controlled easel for selecting pattern, colors and intensities.
48K Disk, ~~$34.95~~ *Save 15%!*
Now Thru Oct. 1 You Pay Just $21.21



PACIFIC COAST HIGHWAY
From DataSoft by Ron Rosen
Fast-action game with good graphics for 1 or 2 players. You're a tortoise, trying to cross the Highway without getting hit by the rush hour traffic. Once across, you're at the shore, and must leap from boats to rafts on the Pacific Ocean. Be careful 'cause you can't swim! Requires joystick.
16K Tape or Disk, $29.95

TEXT WIZARD 2.0
From DataSoft
This is the newest version of the Atari word processing package TextWizard, so popular with writers, students annd small business people. Like the original, you can learn and use TextWizard 2.0 in about an hour; write copy, then edit; underline, insert or replace words or paragraphs (in one place or **throughout** text); print in 1 or 2-column format; get multiple copies, print in boldface, elongated or condensed lettering with proportional spacing. It even warns you about errors *before* you make them!
TEXTWIZARD 2.0 now has file compatibility with File Manager 800 so you can change your text with information taken from File Manager files; address your form letters from File Manager's mailing lists, etc. It's also compatible with: EPSON MX-80 with Graftrax Plus ROMS; NEC 8023-A; Okidata and Pro-Writer. Instruction manual with 3-ring binder.
32K Disk $99.95

Over 1500 Programs for TRS-80, ATARI 400/800, APPLE & IBM.

For Information Call
202-363-9797

Visit our other stores:
Seven Corners Center, Falls Church, VA
W. Bell Plaza, 6600 Security Blvd., Baltimore MD
Coming Soon to Columbus, OH and Greensboro, NC

the PROGRAM STORE™

To Order Call Toll-Free
VISA **800-424-2738** MasterCard

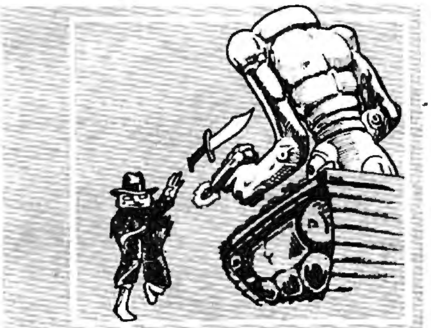
MAIL ORDERS: Send check or M.O. for total purchase price, plus $1.00 postage & handling. D.C., MD. & VA.: add sales tax. Charge cards: include all embossed information on the card.

4200 Wisconsin Avenue, NW, Dept. U209 Box 9609 Washington, D.C. 20016

For Your ATARI™ 400/800

the PROGRAM STORE

From Synapse Software
Arcade-Type Action Games, Application Software



SHAMUS
From Synapse Software
It's the 21st century, and you're the SHAMUS, looking for your arch-enemy, the Shadow, to destroy him. You're armed with Ion-Shivs, the most powerful weapons in the galaxy. Can you handle the Shadow's henchmen: Robo-Droids, Whirling Drones and Snap-Jumpers, all armed and evil? Can you find the Shadow in his lair of 4 levels with 32 rooms each . . . every one of them dangerous? Intensive arcade action; requires joysticks.
16K Tape or Disk, ~~$29.95~~ *Save 15%!*
Now Thru Oct. 1 You Pay Just $25.46

PROTECTOR
From Synapse Software
The planet's inhabitants are endangered by a malevolent alien that beams them to his ship and transports them to an active volcano. You must pick them up one-by-one with your rescue/attack ship and transport them to another city while dodging lasers and rough terrain. After you save as many as possible, the volcano erupts. You must then move each person to a volcano-proof vault in the mountains! Great graphics and sound in this arcade game. Requires joystick.
32K Tape or Disk, $29.95

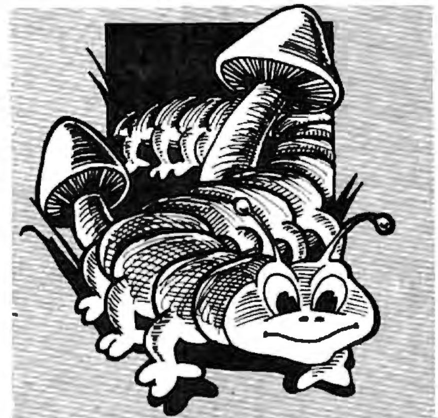
FILE MANAGER 800
From Synapse Software
An extremely powerful and versatile database manager for use in both professional and personal applications. You define the format of the records to be filed and FILE MANAGER 800 gives you full control over sorting, searches, and retrieval.
You can store up to 1000 records on each disk and the program will allow up to four drives. Access to any record takes less than 1.5 seconds, and most commands can be entered with self-prompting single key strokes. Works with Text Wizard 2.0 to make editing easier.
40K Disk, $99.95

FORT APOCALYPSE
From Synapse Software
The Warlords of Kraltha have constructed a prison deep inside Earth—the terrifying FORT APOCALYPSE. All those who have tried to rescue their slave/captives have vanished without a trace. ***Now it's your turn!*** Can you descend thru the Kralthian disruptor fields and penetrate the vast underground Vaults of Draconis? Can your Rocket-Copter fight off the Wormlings, Servo-Tank Interceptors and Robo-Choppers? Multi-player game with arcade action; requires joysticks.
32K Tape or Disk, ~~$29.95~~ *Save 15%!*
Now Thru Oct. 1 You Pay Just $25.46

SLIME
From Synapse by Mike Hales
Large drops of "Slime" fall from the sky onto your ships. If *one drop* hits your ship, it will sink. Use the triangular diverters to aim Slime into buckets on the sides of the screen. If Slime falls into the ocean, the level of the ocean rises. When the ocean reaches the top of the screen, the game is over, and your score is calculated. One-player game with excellent graphics.
16K Tape or Disk, $29.95 Requires joystick.

From Synapse Software
There's trouble in the barnyard, and this machine-language, arcade game will challenge even the most experienced arcade player. You try to help Ma Hen save the eggs and chickens from the wily fox. The action gets faster and faster as eggs turn into chicks, feathers fly, chickens squawk, and all bedlam breaks loose. You'll really have to think fast to outwit this fox.
Requires paddles.
16K Tape, $29.95

NAUTILUS
From Synapse by Mike Potter
Two-player interaction combat game: the underwater cities and their destroyers against the helicopters and submarines. The helicopters bomb the ships; the subs fire torpedoes at the cities and surface to fire at the destroyers. The destroyers fire at the helicopters and drop depth charges on the subs. They can also lower men into the ocean to rebuild cities hit by torpedoes. Top and bottom halves of screen scroll separately for maximum field of play. Great graphics. Requires joystick.
16K Tape or Disk, $29.95



CENTIPEDE
From Atari
Very fast arcade game for the nimble-fingered. The green centipede is streaking across the screen, trying to get from the top to the bottom row. If you're fast enough, you can shoot it—but don't hit it in the ***wrong spot.*** Gain points by killing the spiders and mushrooms, too. Requires joystick.
ROM Cartridge, ~~$42.95~~ *Save 15%!*
Now Thru Oct. 1 You Pay Just $36.51

FROGGER
From On Line Systems
Based on arcade game Frogger. Dodge traffic and jump the moving logs in the stream to get home. Fantastic graphics. Requires joystick.
16K Tape or 32K Disk, $34.95

TEMPLE OF APSHAI
From Automated Simulations
The first of the DujonQuest series, and still one of the most popular. In exploring over 200 rooms in the magical labyrinth, you will encounter more than 30 kinds of fearsome monsters guarding over 70 treasures. Some of the treasures will help you in your quest, but you must still watch out for the many monsters and traps that spring out from the walls and shadows.
32K Tape or Disk, $39.95

THE PROGRAM STORE • Dept. U209 • Box 9609 • 4200 Wisconsin Avenue, NW • Washington, D.C. 20016

Item ___ Tape/Disk/Book ___ Price ___ Postage $1.00 ___ Total ___

☐ CHECK ☐ VISA ☐ MASTERCARD

Name ___
Address ___
City ___ State ___ Zip ___
Card# ___ Exp ___

Table 1. The Speed Matrix

Lines 20 and 30	Cromemco 32K BASIC	Atari BASIC	Atari uSoft	BASIC A+ (Atari)	Applesoft
20 A=0 : B=12345.6 30 A=A + 1.234567	37.0	72.6 (63.6)	270.	62.9 (59.3)	275.
20 A=0 : B=12345.6 30 A=A + 1.23456789	37.0	72.6 (63.6)	710.	62.9 (59.3)	350.
20 A=0:B=12345.6:C=1.234567 30 A=A+C	37.0	73.1 (64.1)	56.3	63.4 (59.8)	50.8
20 LONGVARIABLEA=0: LONGVARIABLEB=12345.6 30 LONGVARIABLEA= LONGVARIABLEA+1.234567 **	37.0	72.6 (63.6)	320.	62.9 (59.3)	can't do
20 LONGVARIABLEA=0: LONGVARIABLEB=12345.6 30 LONGVARIABLEA= LONGVARIABLEA+1.23456789 **	37.0	72.6 (63.6)	752.	62.9 (59.3)	can't do
20 LONGVARIABLEA=0: LONGVARIABLEB=12345.6: C=1.234567 30 LONGVARIABLEA= LONGVARIABLEA+C **	37.0	73.1 (64.1)	106.	63.4 (59.8)	can't do

** These tests made using double precision variables in Cromemco BASIC and Atari Microsoft BASIC. Single precision times were shorter, but not significantly so.
() Times shown in parentheses are explained in the text.

A+, and nothing at all to Cromemco BASIC.

Also, I tried the effects of combining lines 30 and 40 into a single line. For example:

```
30 A=A+C : IF A<B THEN 30
```

The time savings were all in the area of one second, not surprisingly, so I have not detailed them here.

But, look at the surprises! Let's look at the "foreigner," Cromemco 32K BASIC, first. *Nothing* seems to make a difference to it! Actually, I knew that this would happen before I ran the tests. Of all the BASICs shown, Cromemco's is the most like a compiler. I simply included it to give you an idea of what a truly properly structured interpreter can accomplish, but we must be fair and admit that the language is 26K bytes in its smallest usable incarnation.

For you Atari BASIC and BASIC A+ programmers, the happiest surprise is perhaps simply finding out that these languages do as well as they do. Also, note that the various program changes have only a small effect on the running times. So you don't have to be too careful about how you write your programs. (But it is still true that putting subroutines and FOR/NEXT loops at the beginning of a program will make a noticeable speed difference. Don't feel too bad: all Microsoft BASICs have this same quirk.)

And now to the Microsoft BASICs. Obviously, you pay a penalty for using constants in a loop. Using double precision constants (1.23456789 in our examples) costs so much that you should try to avoid them. Watch for long variable names: 41 seconds to go from a one-character name to LONGVARIABLEA ? Ouch! (Actually, I also tried three-character names and found the penalty there to be over seven seconds.) And there is a penalty for having lots of variable names in use. Hmmm... we need to use variable names instead of constants, because constants are so slow; but using lots of variable names costs time also, so...

How about the other side of the Microsoft coin? What can we do that will show off the Microsoft BASIC speed? Two answers: use integer variables and do some transcendental function calculations. It's reasonably obvious why integer variables help: integer arithmetic is guaranteed to take less time than floating point. But why the transcendentals, if we just showed that the speeds are similar? Simple. I cheated. I used only addition, where the Atari BASIC floating point package shows up pretty good. But oh boy! Did we blow it when it comes to multiply! When using SIN, COS, etc., Atari Microsoft BASIC is three to six times faster

than Atari BASIC. Until now. But before I explain that "until," let me make a few points.

Microsoft BASIC is definitely capable of more speed than Atari BASIC, but *only* if you are *very* careful and use lots of programming tricks. If you are an advanced programmer, this won't bother you. But I still believe, as I did over three years ago when we designed Atari BASIC, that for most people (and especially for beginners and hackers like me) the ease of use that is the hallmark of Atari BASIC makes it a real standout. But of course I'm not the perfect, objective judge. So try all of the BASICs, if your budget can afford it, and judge for yourself.

Fast, Faster, Fastest

This section will explain that "until now" that I wrote in the next to last paragraph. As I said, we (OSS and predecessors) blew it when it came to implementing the multiply algorithm, and as a result the transcendental routines take long enough for you to go out and get a cup of coffee. *But...*

Newell Industries (alias Wes Newell) of 3340 Nottingham, Plano, Texas (75074) has introduced the *Fastchip*. Actually, the *Fastchip* is a ROM which replaces the OS Floating Point ROM in an Atari 400 or 800. Major portions of the 2K bytes of ROM have been changed, resulting in several speed and/or accuracy improvements. The biggest changes were to the multiplication (ta da!) routine and floating-point to integer conversion (which is used *all* the time, by GOTO, POKE, SETCOLOR, XIO, OPEN, and many, many other statements and functions).

I have said that I will not normally review software, but I think the *Fastchip* deserves an exception to this rule on two points: it *can* be considered hardware, and it is a must for anyone contemplating heavy math usage with an Atari. Just as an example, note the times in parentheses in Table 1. These times are those recorded with a *Fastchip* installed. And this in a benchmark which does *not* make heavy use of *Fastchip's* best features!

Newell Industries has done some fairly complete timings of the various routines, so I won't belabor that point here. I will, however, include my own small benchmark program, just to give you an idea of the improvements available.

As you will note, I have included the Microsoft timings, also. Quite frankly, comparing Microsoft with Atari BASIC in this benchmark is almost as ludicrous as the reverse comparisons in Table 1. Which perhaps says a lot about how worthwhile benchmark programs *really* are.

Anyway, note that using the *Fastchip* brings the Atari BASIC timings within striking range of the Microsoft timings. A *most* respectable performance when you consider that the Atari BASIC routines use six byte floating point while Microsoft uses a four byte floating point. Incidentally, the BASIC A+ timings were all only a small fraction of a second faster than the Atari BASIC times here, so I omitted them.

Enough hard work. On with the games!

Table 2. Transcendental Timings

line 30	Atari Microsoft	Atari BASIC	Atari BASIC with Fastchip
30 J=ABS(I)	1.15	1.53	1.48
30 J=SIN(I)	6.85	25.3	10.9
30 J=EXP(I)	6.75	33.7	9.93
30 J=I*I	12.4	74.0	20.8

```
10 <start timer>
20 FOR I=0 TO 6.3 STEP 0.02
30 J= <a function of I...see table>
40 NEXT I
50 <print elapsed time>
```

(program used with Table 2)

BOING ... Part 2

Last month, we started with a simple program to bounce a ball around in a box. We noted some problems having to do with bouncing fast balls against a wall when the "clock" is slow: either the ball hits the wall "invisibly" or the bounce has to look strange. This month, we will extend that program into a real game and present an alternative method of moving the ball.

If you did type in last month's program, you might try changing it so that you assign XMOVE and YMOVE instead of having the program pick random directions. I would suggest that you try values of 0, 0.5, 1.0, and 2.0 in various combinations. If you choose XMOVE=1 and YMOVE=0.5, you will accomplish roughly what this month's program will use. Note, though, that the ball appears to jerk across the screen in strange directions. If you slow down the movement loop (put a delay in it), you will see that the ball really does go in as straight a line as it can (given the coarseness of the display we chose, Graphics 3). The jerkiness is simply an optical illusion, as far as I can tell, due to your eye expecting a certain movement and then being fooled.

The solution? Really, with finite pixel positioning, there is none. But you can greatly improve the situation by using a higher resolution graphics mode while retaining a relatively large ball: the

jumps in the higher resolution mode are small in comparison to the ball and so are not perceived as readily. With an Atari, the easiest way to accomplish this is with Player/Missile Graphics; but I will not delve into that in this series of articles since the subject has been covered so thoroughly and well elsewhere. Rather, the intent of these articles is simply to give beginners to graphics and/or assembly language a start in converting ideas from paper to BASIC to assembler.

This month, though, there simply isn't room or time to show and explain both the BASIC program and its assembly language counterpart. So the assembly language version will wait for next month, but I promise that it will be as closely related to this month's BASIC as last month's pair of programs were interrelated.

By the way, for those of you who simply want to play the game, just type it in as carefully as possible. Then simply RUN it for a two player Table Tennis-like game, using joysticks (not paddles – and, by the way, you must hold the joysticks turned 90 degrees left from normal position). For a one player game (not exciting, but a good demo), hold down the START key as you hit the RETURN key after typing RUN. And thus we start a skeletal explanation of how this program works.

What Makes BOING Ping?

First, note that YP(x) and SCORE(x) are simply the Y (vertical) paddle position of player "x" and a count of that same player's misses (x is 0 or 1, only). SINGLE is a flag set by examining the console switches which creates either a two player or one player game. LASTWIN is a -1 or +1 flag which indicates who scored the last point (we initialize it randomly).

At line 2000, the real work begins. In Graphics mode 3, we draw top and bottom boundaries and left and right paddles and print the current score. If this is a single person game, we overlay the right paddle with another wall. Also, in line 2060, we initialize each player's paddle position to 10, smack in the middle of each side. The ball is also initialized somewhere in the middle of the screen and given a starting shove.

At lines 2600 and 2700, we use my trick for reading the left and right joystick positions (this is the reason for turning the paddles), and we skip moving the paddle if the joystick is centered (and we never move the right paddle in a SINGLEs game). The method of moving a paddle is sheer simplicity: since each paddle is three units high, we erase the pixel on one end and create a new one on the other end. Presto, the paddle is moved. Oh, yes, we update YP(x).

Then, at line 3000, we start moving the ball. This is pretty much like last month, except that the XMOVE is always plus or minus one while the YMOVE is -1, -0.5, 0, +0.5, or +1. Note that if the ball won't hit something on its next move, it is because it will miss a paddle, so someone (HITP) will lose a point.

But if the ball is hit by a paddle, its YMOVEment is not determined by simple reflection. Rather, if the ball hits the center of a paddle, it is reflected straight across the playing field (with YMOVE=0). If it hits directly on either side of center, it returns at a slight angle (YMOVE = -0.5 or +0.5). But if it just barely hits the edge of the paddle, it rebounds at a satisfactorily nasty angle (YMOVE= -1 or +1). All this is done in line 3080.

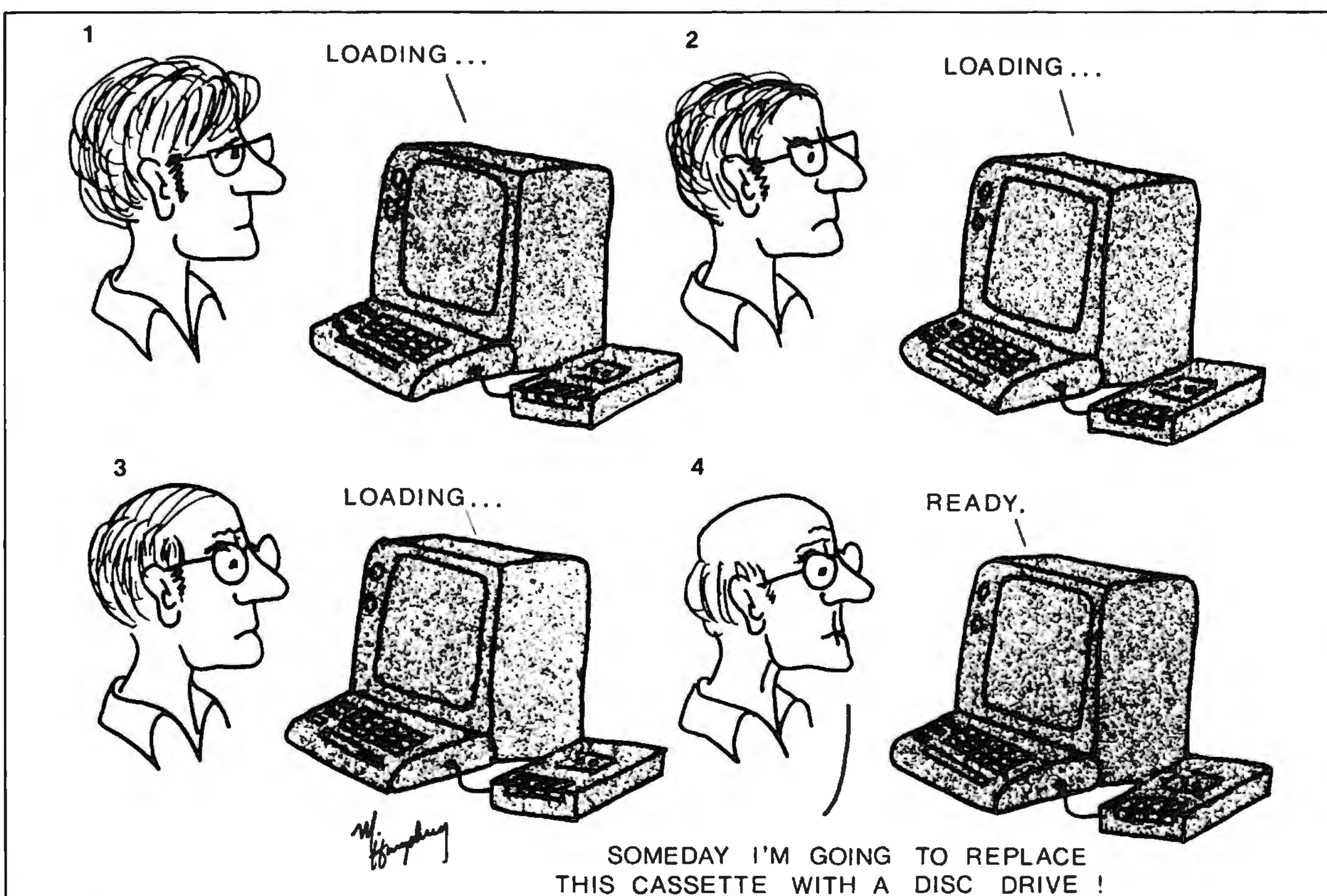
Finally, the "LOSE" and "SCORE" routines are fairly simple. We force the ball to continue its flight for two more steps and then make a nasty noise and a simple but flashy display. We award a hit point as appropriate and figure out who LASTWIN should be.

This is *not* a sophisticated game. It is *not* intended to awe you with the power and flexibility of the Atari computer. It is intended to be a simple enough game that most of you will be able to follow its logic. And it certainly is intended to be easily translated to assembly language. But that's next month.

```
1000 REM *** STARTUP THE GAME ****
1010 DIM YP (1),SCORE(1):SCORE(0)=0:SCORE(1)=0
1020 SINGLE=(PEEK(53279)<>7)
1100 LASTWIN=1:IF RND (0)>=0.5 THEN LASTWIN=-LA
   STWIN
2000 REM *** PREPARE FOR A SERVE ***
2010 GRAPHICS 3: COLOR 2: PLOT 0,0:DRAWTO 39,0
2020 PLOT 0,19:DRAWTO 39,19
2030 PRINT :PRINT SCORE(1),,SCORE(0):PRINT "
       SCORE";
2035 IF SCORE(0)>20 OR SCORE(1)>20 THEN END
2040 COLOR 3 :PLOT 0,9:DRAWTO 0,11:PLOT 39,9:DR
   AWTO 39,11
2050 IF SINGLE THEN COLOR 2:PLOT 39,0:DRAWTO 39
   ,19
2060 YP(0)=10:YP(1)=10:REM VERTICAL POSITION
2070 IF SINGLE THEN LASTWIN=1
2100 REM SET UP BALL
2110 XMOVE=LASTWIN:YMOVE=INT(3*RND(0))--1:Y=INT
   (12*RND(0))+4
2120 YNEW=Y:X=19-5*XMOVE:XNEW=X
2500 REM *** MAIN PLAYING LOOP ***
2510 REM
2520 REM 1. CHECK AND MOVE PADDLES
2530 REM 2. SHOW NEW BALL POSITION
2540 REM 3. CHECK FOR COLLISIONS, ETC.
2550 REM
2590 REM *** FIRST CHECK AND MOVE PADDLES
2600 V0=PTRIG(0)-PTRIG(1):IF NOT V0 THEN 2700
2610 VP0=YP(0)-V0:IF VP0<2 OR VP0>17 THEN 2700
2620 COLOR 0:PLOT 0,YP(0)+V0:COLOR 3:PLOT 0,VP0
   -V0:YP(0)=VP0
2700 V1=PTRIG(2)-PTRIG(3):IF SINGLE OR V1=0 THE
   N 3000
2710 VP1=YP(1)-V1:IF VP1<2 OR VP1>17 THEN 3000
2720 COLOR 0:PLOT 39,YP(1)+V1:COLOR 3:PLOT 39,V
```

```
    P1-V1:YP(1)=VP1
3000 REM *** BALL CONTROL ***
3010 COLOR 0:PLOT X,Y
3020 COLOR 1:PLOT XNEW,YNEW
3030 X=XNEW:Y=YNEW
3040 XNEW=XNEW+XMOVE:YNEW=YNEW+YMOVE
3050 IF XNEW<38 AND XNEW>1 THEN 3200
3060 HITP=(XNEW>20):XHIT=39*HITP
3070 IF SINGLE THEN IF HITP THEN 3100
3080 YMSAVE=YMOVE:YNEW=INT(YNEW):YMOVE=(YNEW-YP
    (HITP))/2
3090 IF ABS(YMOVE)>1 THEN GOTO 4000
3100 XMOVE=-XMOVE
3200 IF YNEW=1 OR YNEW=18 THEN YMOVE=-YMOVE
3290 GOTO 2600
4000 REM *** THE 'LOSE' ROUTINE
4010 COLOR 0:PLOT X,Y
4020 COLOR 1:PLOT XNEW,YNEW
4030 FOR I=1 TO 10:NEXT I
4040 COLOR 0:PLOT XNEW,YNEW
4050 COLOR 2:PLOT XNEW+XMOVE,YNEW+YMSAVE
4130 SOUND0,132,12,12:POKE 20,0
4140 SETCOLOR 1,0,PEEK(20)*4:IF PEEK (20)<32 TH
    EN 4140
4150 SOUND 0,0,0,0
4200 REM *** SCORE IT ***
4210 SCORE(HITP)=SCORE(HITP)+1
4220 LASTWIN=1:IF HITP THEN LASTWIN=-LASLTWIN
4990 GOTO 2000
```

©



PET Autoload

Lou Cargile and Richard Mansfield

"Autoload" for 80-column PETs allows you to easily LOAD and RUN any program from disk drive zero. You put the directory on screen and then move the cursor to the line which lists the target program. Hit the backslash key "\ " and the program boots into the computer and takes control. The cursor need not be in the first position on a line; it can be anywhere.

The routine is turned on (or off) by a SYS 634. The only potential problem would be inadvertently LOADing a program which overwrites the second cassette buffer. While Autoload is armed, all output to the screen jumps through the code at $0287. If this is overwritten, there will be trouble. Otherwise, this routine might be handy to place, for example, at the beginning of a disk of game programs. It would facilitate easy selections.

Autoload makes use of the vector at $EB, $EC, which can intercept characters on the way to the screen. This is an aspect of the newer Commodore machines which perhaps deserves further exploration. The disassembly (Program 2) illustrates how all characters typed are first examined by the routine starting at $0287. At $027A, the contents of $EB and $EC are toggled to alternatively point to $0287 or back to the normal $E20C. The routine in BASIC ROM which handles output to the screen passes through $E209 which is an indirect JMP down to $00EB. Normally, it bounces right back to $E20C because this vector at $EB is loaded with $E20C during initialization.

With the output to the screen flowing through $0287, a quick test of $D9 reveals whether or not the backslash key is being pressed. If it is, the keyboard buffer is loaded with the characters which will cause an automatic LOAD/RUN. Since LOAD itself returns BASIC to a warm start condition, the RUN must be pushed into the keyboard buffer. It would not survive LOAD if printed to the screen on the same line as LOAD.

In addition to this application, the concept of pre-screen interception could solve a variety of other programming problems. For example, you could assign certain keys to transmit special characters to a printer. Or a LISTing could be sent to the printer, spelling out cursor characters, in the fashion of **COMPUTE!**'s listing conventions. In effect, this technique gives you real control over the keyboard.

Program 1: Autoload

```
100 DIMCODE(87)
110 FORN=634TO721:BYTE=N-634
120 READCODE(BYTE):POKEN,CODE(BYTE)
130 NEXT
140 IFPEEK(235)=12ANDPEEK(236)=226THEN
   SYS634
150 DIRECTORY
200 DATA165,235,73,139,133,235,165,236
210 DATA73,224,133,236,96,165,217,41
220 DATA127,201,92,208,64,169,13,141
230 DATA111,2,141,115,2,169,147,141
240 DATA112,2,169,82,141,113,2,169
250 DATA213,141,114,2,169,5,133,158
260 DATA165,217,201,92,208,4,169,1
270 DATA133,158,169,1,133,198,169,68
280 DATA32,210,255,169,204,32,210,255
290 DATA169,32,32,210,255,169,24,133
300 DATA198,169,58,133,217,76,12,226
```

Program 2: Autoload Disassembly

```
027A A5 EB       LDA $EB
027C 49 8B       EOR #$8B
027E 85 EB       STA $EB
0280 A5 EC       LDA $EC
0282 49 E0       EOR #$E0
0284 85 EC       STA $EC
0286 60          RTS
0287 A5 D9       LDA $D9
0289 29 7F       AND #$7F
028B C9 5C       CMP #$5C
028D D0 36       BNE $02C5
028F A9 0D       LDA #$0D
0291 8D 6F 02    STA $026F
0294 8D 73 02    STA $0273
0297 A9 93       LDA #$93
0299 8D 70 02    STA $0270
029C A9 52       LDA #$52
029E 8D 71 02    STA $0271
02A1 A9 D5       LDA #$D5
02A3 8D 72 02    STA $0272
02A6 A9 05       LDA #$05
02A8 85 9E       STA $9E
02AA A9 01       LDA #$01
02AC 85 C6       STA $C6
02AE A9 44       LDA #$44
02B0 20 D2 FF    JSR $FFD2
02B3 A9 CC       LDA #$CC
02B5 20 D2 FF    JSR $FFD2
02B8 A9 20       LDA #$20
02BA 20 D2 FF    JSR $FFD2
02BD A9 18       LDA #$18
02BF 85 C6       STA $C6
02C1 A9 3A       LDA #$3A
02C3 85 D9       STA $D9
02C5 4C 0C E2    JMP $E20C
```

There are times when you'll want to process other kinds of disk files besides Text files. The technique is illustrated with a useful cross-reference program which shows where and how often variables are used in a BASIC program.

Process Any Apple Disk File

Keith Falkner
Venice, FL

Apple's Disk Operating System recognizes four types of files: Applesoft, Integer, Binary, and Text. When the DOS command CATALOG is entered, the names of all files on the disk are displayed, and the type of each file is indicated by a letter A, I, B, or T. A-files are of course Applesoft programs and are stored by the DOS command: SAVE programname. Similarly, I-files are programs in Integer BASIC. B-files are merely copies of memory onto disk, although they are often machine-language programs or subprograms. T-files are the only genuine data files, and these have invariably been written by programs.

Apple DOS contains a very sensible restriction: a program may OPEN only a Text file. Investigation verifies that all the other types of files usually contain many null bytes, that is, bytes with no bits on, or in Applesoft, CHR$(0). Unless a program explicitly writes CHR$(0), a Text file will never contain a null byte. So when data is being read from an open file into memory, Apple DOS tests each byte transferred. If a null byte is found, Apple DOS assumes that the program has read beyond the end of data in the Text file and issues the error message END OF DATA IN #####, where ##### is a line-number.

Many Good Things

This restriction really is a great nuisance, because there are many good things we could do if only a program could process the other types of files. For instance, a program could print a program listing neatly, produce a cross-reference report, or devise some documentation from the REM statements in an Applesoft or Integer BASIC program.

In fact, a program can circumvent the restriction and OPEN any type of file, by patching DOS as follows:

POKE 42948,234: POKE 42949,169: POKE 42950,0 : REM DOS 3.2 OR 3.3 IN 48K. This changes the instruction at $A7C4 from EOR $B5C2 into NOP and LDA #0, and thus circumvents the test for type of file.

Those POKEs are effective until DOS is rebooted (via PR#6 for example). If the DOS command INIT is issued after the POKEs, the disk so initialized will contain the patch permanently, thus the version of DOS on it will never be able to issue the error message FILE TYPE MISMATCH for the OPEN command.

Figure 1. The Cross-Reference Program's Variables

```
A     200 280   330 340 420 430
A$    330 490   520 530 540 550 620 630
A$(   100 360   370 380 390
B     210 220   260 350 360 370 380 390
B$    510 530   550 630 640
B$(   100 510
C     310 400   420 440 450 460 470 480 580
C$    510 550   630 670
C(    100 140   150 160 340
C1    310 340   400 450 470
C2    150 230   310 400 410 480 610
C9    340 440   450 470 500 510
J     140 150   220 250 260 360 370 380 680
K     220 250   260 350 370 380 390 610 620 630
L     300 310
L$    220 240   310
M$    360 370   390 400 480 490
P$    180
Q     200 280   300 330 520 690
Q$    330 520   690
QQ$   120 540
S$    120 310   640 660 670
X     220 260
X$    230 240   250 260 600 630 640
X$(   100 250   260
Y     640 650   660
Z     580 650
Z$    580
```

Figure 2. The Cross-Reference Program's Line Numbers

```
210  500  520
250  250
280  210
330  500  530
390  350  370
400  360
410  340
450  410
490  470
500  420  430  440  460  480
510  500
520  540  550
560  280
630  610
650  630
670  650
690  200  280  300  330  520
700  190
```

Any program using the above technique to OPEN a non-Text file must be prepared to detect and process the expected null bytes; therefore, the commands GET and INPUT will not work. Fortunately there is a convenient routine in DOS, and here is how to use it:

PRINT CHR$(4) "READ" filename
CALL 42636 : Q=PEEK (46531)

The above line corresponds to a GET statement and delivers in Q the ASCII value of one byte. If the byte is a null byte, then Q will simply be zero, and that can be processed as easily as any other value. DOS will not issue the error message END OF DATA, unless the program reads a byte past the last sector containing the disk file.

The final tip is therefore how to detect end-of-data when processing a file other than a Text file. Actually that is the easiest part, and is simple to deal with when the file has just been opened. Both A-files and I-files start with a two-byte counter which indicates how many bytes of data are in the program, i.e., how many bytes to process. B-files contain this same counter, but before it is another two-byte counter which tells where the image of memory is to be loaded. As usual with 6502 software, all these two-byte counters have the less significant byte first and the more significant last.

Illustrating these techniques is an adaptation of a program which first appeared in **COMPUTE!**, May/June 1980. The program prints a cross-reference list of either the variables or line-numbers in an Applesoft program. The program is very handy and was admirably explained by its esteemed author.

```
100 TEXT : HOME : NORMAL : DIM A$(15),B$(3),X$
    (500),C(255)
110 PRINT "CROSS-REF    JIM BUTTERFIELD": PRIN
    T
120 QQ$ = CHR$ (34):S$ = "      ":B$(1) = Q$:B$
    (3) = CHR$ (58)
130 INPUT "VARIABLES OR LINES? ";Z$:C2 = 5: IF
     ASC (Z$) = 76 THEN C2 =6
140 FOR J = 1 TO 255:C(J) = 4: NEXT : FOR J = ~
    48 TO 57:C(J) = 6: NEXT
150 IFC2=5THENFORJ=65TO90:C(J)=5:NEXT:C(36)=7:
    C(37)=7:C(40)=8
160 C(34) = 1:C(178) = 2:C(131) = 3
170 PRINT : INPUT "PROGRAM NAME: ";P$
180 IF P$ < "A" THEN PRINT CHR$ (4)"CATALOG": ~
    GOTO 170
190 GOSUB 700: PRINT CHR$ (4)"OPEN"P$: PRINT C
    HR$ (4)"READ"P$
200 GOSUB 690:A = Q: GOSUB 690: PRINT 256 * Q ~
    + A" BYTES"
210 IF B = 0 GOTO 280
220 PRINT L$;:K = X: FOR J = B TO 1 STEP - 1: ~
    PRINT " ";A$(J);:X$ = A$(J)
230 IF C2 = 6 AND LEN (X$) < 5 THEN X$  = " " ~
    + X$: GOTO 230
240 X$ = X$ + L$
250 IF X$(K) > = X$ THEN X$(K + J) = X$(K):K =
    K - 1: GOTO 250
260 X$(K + J) = X$: NEXT J:X = X + B: PRINT :B
    = 0
270 REM : GET NEXT LINE, TEST END
280 GOSUB 690:A = Q: GOSUB 690: IF A + Q = 0 G
    OTO 560
290 REM GET LINE NUMBER
300 GOSUB 690:L = Q: GOSUB 690:L = Q * 256 + L

310 C = C2:C1 = - 1:L$ = RIGHT$ (S$ + STR$ (L)
    ,6)
320 REM GET BASIC STUFF
330 GOSUB 690:A = Q:A$ = Q$
340 C9 = C(A): IF C9 > C1 GOTO 410
350 K = 0: IF B = 0 GOTO 390
360 FOR J = 1 TO B: IF A$(J) = M$ GOTO 400
370 IF A$(J) < M$ THEN NEXT J:K = B: GOTO 390
380 FOR K = B TO J STEP - 1:A$(K + 1) = A$(K):
     NEXT K
390 B = B + 1:A$(K + 1) = M$
400 C = C2:C1 = - 1:M$ = ""
410 IF C2 = 5 GOTO 450
420 IF A = 171 OR A = 172 OR A = 176 OR A = 19
    6 THEN C = 6: GOTO 500
430 IF A = 44 OR A = 32 GOTO 500
440 IF C9 < > 6 THEN C = 9: GOTO 500
450 IF C9 = C THEN C = - 1:C1 = 4
460 IF C > 6 GOTO 500
470 IF C < 0 AND C9 > C1 AND C9 > 6 THEN C1 = ~
    C9: GOTO 490
480 IF C2 = 5 THEN IF LEN (M$) > 2 OR C > 0 GO
    TO 500
490 M$ = M$ + A$
500 ON C9 + 1 GOTO 210,510,510,510: GOTO 330
510 B$ = B$(C9):C$ = ""
520 GOSUB 690:A$ = Q$: IF Q = 0 GOTO 210
530 IF A$ = B$ GOTO 330
540 IF A$ < > QQ$ GOTO 520
550 A$ = B$:B$ = C$:C$ = A$: GOTO 520
560 PRINT : PRINT CHR$ (4)"CLOSE"
570 INPUT "PRINTER? ";Z$
580 C= 3:Z = 6: IF ASC (Z$) = 89 THEN C = 4:Z ~
    = 12: PRINT CHR$ (4)"PR#1"
590 PRINT : PRINT "CROSS REFERENCE - PROGRAM "
    ;P$
600 X$ = "": FOR J = 1 TO X:A$ = X$(J)
610 IF C2 = 6 THEN K = 6: GOTO 630
620 FOR K = 1 TO LEN (A$): IF MID$ (A$,K,1) < ~
    > " " THEN NEXT K: STOP
630 B$ = LEFT$ (A$,K - 1):C$ = MID$ (A$,K + 1)
    : IF X$ = B$ GOTO 650
640 PRINT :Y = 0:X$ = B$: PRINT X$; MID$ (S$,1
    ,5 - LEN (X$));
650 Y = Y + 1: IF Y < Z GOTO 670
660 Y = 1: PRINT : PRINT S$;
670 PRINT LEFT$ (S$,6 - LEN (C$));C$;
680 NEXT J: PRINT : PRINT CHR$ (4)"PR#0"; END
690 CALL 42636:Q = PEEK (46531):Q$ = CHR$ (Q):
    RETURN
700 POKE 42948,234: POKE 42949,169: POKE 42950
    ,0: RETURN
```

©

This extension of the TRS-80 "Energy Monitor" program requires Extended BASIC and 16K.

TRS-80 Color Computer Energy Monitor Graphics

Linton S. Chastain
Greensboro, NC

For those of you who have added "Energy Monitor" (**COMPUTE!**, August 1982) to your software repertoire, I have written a corollary program called "Energy Monitor Graphics." This new program will give you a visual display of your annual energy consumption or cost. It can be either displayed to your TV screen or printed out. This new program uses the data files created by the "Energy Monitor" program to generate the graphics.

The program was written on a TRS-80 Color Computer with Extended BASIC with disk drive. For those of you who do not have disk yet, you will have to change lines 70, 140, 160, and 180. The following changes will permit you to enter the data files from a cassette recorder:

```
70 CLS:PRINT"1-READ OLD MASTER FILE":
   PRINT"FROM CASSETTE
140 OPEN"I",#-1,T$:PRINT"READING FILE: ";T$:
   INPUT#-1,N
```

```
 183   GAS UNITS        UTIL80/A
l    G
l    G   G
l    G   G   G
l    G   G   G
l    G   G   G
l    G   G   G
l    G   G   G                                   G
l    G   G   G                                   G
l    G   G   G   G                               G
l    G   G   G   G                               G
l    G   G   G   G   G                       G   G
l    G   G   G   G   G   G   G   G   G   G   G   G
----------------------------------------------------
    JAN FEB MAR APR MAY JUN JUL AUG SEP OCT NOV DEC

 63.92   ELECTRIC COST   UTIL80/A
l                                    E
l                                E   E
l                                E   E
l                                E   E
l                                E   E   E
l                                E   E   E   E
l        E                   E   E   E   E   E
l    E   E   E   E           E   E   E   E   E   E
l    E   E   E   E   E   E   E   E   E   E   E   E
l    E   E   E   E   E   E   E   E   E   E   E   E
l    E   E   E   E   E   E   E   E   E   E   E   E
l    E   E   E   E   E   E   E   E   E   E   E   E
----------------------------------------------------
    JAN FEB MAR APR MAY JUN JUL AUG SEP OCT NOV DEC
```

```
160 FORJ=1 TO N:INPUT#
   -1,D$,A(J),B(J),C(J),D(J),E(J),
   F(J),G(J):PRINTJ:NEXTJ
100 CLOSE
```

This program's displays will help you see the effects of the seasons on your energy consumption and cost. It may help you organize your strategy to conserve energy.

```
30 CLEAR200
40 MR=20:N=0
50 DIMD$(MR),A(MR),B(MR),C(MR),D(MR),E(MR),F(
   MR),G(MR),K(MR),T(MR)
60 R=0
70 CLS:PRINT"1-READ OLD MASTER FILE FROM DISK
   "
80 PRINT"2-DISPLAY GRAPH"
90 PRINT"3-RETURN TO COMMAND LIST"
100 INPUT"ENTER COMMAND BY NUMBER";R:IFR<1 OR
    R>3 THEN60
110 ON R GOSUB 120,200:GOTO60
120 R$*"READING":PRINT
130 INPUT"NAME OF FILE ";T$
140 OPEN"I",#1,T$:PRINT"READING FILE: ";T$:INP
    UT#1,N
150 PRINT"READING RECORDS # "
160 FORJ=1 TO N:INPUT#1,D$(J),A(J),B(J),C(J),D
    (J),E(J),F(J),G(J):PRINTJ:NEXTJ
170 PRINT:PRINTN;" DATA RECORDS READ"
180 CLOSE#1
190 PRINT"PRESS <ENTER> TO RETURN.":GOSUB1120:
    RETURN
200 CLS:PRINT"COMMAND LIST # 2":PRINT
210 PRINT "1-DISPLAY WATER COST"
215 PRINT "2-DISPLAY WATER UNITS"
220 PRINT "3-DISPLAY GAS COST"
225 PRINT "4-DISPLAY GAS UNITS"
230 PRINT "5-DISPLAY ELECTRIC COST"
235 PRINT "6-DISPLAY ELECTRIC UNITS"
240 PRINT "7-RETURN TO COMMAND LIST #1"
250 INPUT"ENTER COMMAND BY NUMBER";R:IFR<1 OR
    R>7 THEN200
260 ON R GOSUB 340,350,360,370,380,390:GOTO270
265 GOTO60
270 CLS:PRINT"1-DISPLAY TO CRT"
280 PRINT"2-DISPLAY TO PRINTER"
290 PRINT"3-RETURN TO COMMAND LIST # 1"
300 INPUT"ENTER COMMAND BY NUMBER";R:IFR<1 OR
    R>3 THEN 270
330 ON R GOSUB 900,1200,335:GOTO60
335 GOTO60
340 C$="WATER COST":FORJ=1 TO N:K(J)=A(J):NEXT
    J:RETURN
350 C$="WATER UNITS":FORJ=1 TO N:K(J)=B(J):NEX
    TJ:RETURN
360 C$="GAS COST":FORJ=1 TO N:K(J)=C(J):NEXTJ:
    RETURN
370 C$="GAS UNITS":FORJ=1 TO N:K(J)=D(J):NEXTJ
    :RETURN
380 C$="ELECTRIC COST":FORJ=1 TO N:K(J)=E(J):N
    EXTJ:RETURN
390 C$="ELECTRIC UNITS":FORJ=1 TO N:K(J)=F(J):
    NEXTJ:RETURN
900 CLS:P=0:A$=CHR$(133):B$=CHR$(131):FORI=68
    TO 388 STEP32:PRINT@I,A$;:NEXTI
901 RESTORE
910 FORI=388 TO 412:PRINT@I,B$;:NEXTI
940 FORI=421 TO 443 STEP2:READE$:PRINT@I,E$;:N
    EXTI
950 DATA"J","F","M","A","M","J","J","A","S","O
    ","N","D"
960 FORJ=1 TO N
970 IFK(J)-P=>0 THEN P=K(J) ELSE P=P
```

```
980 NEXTJ
1020 PRINT@32,P;" ";C$
1030 FORJ=1 TO N
1040 IFP=0 AND K(J)=0 THEN 1110 ELSE 1050
1050 IFK(J)/P*10<1 THEN KK=1 ELSE KK=FIX(K(J)/P
     *10)
1060 T(J)=INT(36+((11-KK)*32))
1070 FORI=(T(J)+J*2) TO 388 STEP32
1080 IFT(J)=356 AND K(J)=0 THEN 1100 ELSE 1090
1090 PRINT@I,A$
1100 NEXTI:NEXTJ
1110 PRINT@453,"PRESS <ENTER> TO RETURN";GOSUB1
    120
1120 Z$="":R$*INKEY$:IFR$=Z$ THEN 1120
1130 RETURN
1200 CLS:PRINT"PRINTING GRAPHICS":P=0:L=0
1210 FORJ=1 TO N
1220 IFK(J)-P=>0 THEN P=K(J) ELSE P=P
1230 NEXTJ
1240 PRINT#-2,P;" ";C$;TAB(24)T$
1250 A$=LEFT$(C$,1)
1260 'FORL=1 TO 12
1270 IFP=0 THEN GOSUB 1380 ELSE 1280
1275 GOTO1350
1280 GOSUB1400
1340 'NEXTL
1350 PRINT#-2,STRING$(80,45)
1360 PRINT#-2,TAB(4)"JAN";TAB(8)"FEB";TAB(12)"M
    AR";TAB(16)"APR";TAB(20)"MAY";
1361 TAB(24)"JUN";TAB(28)"JUL";TAB(32)"AUG";TAB
    (36)"SEP";TAB(40)"OCT";
1362 TAB(44)"NOV";TAB(48)"DEC"
1370 RETURN
1380 FORL=1 TO 12
1385 PRINT#-2,CHR$(108)
1386 NEXTL
1390 RETURN
1400 FORL=1 TO 12
1405 PRINT#-2,CHR$(108
1410 FORJ=1 TO N
1420 GOSUB1460
1430 NEXTJ
1450 NEXTL
1455 RETURN
1460 IFINT((K(J)/P)*12)<=12-LTHEN PRINT#-2,TAB(
    4*J)"";ELSE PRINT#-2,TAB(4*J)A$;
1470 IFJ=>N THEN 1480 ELSE 1490
1480 PRINT#-2,CHR$(13);
1490 RETURN
```

A digitizer allows you to draw in free form on an "electric tablet" and communicate the results to your computer. It's similar to using a light pen, but more versatile.

$20 VIC Digitizer

Jeff Knapp
Charleston, WV

Have you ever looked at the commercially available digitizer tablets available for the TRS-80, PET, Apple, and other microcomputers and wished they were available in an affordable form for the VIC? Well, here's an inexpensive version that you can assemble in a few hours. If you don't have the parts in your junk box, it will probably cost less than $20 to buy all the parts new.

Take a look at Figure 1. This drawing shows the digitizer in its finished form. It consists of a base plate that becomes the stage for the materials to be digitized or a reference grid for input to the computer; a small box mounted on that contains the X-axis potentiometer (POTX); an arm extending from that pot that has the Y-axis pot (POTY) at its other end; and a second arm attached to the shaft of the Y-axis pot that has a hole drilled in the free end. This last arm becomes our cursor.

Construction is straightforward and simple. First, determine the desired size of your base plate. I recommend that it be at least 12" deep and 16" wide in order to allow for reasonable tolerances in the finished grid effective area. Choice of materials is up to you. I used Plexiglas plastic sheeting for mine. In the center of one long edge, mount one half of a mini-box or some other suitable container. It should be narrow enough that it doesn't take up too much space, and shallow enough that it is barely taller than the mounted potentiometer. About 2"x3"x1" will do for a 12"x16" base plate.

Next, construct the arms. They should not be too long, since the cursor should be able to be placed over every intended coordinate on the grid. Mine were 7" long each. You can fudge a lot on the measurements, as long as everything is secure and the arms reach every point on the grid you will later place on the stage. You may find that it will be necessary to counterweight the first arm because of the weight of the second arm. All the construction can be as fancy or as simple as you please, as long as the arms cover the grid area adequately.

Now for the electronics. Figure 2 shows the schematic diagram for the digitizer. As you can see, it's very simple. One caution, though: try to use potentiometers that have permanently mounted metal shafts. Some pots are made to accept interchangeable shafts, and these have tolerances too loose for our purposes. If the shaft wobbles in the pot, chances are that your arms will sag because the weight of the arms pulls the shaft to one side. Use 250K pots, linear taper, and a DP9S 9-pin socket for termination. If you can't find the socket at the local electronics supply shop, try a hobby store. They're often used in radio controlled models. The cable linking the digitizer to the VIC can be just about anything. I used ribbon cable, about three feet long.

Mount one pot in the center of the other half of the mini-box, and provide a hole in the side for the cable to exit and enter. Mount the other pot in one end of the first arm, using the hardware provided with the pot. Mount the other pot in one end of the first arm, using the hardware provided with the pot. Mount the free end of that arm to the pot in the box. Use any method you wish, even press-fit, but be sure that it is secure and that the arm does not rotate without rotating the shaft as well. Mount the other arm to the pot at the end of the first arm in a similar manner. If you have drilled a hole in the free end of the second arm, that becomes the cursor. If not, mark the arm with cross-hairs of some sort.

The Secret Is The Game Port

How does such a simple device work? Well, we have to depend on the VIC and some software for a lot of the job. The secret is that the VIC has two A-D (analog-to-digital) converters available at the game port. Although these are usually available as accessories for most microcomputers, the VIC has not one but *two* A-D converters built-in! Each generates a number between 0 and 255 based on the resistance of the two pots in our digitizer, or game paddles, or photocells, or any other variable resistance. The scale involved is one count for every 1K of resistance. Full scale would require a 255K pot, but ask your parts jobber for one and see the funny looks he gives you. 250K is as close as we can get. Plug the digitizer socket into the game port on the side of the VIC. Turn the VIC on and type in the following program:

```
10 PRINT"X=";PEEK(36872),"Y=";PEEK(36873)
20 GOTO 10
```

Run the program, and you will see X and Y values of the digitizer scroll up the screen. As you move the arms, watch the values change. In order to calibrate your digitzer, place a grid of your own construction, or some graph paper, on the base.

Adjust the positions of the pots in the arms so that when the cursor is pointing to the upper left coordinate of the grid, the X and Y values are 0. If the numbers on the screen get larger rather than smaller, then reverse the outside pin connection on the affected pot from one end of the pot to the other.

What can you do with it? That's up to your imagination, but how about putting a map of the U.S. on the grid and creating a program that asks a student to identify states by placing the cursor over them? You might put the alphabet at the bottom of the grid and let your program recognize the characters by the position of the cursor.

In the digitizer described, data is being continuously sent to the VIC, and it is up to your program to decide what is valid data and what is garbage. In order to help make the system more "intelligent," you might add a push-button to the tablet and connect it to the game port or the User port; then make your program wait for the button to be pushed before accepting data from the digitizer.

References:

6560/6561 Video Interface Chip, Commodore International, 950 Rittenhouse Road, Norristown, PA, 19403

Personal Computing on the VIC, Commodore International, 950 Rittenhouse Road, Norristown, PA, 19403

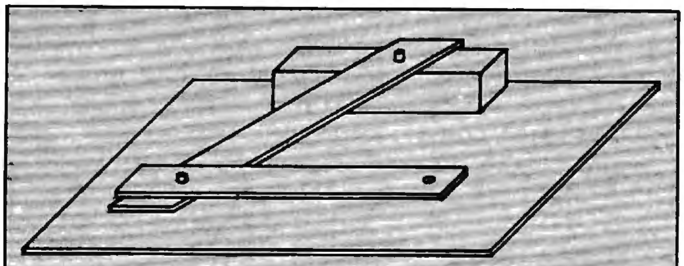


Figure 1.

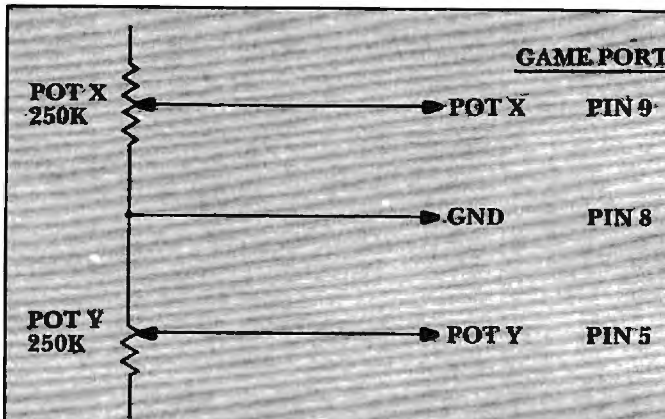


Figure 2.

There are many things you can do with disk files to clean up some problem with their contents. But what do you do when you have a problem right now, and need to take action on the spot?

On-The-Spot Commodore Disk Fixes

Jim Butterfield
Associate Editor

Dual Personality

Remember that Commodore machines have intelligent disks. When you have a file open for reading or writing, there are two systems that are keeping track of it: your computer and your disk. Normally they will have the same information and be working on the same file; but if they ever get out of step, watch out!

Let's trace a simple activity and see how the problem might arise. You have this program which reads file A and writes file B. When you say RUN on your computer, the two files are opened. Your computer knows about both of them, and so does the disk.

Everything goes smoothly for a while, and then the program stops for some reason. It might be a syntax error or some other problem. Now, the files are still open. The computer and the disk both know they are open, but the computer program is stopped, and the files won't close without help from you. You should spot this: glance at the disk and you'll see that the drive light is still on. You can fix up the open files. But first, let's detour a little.

The Problem

You can abandon file A, the one you are reading, without serious harm. You may not have gotten all the information from the file, but you can always do another RUN – the file won't be harmed.

The file you are writing, file B, is a different matter. The data you have sent to the file is *not* all on the disk yet. Some of it is being held in a buffer within the disk, and that buffer will be written only when it's full or when the file is closed. If you take the diskette out of the drive now, the information won't be there. You must CLOSE the file.

Compounding The Problem

All you need to do is to say CLOSE 1, CLOSE 2, or whatever the logical file numbers are, and the write file will be properly closed. If you have BASIC version 4.0, you may say DCLOSE. But sometimes we don't think.

When we get an error like Syntax or Subscript or whatever, our first instinct is to look at the bad line. So we say LIST and see a line such as CHOSE 1 instead of CLOSE 1. Still no problem. But when we decide to fix it...

Most programmers know that the moment they change a line of a BASIC program, the variables are scrapped. It's not hard to see that the identity of a live file is going to be scrapped, too. So if we correct that bad line – which is the natural and instinctive thing to do – the computer has lost all its file information.

Now we have a bigger problem. The disk still knows it's got a couple of open files, but BASIC has forgotten about it. The drive light is still on, but we can't CLOSE the files in BASIC any more. The job gets tougher.

Even though the computer has lost track of what files are live on disk, there's a way to get those files closed. Here's how it's done: when you close the Command Channel to disk (secondary address 15), the disk closes all files. So to close those files, we type: OPEN 1,8,15:CLOSE 1. We haven't done anything on the Command Channel, but the act of closing it causes the disk to wrap everything up.

From Bad To Worse

But suppose you didn't fix it. What then? If you take your disk out, or start a new file-handling program, or shut the power off, your write file can never be complete. The information in the buffer never got written, remember? Your write file is no good, and the next time you take a CATALOG you'll see two odd things. First, the file length for the bad file will be shown as zero. Second, there will be an asterisk by the file type, e.g., *SEQ.

The natural thing to do when you see this type of bad file is to SCRATCH it. A word of advice: don't. SCRATCHing an asterisked file may create a poisoned disk. You are planting a bad seed on your disk which may wreck an important file or program two or three months from now.

If you want to get rid of the file, perform a COLLECT (sometimes called a disk Verify or Validate). That will do the job, and do it correctly.

Looking Back

If you have doubts about any of your disks, you can always check them out with program Disk Checker (**COMPUTE!**, April 1982, #23). Start reading at the subheading Disk Test. The information you need starts there. If you invoke the program's Option 1, File Check, all files will be closely checked for any traces of "poison."

A series of seemingly natural actions on your part can lead to wrecking a file – or worse, endangering a disk. In case of program troubles, learn how to close your files and avoid trouble. ©

VIC Sticks

Jim Butterfield
Associate Editor

VIC joysticks have already been described by David Malmberg (*Home and Educational COMPUTING!*, Number 1). There's also a more formal description in the *Programmer's Reference Guide*. Let's pick up on joysticks one more time.

Collision With Keyboard

Try this. Hold the joystick over to the right and, while you're holding it, press the VIC number keys. You'll see that the odd numbers appear correctly on the screen, but the even digits are either missing or butchered. As soon as you release the joystick, the keyboard action returns to normal.

What's going on? In the interests of economy, Commodore has made one of the keyboard lines do "double duty" and test part of the joystick as well as perform its normal keyboard-checking functions. This is a two-way interference. We've seen that the joystick can interfere with the keyboard; in addition, the keyboard-servicing routines can make it impossible to check part of the joystick. The routines which read the keyboard are a special type called "interrupt" programs; this makes them hard to control.

Once you know the question, the answer isn't hard. To check the joystick completely, we must shut off part of the keyboard. If we need to keep the whole keyboard live, we must turn it back on again after checking the joystick.

We may shut down part of the keyboard with POKE 37154,127 and restore it with POKE 37154,255. We need to do this only to check the Right position of the joystick, which is done by looking at (PEEK(37152) AND 128).

Solving The Collision

What are our options? First, if we have a program that doesn't need the joystick's Right position, we can ignore the whole question.

If we have a program that doesn't need the keyboard, we can start with POKE 37154,127 as our first statement and restore the full keyboard only when the program ends. It won't matter that the keyboard is partly disabled during the program run. If the user/player stops the program rather than allowing it to end normally, however, he'll find his keyboard is acting badly. This isn't serious: the RUN/STOP-RESTORE key combination will fix everything up.

If we want to keep the keyboard live during play, each check of the Right position must include the whole set of three: disconnect part of the keyboard, check Right, reconnect keyboard. It will cost us a little more running time, but it's neater. It's not perfect, however: some keys will tend to hiccup if held down.

Machine language programs can solve everything, of course. They won't have a speed problem, so the keyboard can be quickly disabled and re-established. And the "hiccup" will go away if the interrupt is disabled during joystick checking; the interrupt routine won't jump in and be fooled during this check. Even in BASIC, however, you can do a competent job.

Difficult Diagonals

Joysticks are often inexact. You may think you are pushing Up, but you are slightly off true and the joystick might record both Up and Left.

The computer detects this, but your program must make a decision. If your program doesn't want diagonals, you must decide which of the two directions – say, Up and Left – is intended. It's easy to get the wrong one.

Directions are picked up as follows: UP is PEEK(37151) AND 4; DOWN is PEEK(37151) AND 8; LEFT is PEEK(37151) AND 16. The "fire" button is detected with PEEK(37151) AND 32, and RIGHT is checked as above, doing a partial keyboard disable and then working with PEEK(37152) AND 128. Each of these values becomes zero when the appropriate direction/button is activated.

You might write your program to check UP, then DOWN, then LEFT, etc., and to go to the appropriate action when you find an active position. If so, you'll miss the diagonals: UP/LEFT will exit on the UP condition and never check the LEFT, for example. This might be good for your particular game, but think of the human interface: the player might believe that he is pressing LEFT; the joystick is signalling LEFT and UP; and your program is reading only UP.

There's no absolute answer to this kind of question. Depending on the application, you can make certain choices. If you have on the screen a missile which is flying to the right, for example, you might choose to ignore all RIGHT/LEFT signals from the joystick and honor only UP/DOWN. Another approach is to design your game so as to use diagonals.

It's possible to write programs which "debounce" the joystick – that is, it must be returned to the center or rest position before a new signal will be accepted from it. This gives the effect of an impulse type of stick – action takes place only when the stick is moved.

A Simple Joystick Algorithm

One of the annoying things about joystick testing is that the input is logically inverted: the appropriate input is zero when activated, rather than zero when off. Although the information is the same either way, our minds don't like it. It seems more sensible to us to have bits turned on when the joystick is pushed; this allows us to extract combinations of bits with the logical AND function. A simple conversion statement which allows this is:

X=(NOT PEEK(37151))AND 60-((PEEK(37152)AND 128)=0)

Don't forget to POKE 37154 with 127 before doing this test, or the Right position won't be detected properly; and remember to POKE 37154 back to 255 after the test.

After executing the above statement, variable X will contain complete information about the joystick. If nothing is active, X will be zero. If we want to check a change in the joystick status, we can see if the value of X has changed since last time.

We may now detect the various control positions with the appropriate AND statements:

Fire Button	**–X AND 32**
Left	**–X AND 16**
Down	**–X AND 8**
Up	**–X AND 4**
Right	**–X AND 1**

In each case, the result of the AND will be zero if this position is not active. Combinations can be used: for example, if we are interested in only UP and DOWN at this moment, we could check X AND 12.

When coding this, use parentheses liberally around the AND statements. For example, to test for Left, code: IF (X AND 16)<>0 THEN ... It won't work properly otherwise.

For motion, we can extract the Left/Right and Up/Down components with coding such as:

H=SGN(X AND 1)–SGN(X AND 16)
V=SGN(X AND 8)–SGN(X AND 4)

This produces value for H and V as follows: 0 for no motion in this direction; +1 or -1 for motion in the appropriate direction.

Putting It All Together

The following simple program gathers together the joystick techniques we have discussed. It's a simple sketching program.

```
100 REM JOYSTICK PROGRAM
110 PRINT CHR$(147);CHR$(142) : REM
   CLEAR SCREEN
120 DATA 5,28,30,31,144,156,158,159
130 DIM C(7) : REM COLOURS
140 FOR J=0 TO 7:READ C(J):NEXT J
150 S=1:PRINT CHR$(C(S));
200 REM TEST JOYSTICK
210 POKE 37154,127
220 X=(NOT PEEK(37151))AND 60-((PEE
   K(37152)AND 128)=0)
230 POKE 37154,255 : REM  RESTORE K
   EYBOARD
240 IF (X AND 32)=0 GOTO 300 : REM
   NO BUTTON
250 IF B>0 GOTO 200 : REM DEBOUNCE
   BUTTON
260 B=1:S=S+1:IF S>7 THEN S=0
270 PRINT CHR$(C(S)); : REM CHANGE
   COLOUR
280 GOTO 200
300 B=0
310 H=SGN(X AND 16) - SGN(X AND 1)
320 V=SGN(X AND 4) - SGN(X AND 8)
330 PRINT CHR$(209);CHR$(17);CHR$(1
   7);CHR$(29);
340 FOR J=0 TO H+1:PRINT CHR$(157);
   :NEXT J
350 FOR J=0 TO V+1:PRINT CHR$(145);
   :NEXT J
360 GOTO 200
```

A few comments on the above coding. The Fire button is used to change color on the screen; the program debounces the button (using variable B) so that holding down the button does not cause a continuous color change.

Lines 310 and 320 compute reverse values of V and H compared to the algorithms given previously. In this case, we're computing an inverse activity – how many places to back the cursor up for a given position.

Lines 330 to 350 are rather "gimmicky"; we force the cursor right and down, and then count our way back to the desired position using cursor-left and cursor-up characters. The intent here is to illustrate the use of the V and H directional values. You may find other ways to achieve the same objective when you write your own programs.

The program prints the "ball" character, CHR$(209); you can switch to another character by making the appropriate change in line 330.

The joystick can indeed be interfaced with your program; all you need is to learn a few rules. You must set your own objectives as to how the joystick best interfaces with the user in your application. Once you have learned the mechanics, it's not hard to make everything work. ©

In this, the conclusion of a three-part series, several demonstration programs teach the concepts (and show off) the new Atari GTIA graphics chip. The GTIA costs nothing if your machine is still under warranty. If you have an older Atari, your nearest authorized center should be stocking it by now and will install it for about $60 according to Atari. If you just bought your computer. it's in there.

Atari Video Graphics And The New GTIA: *Part 3*

Craig Chamberlain
Birmingham, MI

Welcome back to our discussion of Atari playfield graphics and the exciting new GTIA chip. In Parts 1 and 2 I presented definitions of various terms related to graphics, explained the normal graphics modes, and then introduced the three new modes provided by the GTIA. Specifically, these new modes are:

MODE	DESCRIPTION
9	16 shades of one color
10	8 indirected colors
11	16 colors (one luminance)

Here are several programs in Atari BASIC to demonstrate how these new modes might be put to use. But first, let's tie up a few loose ends from the previous articles.

We used a standard method to show bit designations in the first parts of this article. If you are not familiar with this convention, here's how it works. Any given memory location or hardware address consists of one byte made up of eight binary units called bits. These bits are numbered zero to seven and are frequently shown as D0, D1, D7, etc. Individually, each bit can have two values, zero or one, but from the viewpoint of a byte, they take on quite different values known as "powers of two." For example, D3 means "two to the power of three," which also means "the number two used as a factor three times." Two times two times two is eight, so if we wanted to turn on only bit three in a given hardware register, we would POKE it with an eight. If we want to turn several bits on, we must add all the proper values together.

BIT	VALUE
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128

Mode 11 can be invoked by turning on bits six and seven of GPRIOR, location 623 (decimal). Thus we would POKE 623 with 64 + 128, which is 192. This brief explanation should help you deal with the memory locations and hardware registers described in the previous articles. Now for a review of the primary graphics statements of Atari BASIC and some special notes about the GTIA.

Graphics Statements

GRAPHICS aexp

This statement is the same as OPEN #6, 12 + 16,aexp, "S:", and tells the screen handler to open the screen to one of 12 modes. The number "aexp", which means "arithmetic expression," can range from zero to 11. Characteristics of these modes are explained in chapter nine of the *Atari BASIC Reference Manual.*

Some modes allow split screen configurations, which means that a text window appears at the bottom of the screen. Of course, mode zero does not allow a text window because mode zero is the text mode, although you can experiment with POKE 703,4. Modes one through eight do support text windows, and the only way to get a full screen (no text window) in one of these modes is to add 16 to aexp in the Graphics statement. When using a full screen mode, Atari BASIC forces a mode zero if it has to print normal text. It is impossible to use these full screens in the immediate programming mode because the "READY" prompt forces the mode zero screen.

Due to technical reasons explained last month, modes 9, 10, and 11 do not normally allow text windows. You can fool the operating system into giving you one of these modes with a text window by asking for mode 8 and then doing a couple of POKE statements, like this:

MODE	POKES
9	POKE 87,9: POKE 623,64
10	POKE 87,10: POKE 623,128
11	POKE 87,11: POKE 623,192

Location 87, known as DINDEX, tells the operating system the current mode and is used in the computation of row and column addresses for plotting, so any number from nine to eleven will give the same results. Unfortunately, the text win-

dow obtained by this method looks weird. The only way to get a real text window is to use a display list interrupt, discussed later.

If you add 32 to aexp, the screen will not be cleared when the new mode is requested.

Finally, the CTIA and GTIA support five other modes which the operating system does not recognize. They are the eight by ten matrix character version of mode zero, the multi-color text character modes, and the single scan line versions of modes six and seven, for 160 by 192 plotting in one or three colors. The only way to access these modes is to write a custom display list, which has been discussed in previous **COMPUTE!** articles.

COLOR aexp
This specifies the playfield that will be used for PLOT and DRAWTO statements, until changed by another COLOR statement. It does not in any way change any of the color/luminance registers for the various playfields! The range of aexp depends on the number of different playfields available in the current graphics mode. This still holds true for the new GTIA modes. For example, a COLOR 2 in mode 9 means that future plotted points will be rather dark, whereas bright lines will be drawn after a COLOR 12.

In mode 11, COLOR 5 chooses a purple color, as indicated by the chart in part one of this article. For all modes, COLOR 0 (zero) is the background or "erasing" color. Normally, the operating system wants you to specify the playfield each time you write to the screen, but Atari BASIC automatically tells the operating system which playfield you have chosen every time you use PLOT or DRAWTO. Incidentally, the data part of the COLOR statement is stored in memory location 200 (decimal), but I would not recommend using that.

One other note. To be technically accurate, COLOR 1 corresponds to playfield zero, COLOR 2 means playfield one, and so on.

POSITION aexp1,aexp2
This statement moves the graphics cursor to the location on the screen designated by the two numbers, according to the Cartesian coordinate system. No range checking is done.

PLOT aexp1,aexp2
This is the same as POSITION aexp1,aexp2: PUT #6,color where "color" is the playfield type chosen by the most recent COLOR statement. You will get an error number 141 if you try to PLOT outside the bounds of the screen. All three new modes have resolution of 80 by 192.

DRAWTO aexp1, aexp2
Essentially, this is the same as PLOT except that a line is drawn from the most recently plotted point to the new point indicated by aexp1 and aexp2. You can also do this with an XIO 17,#6,0,0,"S:". See page 54 of the *Atari BASIC Reference Manual* to see how XIO 18 can be used to fill areas with a playfield.

LOCATE aexp1,aexp2,avar
I don't know why, but nobody seems to know about this statement. It could be considered the reverse of PLOT. Instead of putting a playfield point at a certain location on the screen, this statement returns to you, in the arithmetic variable "avar", the playfield number of the point at location aexp1,aexp2. This playfield number will be the same as the value of COLOR that was in effect when the point was plotted. LOCATE is actually quite handy. It is very useful in games where collisions occur between differently colored players, but it has many other applications, too. LOCATE is the same as POSITION aexp1,aexp2: GET #6,avar.

SETCOLOR aexp1,aexp2,aexp3
This is the statement which changes the color and luminance of a playfield register. The number aexp1 designates which playfield register is being changed, and is related to the number in the COLOR statement in the following way:

COLOR	SETCOLOR (playfield number)
1	0
2	1 (also used for luminance in modes zero and eight)
3	2
–	3 (used only in four color text modes one and two)
0	4 (background, or border in modes zero and eight)

The value for aexp2 is a number from zero to 15 which specifies one of 16 colors. See the chart in part one of this article, or on page 50 of the *Atari BASIC Reference Manual*, to find which numbers go with which colors. The luminance is chosen by aexp3, which can range from zero to 15, but only eight true luminances can be selected. A value of zero here gives the same luminance as one, two the same as three, and so on. The larger the number, the greater the luminance.

Remember that modes 9 and 11 do not use color indirection or the playfield registers, so SETCOLOR has little use in these modes. It can be used to set the background color/luminance in these two modes, but that's about it.

Now for mode 10. This mode uses the player/missile color/luminance registers, which cannot be accessed using SETCOLOR. An equivalent POKE statement must be used instead. The location to POKE is similar to the aexp1 of SETCOLOR. The shadows of the playfield registers run from locations 708 (decimal) to 712. The value to POKE contains the color and luminance information and is a combination of aexp2 and aexp3. This value is

the sum of 16 times the color number, plus the luminance. In effect, SETCOLOR X,Y,Z will do the same as POKE 708+X, 16*Y+Z. If you want to change the player/missile color/luminance registers, which run from locations 704 to 707, use the same procedure of multiplying the color by 16 and then adding the luminance. Refer to part one of this article for a chart that tells which COLOR numbers match with which registers.

Some Lively Demos

Now comes the good part, where the action is! If your Atari computer has a GTIA in it, here are some programs to show off the talents of this remarkable chip.

How to put 16 colors on the screen? It could be done in one line:

```
GRAPHICS 11: FOR K=0 TO 79: COLOR K: PLOT
K,0: DRAWTO K,191: NEXT K: FOR K=0 TO 0 STEP 0:
NEXT K
```

The endless loop is necessary to prevent Atari BASIC from printing a "READY" prompt which would force mode zero. To make the vertical color bands wider, change the COLOR K to COLOR K/5. To see 16 shades, change the GRAPHICS 11 to GRAPHICS 9.

A fancier way of showing 16 shades is found in Program 1. After drawing the shades, the background color is rotated through all 16 colors.

Program 3 randomly draws lines in 16 colors. You can make these colors appear darker or more pastel by changing the luminance of the background. Please note that mode 11 is the only mode in which the background is set by the operating system to a luminance of six. All other modes have backgrounds of color/luminance zero (black).

Program 2 demonstrates the color indirection capabilities of mode 10. Location 20 is the lowest counter of the realtime clock, so it is always changing. Continuously PEEKing this location and POKEing the value into a color register gives a nice "throbbing" color spectrum effect.

How about a doodling program that lets you draw in 16 colors using the joystick? Program 4 does this in only three lines of Atari BASIC code! Press the joystick trigger to change colors.

Program 5 is a really beautiful color kaleidoscope generator, considering it is only four lines long. It's not something you will spend hours watching, but it can produce some nice pictures. Try changing the K=I+J in the second line to K=I for a different picture. Or you can reverse the direction of the main loops, as in FOR I=31 TO 1 STEP -1. If you change the J loop (note that it starts at zero, FOR J=31 TO 0 STEP -1), you will also want to change the H loop (FOR H=1 TO 3 STEP 1).

To show 256 colors on the screen all at once, use Program 6. This program does not show the colors. Rather, it produces a single line which you can ENTER from disk or cassette. This single line performs all the magic. What is also neat about this program is that when you ENTER the line in, the program already in memory is untouched. If you examine Program 6, you will see that it writes a line to the chosen device, but the line has no line number in front of it. When you ENTER this line, it is the same as typing it in the immediate mode. When Program 6 asks for a device specification, respond with C: for cassette or D:filename for disk.

I included the assembly source code and Atari BASIC installation routine for a display list interrupt service routine (Program 8) that creates a text window on modes 9, 10, or 11. An interrupt is requested at the last mode line of the graphics mode portion of the screen. The service routine takes the value of GPRIOR, sets the GTIA mode select bits to zero, and stores the result in PRIOR, the hardware register. PRIOR gets reset to the value of GPRIOR as part of the vertical blank service routine. The routine also stores a zero into the background hardware register. This was necessary to fix a conflict in mode 11. Setting the luminance in 712 also changes the border around the text window. But this "fix" created another problem in mode 10. For mode 10, change the fourteenth DATA element, which normally should be a zero, to be the same as the number POKEd into 704.

The service routine is written using relocatable code, so you can put the routine anyplace in memory simply by changing the assignment of ADDRES in the second line. It is currently set to start at the beginning of page six. The first three lines of Program 7 actually install the routine. The fourth line just draws a picture for purposes of demonstration. Notice the luminance change of the colors when 712 is POKEd.

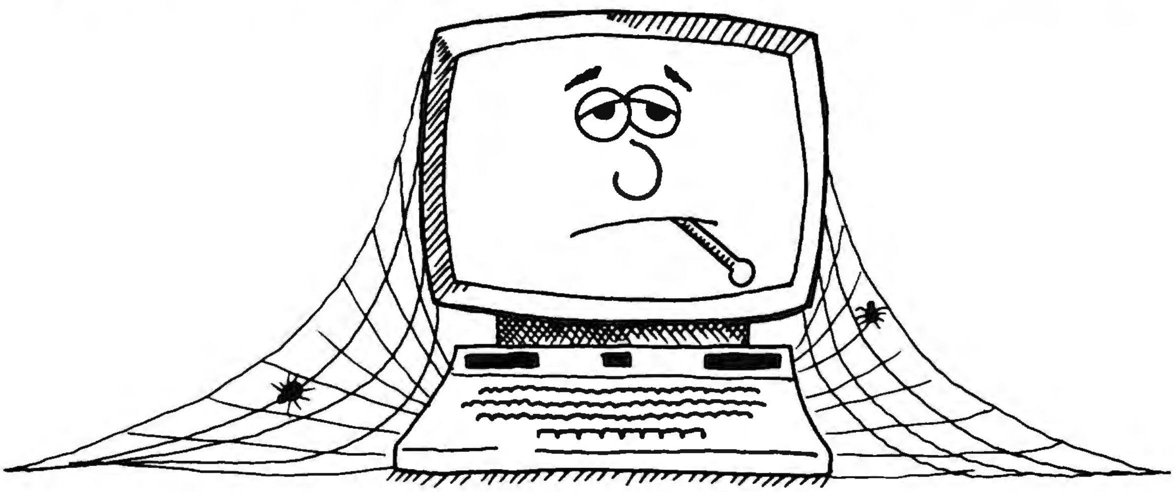
My routine shares the problem of many display list interrupt service routines; keyboard clicks can affect the display. Obviously this routine is suitable only for programs that do not accept keyboard input (use the joystick or PEEK the hardware keycode register 764 directly) or use serial I/O (the vertical blank routine is abbreviated and PRIOR does not get reset).

Program 1.

```
10 GRAPHICS 9:FOR K=1 TO 10 STEP 2:FOR J
   =0 TO 15:COLOR J:PLOT 0,K*16+J+1:DRAW
   TO 79,K*16+J+1
20 PLOT 0,K*16-J:DRAWTO 79,K*16-J:NEXT J
   :NEXT K
30 FOR K=1 TO 255 STEP 16:POKE 712,K:FOR
    J=1 TO 500:NEXT J:NEXT K:GOTO 30
```

QUICKSOFT™

CURES THE #1 TERMINAL ILLNESS:



S.D.S.

(SLOW DELIVERY SYNDROME)

QUICKSOFT'S cure doesn't come in a bottle, but it does come in handy when your computer needs software delivered promptly.

24 HOUR SHIPPING: QUICKSOFT is the software-house with a difference: **SPEED!!** When you place a phone order with **QUICKSOFT we guarantee that it will be shipped within twenty-four hours.***

WIDE SELECTION: QUICKSOFT carries thousands of software disks, tapes and cartridges for every purpose. We have business systems to cure those overflowing files, educational disks to relieve the worry of a D in chemistry, and adventure games to aid a case of the blues.

QUICKSOFT HAS THEM ALL, and right now we're having specials on these items from our "most wanted" list:

Wordrace (D)	**$19.95**
Bug Attack (T,D)	**23.95**
Canyon Climbers (T,D)	**23.95**
Protector (T,D)	**23.95**
Shooting Arcade (T,D)	**23.95**
Tumble Bugs (D)	**23.95**
Pathfinder (D)	**27.95**
Ulysses and the Golden Fleece (D)	**27.95**
Pac-man (cart)	**35.95**
Text Wizard (D)	**79.95**

Prices effective through October 16, 1982.

FREE DELIVERY: QUICKSOFT offers free delivery on *all* orders of $50.00 or more. We not only ship within 24 hours but we also pay the freight! (For orders of less than $50 add $1.50 for shipping and handling.)

CALL TOLL FREE ANYTIME. QUICKSOFT'S toll free number is available anytime of the day or night. For placing an order or for requesting our **FREE CATALOG** simply dial:

1-800-547-8009

IN OREGON CALL 1-342-1298

Please specify for Apple or Atari computer on your order.

***GUARANTEE: QUICKSOFT** guarantees shipment within 24 hours on each phone order. If the item ordered is not readily available and cannot be shipped on time, you will receive an additional 5% OFF the regular **QUICKSOFT** price.

VISA-MASTERCHARGE and C.O.D. Welcome

QUICKSOFT™

Department 3, P.O. Box 10854, Eugene, OR 97440

Program 2.

```
10 GRAPHICS 10:FOR K=705 TO 712:POKE K,1
   2:NEXT K:FOR K=0 TO 79:COLOR (K+4)/10
   :PLOT K,0:DRAWTO K,191:NEXT K
20 FOR K=704 TO 712:FOR J=1 TO 300:POKE
   K,PEEK(20):NEXT J:NEXT K:GOTO 20
```

Program 3.

```
10 GRAPHICS 11:FOR K=1 TO 124:COLOR K:DR
   AWTO RND(1)*79,RND(1)*191:NEXT K:GOTO
    10
```

Program 4.

```
10 GRAPHICS 11:DIM SX(15),SY(15):FOR K=5
    TO 14:READ X,Y:SX(K)=X:SY(K)=Y:NEXT
   K:X=40:Y=96:COLOR 1
20 PLOT X,Y:X=X+SX(STICK(0)):X=X+(X<0)-(
   X>79):Y=Y+SY(STICK(0)):Y=Y+(Y<0)-(Y>1
   91):IF STRIG(0) THEN 20
30 C=C+1-15*(C=15):COLOR C:GOTO 20:DATA
   1,1,1,-1,1,0,0,0,-1,1,-1,-1,-1,0,0,0,
   0,1,0,-1
```

Program 5.

```
10 GRAPHICS 10:FOR I=705 TO 712:POKE I,P
   EEK(53770):NEXT I:FOR I=1 TO 31 STEP
   1:C=C+1-9*(C=8)
20 POKE 704+C,PEEK(20):FOR J=0 TO 31 STE
   P 1:COLOR INT(RND(1)*15)+1:K=I+J:J3=J
   *3:K3=K*3:J8=J+8:J71=71-J
30 PLOT K+7,J3:DRAWTO K+7,191-J3:PLOT 72
   -K,J3:DRAWTO 72-K,191-J3:FOR H=3 TO 1
    STEP -1
40 PLOT J8,191+H-K3:DRAWTO J71,191+H-K3:
   PLOT J8,K3-H:DRAWTO J71,K3-H:NEXT H:N
   EXT J:NEXT I:POKE 77,0:GOTO 10
```

Program 6.

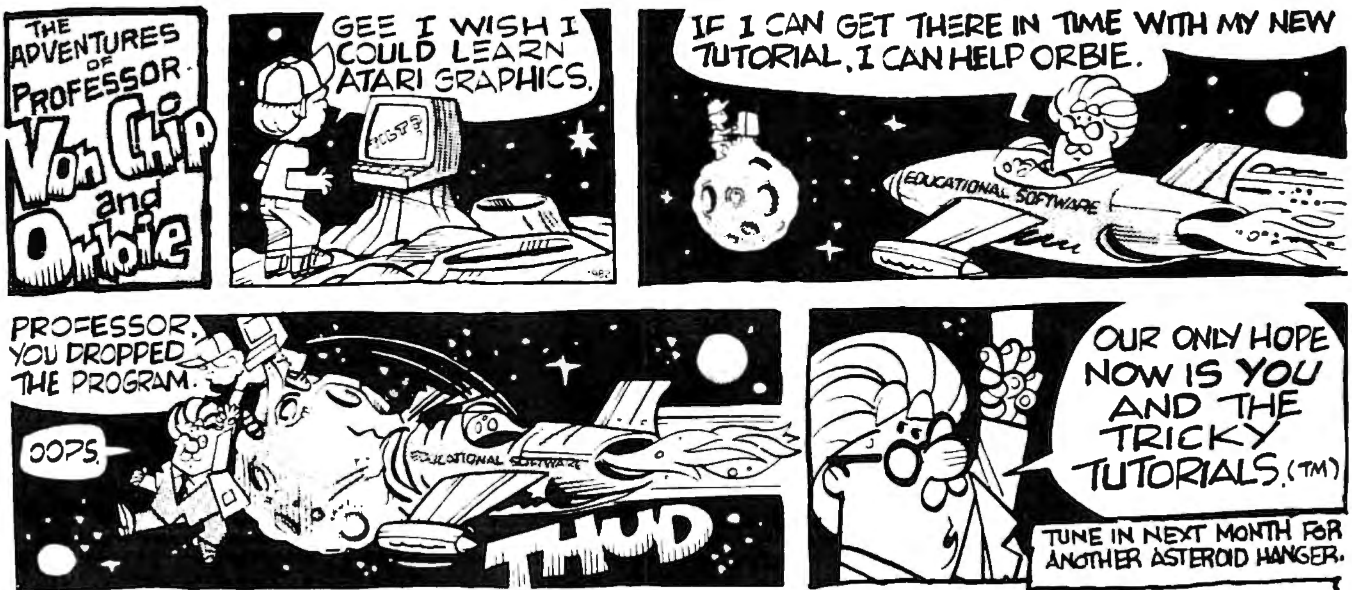
```
100 IF PEEK(87) THEN GRAPHICS 0
105 ? CHR$(125):? "GTIA DEMONSTRATION":?
     "by Tom Giese  4/15/82":?
110 ? "This program creates an ATASCII f
    ile"
120 ? "for ATARI BASIC.  The file consis
    ts"
130 ? "of one line which will produce tw
    o"
140 ? "hundred fifty six colors on your"
150 ? "screen if you have a GTIA install
    ed.":?
170 DIM S$(120):? "Please enter device s
    pecification."
180 INPUT S$:IF S$="" THEN 180
190 ? :TRAP 260:OPEN #1,8,0,S$
200 ? #1;"GR.9:F.K=0TO79:C.K/5:PL.K,0:DR
    .K,191:N.K:K=USR(ADR(";
210 PUT #1,34:FOR K=1 TO 15:READ P:PUT #
    1,P:NEXT K:PUT #1,34
220 DATA 173,11,212,10,229,20,41,240,141
    ,26,208,208,243,240,241
230 ? #1;"))":CLOSE #1:? "File has been
    written."
245 POSITION 2,19:? "ENTER ";CHR$(34);S$
    ;CHR$(34)
250 POSITION 2,15:? "Now press the RETUR
    N key if"
255 ? "you want to ENTER the file.":NEW
260 STATUS #1,P:? "I/O ERROR ";P:END
```

Program 7.

```
10 POKE 54286,0:GRAPHICS 8:POKE 87,11:PO
   KE 623,192:POKE PEEK(560)+256*PEEK(56
   1)+166,143
20 ADDRES=1536:POKE 54286,64:FOR K=0 TO
   18:READ P:POKE ADDRES+K,P:NEXT K:P=IN
   T(ADDRES/256):POKE 513,P
30 POKE 512,ADDRES-256*P:POKE 54286,192:
   DATA 72,173,111,2,41,63,141,10,212,14
   1,27,208,169,0,141,26,208,104,64
40 FOR K=0 TO 159:COLOR K/10:PLOT 0,K:DR
   AWTO 79,K:NEXT K:POKE 712,6:STOP
```

Program 8.

```
0000          10          .PAGE
              11 ;
              12 ;necessary operating system and
hardware equates
              13 ;
026F          14 GPRIOR    =     $026F
;GTIA priority control (shadow)
D01A          15 COLBK     =     $D01A
;background color register
D01B          16 PRIOR     =     $D01B
;GTIA priority control (hardware)
D40A          17 WSYNC     =     $D40A
;horizontal blank synchronization
              18 ;
              19 ;
0000          20          *=     $0600
              21 ;
              22 ;this service routine for the
display list interrupt
              23 ;can be placed anywhere in RAM,
and was placed on page six
              24 ;only for purposes of
demonstration
              25 ;
              26 ;begin interrupt service routine
code
              27 ;
              28 ;save contents of accumulator
              29 ;
0600 48       30          PHA
              31 ;
              32 ;get the multicolor player, fifth
player, and priority bits
              33 ;
0601 AD6F02   34          LDA   GPRIOR
              35 ;
              36 ;force the GTIA mode select bits
to zero but save the other bits
0604 293F     37          AND   #$3F
              38 ;
              39 ;wait until next scan line for a
nice clean change
              40 ;
0606 8D0AD4   41          STA   WSYNC
              42 ;
              43 ;change hardware register until
VBLANK
              44 ;
0609 8D1BD0   45          STA   PRIOR
              46 ;
              47 ;reset COLOR4 to zero (for modes
9 and 11)
060C A900     48          LDA   #$00
060E 8D1AD0   49          STA   COLBK
              50 ;
```



TRICKY TUTORIALS (tm)

There are many things that the ATARI computers can do either better, or easier than other small computers. The following series of programs is designed for anyone who is at least familiar with BASIC programming What each tutorial offers is similar to an extensive magazine article with all discussion in as simple language as possible, plus you get MANY examples already typed in and running. The instruction manuals range from 10 to 50 pages, and some tutorials fill up a complete tape or disk. There is little overlap in what is taught, so anyone wanting to know all they can should buy them all (my banker thanks you) ATARI buys these from us to use in training their own people! Rave reviews have been published in ANTIC, ANALOG, CREATIVE COMPUTING, and even INFOWORLD You trust INFOWORLD, don't you?

TT #1: DISPLAY LISTS—This program teaches you how to alter the program in the ATARI that controls the format of the screen Normally, when you say "Graphics 8", the machine responds with a large Graphics 8 area at the top of the screen and a small text area at the bottom. Now, you will be able to mix various Graphics modes on the screen at the same time. The program does all of the difficult things (like counting scan lines) You will quickly be able to use the subroutines included in your own programs
16K Tape or 24K Disk **$19.95**

TT #2: HORIZONTAL/VERTICAL SCROLLING—The information you put on the screen, either GRAPHICS or TEXT, can be moved up, down, sideways, or diagonally. We provide the basic methods and leave the rest up to your skill and imagination Includes 18 examples to get you started, with several using a small machine language subroutine for smoothness.
16K Tape or 24K Disk **$19.95**

TT #3: PAGE FLIPPING—Now you don't have to redraw the screen every time you change the picture or text. You will learn how to have the computer draw the next screen you want to see while you are still looking at the previous screen, then flip to it instantly. You won't see it being drawn, so a complicated picture can seem to just appear. Depending on your memory size and which graphics or text modes you are using, you can instantly look at up to 50 pages. The basic method takes only 9 lines and the usefulness is infinite
16K Tape or 24K Disk. **$19.95**

TT #4: BASICS OF ANIMATION—This program shows you how to animate simple shapes (with some sound) using the PRINT and PLOT commands, and it also has a nice little PLAYER/MISSILE GRAPHICS game you can learn from The P/M example is explained and will get you started on this complicated subject (more fully explained in TT #5) This would be an excellent way to start making your programs come alive on the screen with movement! Recommended for beginning users.
16K Tape or 24K Disk **$19.95**

TT #5: PLAYER/MISSILE GRAPHICS—Learn to write your own games and other animated applications! The tutorial begins with many small examples that compliment the 50 page manual, then gradually builds up to a complete game where everything you need to know is fully explained Also included are two machine language utilities that you can use to animate Players with from BASIC. Next we include two of the best editors currently available, one for editing playfield shapes (backgrounds); and one to edit your players, and all in glorious Technicolor!! Everything except the two editors run in 16K Tape or 32K Disk. **$29.95**

TT #6: SOUND AND MUSIC—Unless you have spent many years experimenting with the four voice channels, you will learn a lot from this one! Learn to play standard notes, chords, and whole songs using some simple "tricks". One of the nicest parts are the examples of special sound effects that you can refer to whenever you need a sound for a program or to impress a friend. This program will be of interest to all ages and levels of experience!
16K Tape or 24K Disk. **$19.95**

SPECIAL DISCOUNT

Order the first six tutorials in a 3-ring binder for $99.95, a $30.00 savings!

TT #7: DOS UTILITIES—We at Educational Software have been shocked by some of the prices others are charging to offer you small utilities to help in the use of your Disk Drive. We now offer you all of the following plus explanation as to how each was written, and how to use them: A UNIQUE MENU PROGRAM, AN AUTORUN.SYS BUILDER, DISK INSPECTOR (LOOK AT SECTORS), DISK JACKET PRINTER, AUTOMATIC FORMATTER, RECORD SAVE AND LOAD UTILITY.
32K Disk Only. **$29.95**

MASTER MEMORY MAP (tm)

This book is the most valuable source of information for your ATARI you can buy It starts out by explaining how to PEEK and POKE values into memory, so that even new computer owners can use many of these "Tricks". Then you are given 32 pages of the memory locations that are the most useful, along with hints on how to use many of the locations. Finally, it includes hints on problems you may be having with the computer and discusses the new Graphics modes 9 to 11 Even ATARI buys this book from us! **$6.95**

CONTACT YOUR LOCAL DEALER or ORDER BELOW

We have other fine programs for ATARI computers.
Write for a catalog.
Send us your programs to sell too!

USER SUBMITTED PROGRAMS

MINI-DATABASE/DIALER—stores and edits up to 8 lines of information such as names & addresses, phone numbers, messages, inventories, or anything you want. It has the usual sort, search, and print options, but it also has an unusual feature: If your file includes phone numbers and your phone company allows touch-tone phone signals, the program will DIAL THE PHONE NUMBER FOR YOU! 16K Tape or 24K Disk. **$24.95**

THE GRAPHICS MACHINE—allows the ATARI to act like more expensive graphics computers using simple commands like line, box, circle, polygon, fill, and savescreen to get a high resolution picture you can save on disk in only five seconds! Many more features! 48K Disk Only. **$19.95**

BOB'S BUSINESS—14 small business type programs accessed from a common menu 16K Tape or 32K disk **$14.95**

MINI WORDPROCESSOR—A simple text editor to write, save, and print several pages at a time. 32K Tape or Disk **$19.95**

KID'S #1—Includes a MATH QUIZ, a children's TREASURE HUNT, and a DIALOGUE program. 16K Tape or 24K Disk 3 for . **$14.95**

KID'S #2—SPELLING BEE, WORD SCRAMBLE, and TOUCH 16K Tape or 24K Disk. 3 Educational games for **$14.95**

PLAYER PIANO—Turns your keyboard into a mini-piano and more 24K Tape or 32K Disk. **$14.95**

GRAPHIC SYMBOL LABELS—for your keyboard to remind you of the built-in Graphics symbols. 2 complete sets for **$2.95**

OUR NEWEST PROGRAMS

DOG DAZE—Two cute little doggies race for the fire hydrants, shoot their bones, and just have a lot of fun! A fast action program for all ages 8K Tape or 16K Disk, in machine language. **$16.95**

*** OUR BEST GAME ***

SPACE GAMES—Our family is being attacked by ALIENS, and only you can save us. A comic book manual will guide you through three games that test your ability in space skills. Includes ALIENS, SURVIVE, and ROBOT ATTACK, and is for all ages. The first two games require 16K for Tape. The last game and all Disk users need 32K **$24.95**

MATHS FOR FUN—Another ENGLISH import teaching basic math skills. Very colorful and enjoyable to use. For ages 5 to 16. 16K Tape or 24K Disk. **$19.95**

MARATHON—This is a unique math quiz for one or two players You are in a race to move your runner across the screen first! There are four levels of play with five modes of operation for each. The game uses joysticks for all input, so play is easy for young children This wonderful learning tool is imported from ENGLAND for your learning pleasure. Your kids will never even notice they are playing an EDUCATIONAL program. 16K Tape or 24K Disk. **$19.95**

GRADE BOOK—This nice record keeper will maintain a file of 35 students' test scores along with comments. Up to 7 subjects are allowed 24K Tape or 32K Disk. A printer is optional. **$24.95**

To order COD, VISA or MasterCard call — **(408) 476-4901**

By mail include $3.00 postage (USA only) or $1.50 Memory Map only—
California residents add 6.5% TAX.—Specify Tape or disk.

EDUCATIONAL SOFTWARE inc.
5425 JIGGER DR.
SOQUEL,. CA 95073

```
               ;restore accumulator
               ;
0611 68                PLA
               ;
               ;return from the display list
interrupt
               ;
0612 40                RTI
               ;
               ;end of interrupt service routine
               ;

GTIADLI 4/30/82 by Craig Chamberlain

=026F GPRIOR        =D01A COLBK         =D01B
PRIOR         =D40A WSYNC
```

©

Type in these graphics demonstrations and see the startling displays made possible with the new GTIA chip.

Atari GTIA: An Illustrated Overview

Louis and Helen Markoya
Shelton, CN

Have you ever seen computer-generated graphic displays that are truly 3-D? The ones of landscapes or the ones with molecular structures? Have you ever wished you could generate similar graphics on your own machine? If so, and you own a 400 or 800, you're halfway there. The new GTIA chip allows for three more high resolution graphic modes.

These three modes are far from ordinary. They are called from BASIC by typing Graphics 9, 10 or 11. They all offer the same screen resolution, 80 Horizontal x 192 Vertical, but different color selection.

Graphics 9 offers *16* shades of any of the 16 colors, thus raising the machine's color capabilities to *256*. Graphics 10 offers the programmer a choice of nine of any of the 128 colors normally offered by the Atari, and Graphics 11 gives the ability to present 16 different colors in any of the eight luminances (shades).

For those of us who had the machine before this new addition, the authorized Atari service center nearest you should now be stocking this part. If your machine is under warranty, replacing the chip is free. If not, the chip can be bought for a reasonable fee from your dealer. If you are so inclined, you can install the chip yourself, but you have to tear down your unit to the CPU Board. This board is under the aluminum housing that covers most of the mother board of either the 400 or 800. The disassembly is not difficult if you take your time and are cautious. The CPU Board is easy to identify by the large chips on it. The 6502, ANTIC and CTIA are the large 40-pin chips on this board. GTIA replaces CTIA. One word of warning. These chips are not placed as pictured in the *Hardware Manual.* The easiest way to identify CTIA is that it has the same manufacturers' stamp on it as the GTIA. Authorized service centers will make the swap for you for an additional fee. Either way the cost is well worth it.

GTIA is Atari's new television Interface Chip. It is completely compatible with the hardware and software previously available. The only problem arises when software relying on the GTIA Modes is run on a computer without this chip. Something will go to the screen, but not the desired effect.

GTIA is controlled for the most part by ANTIC, a microprocessor dedicated to the screen display. The GTIA processes digital commands from ANTIC or the 6502 (in the case of an interrupt) into the signal that goes to the television. GTIA also handles the tasks of Color, Player/Missile Graphics, and Collision Detection.

GTIA adds powerful capabilities in Graphics Modes 9, 10, and 11. All modes are extensions of Graphics Mode 8 + 16, ANTIC Mode 15. The display list remains the same, and the new modes are selected by the Priority Register. This Operating System Shadow Register, called PRIOR, is located at decimal 623, Hex 26F. Bits 6 and 7 control the GTIA modes. When both are zero, GTIA works exactly the same as CTIA. When only bit 6 is set, Graphics 9 is called; when only bit 7 is set, Graphics 10 is called, and when both bits 6 and 7 are set, Graphics 11 is called.

Graphics 9

Setting bit 6 of PRIOR produces Graphics 9, giving 16 luminances of one color. ANTIC provides the pixel data, and the background register, 712, is used to select your color (POKE 712, Color * 16 or SETCOLOR 4,Color,0). Each screen byte is broken in half for screen formatting. A display block is four pixels across by one pixel down. Each four bits represents 16 color choices. The number you choose (0-15) in your color statement equates the luminance value you wish to use. Here's a simple BASIC program used to demonstrate this:

```
10 GRAPHICS 9:REM GRAPHICS MODE 9 (16 SH
   ADES OF ONE COLOR)
20 SETCOLOR 4,6,0:REM SET BACKGROUND REG
   ISTER TO COLOR DESIRED (PURPLE)
30 FOR I=0 TO 15:REM SET UP VARIABLE FOR
    BOTH COLOR (SHADE) AND POSITION
40 COLOR I:REM SHADE OF COLOR
50 PLOT I,0:REM PLOT FROM UPPER LEFT COR
   NER
60 DRAWTO I,191:REM DRAWTO LOWER LEFT CO
   RNER
70 NEXT I:REM NEXT SHADE AND NEXT LINE
80 GOTO 80:REM HOLD SCREEN
```

The wide choice of luminances or shades available here will be particularly useful for shading objects to give the impression of bas-relief or the third dimension. With some background in perspective and lighting, a person could create scenes with a great illusion of depth, realistic or contrived.

Graphics 10

Graphics 10 is called when bit 7 of PRIOR is set to one and bit 6 to zero. This mode utilizes all nine of the Atari's Color Registers found at decimal 704-712 (hex 2C0 through 2C8). Any nine of the 128 colors normally available to your computer could be used in this mode by simply POKEing the desired color (remember, 16 * Color+luminance) into each register or POKEing the desired color into the Player/Missile registers (704 through 707), using SETCOLOR statements for the playfield and background registers.

Color 0 represents the background and is located at decimal register 704, Colors (for color statements) 1-8 follow in order from 705-712. The big advantage to Mode 10 is that any of the colors you choose can be changed independently of the others. For example, once a scene is created, you could change the color of the sky from dark to light blue very easily (FOR I = 128 TO 144: POKE 704,I: NEXT I). This will rotate the background color smoothly through its eight shades. You may wish to add a loop to delay the color change. Playfield or Player/Missile Colors could be changed at any time. Also, special effects and animation could be achieved by rotating the values in all these registers.

The following program draws a border around the screen in eight colors (first register is used for background) and then rotates the colors to give a special effect:

```
10 GRAPHICS 10
20 POKE 704,96:REM SETS BACKGROUND (COLO
   R 0, COLPMO) TO DARK PURPLE
30 POKE 705,22:REM SETS COLOR 1, COLPM1
   TO YELLOW
40 POKE 706,38:REM SETS COLOR 2, COLPM2
   TO YELLOW ORANGE
50 POKE 707,54:REM SETS COLOR 3, COLPM3
   TO ORANGE
60 POKE 708,70:REM SETS COLOR 4, COLPFO
   TO RED
70 POKE 709,86:REM SETS COLOR 5, COLPF1
   TO PURPLE
80 POKE 710,104:REM SETS COLOR 6, COLPF2
    TO BLUE
90 POKE 711,120:REM SETS COLOR 7, COLPF3
    TO BLUE GREEN
100 POKE 712,180:REM SETS COLOR 8, COLPF
    4 TO GREEN
110 FOR I=1 TO 64:REM SETS UP VARIABLE F
    OR COLOR AND POSITION
120 C=C*(C<8)+1:COLOR C:REM CHANGES COLO
    R VALUE
130 PLOT I,I:REM START AT LEFT HAND CORN
    ER
140 DRAWTO I,191-I:REM DRAWTO BOTTOM LEF
    T CORNER
150 DRAWTO 79-I,191-I:REM DR. BOTTOM RIG
    HT CORNER
160 DRAWTO 79-I,I:REM DR. TOP RIGHT CORN
    ER
170 DRAWTO I,I:REM DR. TOP LEFT TO COMPL
    ETE BORDER
180 NEXT I
190 Z=PEEK(712):REM SETS Z EQUAL TO THE
    VALUE IN THE LAST REGISTER
200 POKE 712,PEEK(711):REM ROTATES VALUE
    S FROM 711 TO 712
210 POKE 711,PEEK(710):REM ROTATES VALUE
    S FROM 710 TO 711
220 POKE 710,PEEK(709):REM ROTATES VALUE
    S FROM 709 TO 710
230 POKE 709,PEEK(708):REM ROTATES VALUE
    S FROM 708 TO 709
240 POKE 708,PEEK(707):REM ROTATES VALUE
    S FORM 707 TO 708
250 POKE 707,PEEK(706):REM ROTATES VALUE
    S FROM 706 TO 707
260 POKE 706,PEEK(705):REM ROTATES VALUE
    S FROM 705 TO 706
270 POKE 705,Z:REM ROTATES VALUES FROM 7
    12 TO 705
280 FOR I=0 TO 15:NEXT I:REM SLOW DOWN R
    OTATION
290 GOTO 190:REM START AGAIN
```

This program rotates the border colors to give a theater marquee effect. To display even more of this mode's capabilities, add the following lines:

```
185 A=96:REM SETS A VARIABLE FOR THE BAC
    KGROUND COLOR
272 A=A+1:POKE 704,A:REM CHANGES BACKGRO
    UND COLOR
275 IF A=255 THEN A=1:REM ALLOWS ONLY GO
    OD COLOR VALUES
```

These additional lines will rotate the background color through all its possibilities while the border is rotating.

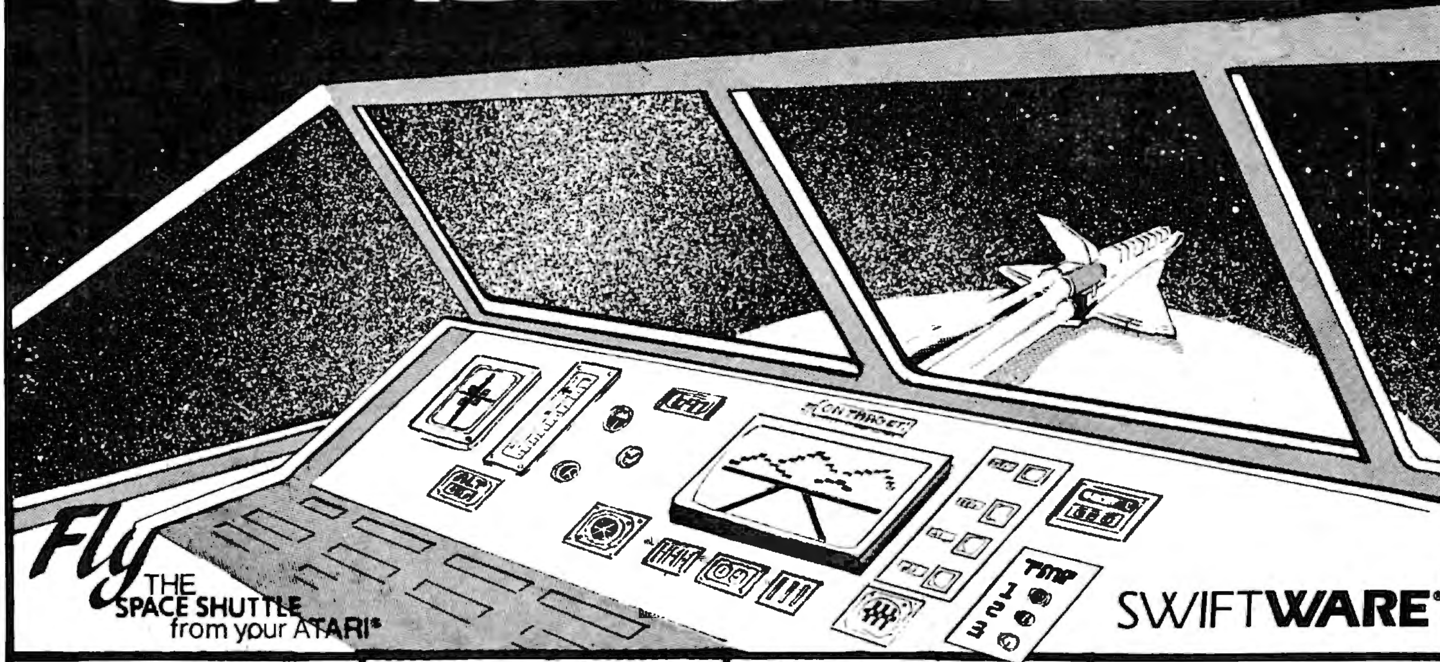
Graphics 11

Mode 11 operates similarly to Mode 9. The difference is that only one luminance or shade is used and a choice of all 16 colors is given. Bits 6 and 7 are set to one for this mode. Again, the background register is used for the colors, with ANTIC supplying the data. COLOR 0-15 relates exactly to the COLOR segment in the SETCOLOR command. To initiate this mode you must choose the luminance or shade you want. The color would be set by your COLOR statement (SE. 4,0,0-15 Lum choice). The background is always COLOR 0 (black). This mode allows fine color blending to produce rainbow effects and therefore a wider color choice for picture making.

The following demonstration program draws a cross in 16 colors, again using a 1 x 1 Display Block, and then draws an ellipse in 16 colors around the center of the cross. This program shows the versatility of color use in Mode 11. No longer are we restricted to horizontal screen architecture for extra color with Display List Interrupts.

```
5 REM GRAPHICS 11 DEMONSTRATION PAGE 5
10 A=1:R=26:REM SETS VARIABLES
20 DIM X(360),Y(360):REM ALLOW STORAGE S
   PACE FOR X AND Y COOORDINATES
30 GRAPHICS 11:SETCOLOR 4,0,12:DEG :REM
   SETS GR. MODE, LUM OF COLORS AND DEGR
   EE MODE FOR ELIPSE
40 FOR I=0 TO 15:REM COLOR AND POSITION
   VARIABLE
```

SPACE SHUTTLE



GAMES

SPACE SHUTTLE ™ by Paul Kindl

Join the crew of the *Space Shuttle* as they prepare to take the next step into the world of space travel. Take control of the world's first reusable spaceship, the *Space Shuttle*, and in an accurate full graphics simulation, place yourself in the cockpit. Pilot the *Space Shuttle* through take-off with booster stage separations, orbit, descent down the glide path and landing to touchdown complete with a chase plane and scrolling runway visible through the cockpit windscreen. You assume command throughout all phases of the mission aided by complete instrumentation.

Price: $29.95 (32K Diskette)

HAUNTED HILL ™ by George Richardson

Fight bats and ghosts in the dark of a cemetery. This exciting, all machine language game has arcade quality graphics and speed. Requires Joystick.

Price: $29.95 (16K Diskette/$24.95 16K Cassette)

TRIVIA TREK ™ by Jerry White

A multiple game of trivia for one or two players. Comes complete with 500 trivia questions in fifty categories and over two thousand multiple choice answers. A program for creating your own trivia questions and answers is also included. Play it for fun, test your knowledge or entertain friends. Fun for the whole family and great at parties. Joystick optional.

Price: $29.95 (32K Diskette)

FUN 'N' GAMES ™ by Jerry White

Wordgames, *Possible* and *Leapfrog* giving you hours of fun, challenge and entertainment. *Wordgames*, two games in one, contains *Guessit* - a deductive alphabetic reasoning game for one or two players and *Wordjumble* - a multiple word descrambling puzzle with play-on-word hints and mystery answers. Instructions show how you can substitute your own words. Use *Possible* to help descramble word jumble puzzles or to create your own. All letter/number combinations or permutations of input are printed to screen or optional printer. *Leapfrog* is a Chinese-Checker type jumping game in which you try to position two sets of animated jumping frogs in a minimum number of moves. Disk version of *Guessit* works with a Votrax Type 'N' Talk. A real crowd pleaser. Joystick optional.

Price: $17.95 (16K Cassette/$19.95 24K Diskette)

DATABASE MANAGEMENT

FILE-IT C ™ by Jerry White

A cassette only start-up database management system, designed to work in a 16K computer, that files and manages personal information and data. This friendly, menu driven package creates, sorts, stores and manipulates information. *File-It C* easily configures and handles data such as appointment calendars, address or telephone records, credit and charge card records, stock investments, medical and prescription information, hobby, coupon and other types of collection inventories. Because of its flexibility, *File-It C* is limited only by the user's imagination and creativity. Supports optional *Atari 825* or *Epson MX-80* printers. Prints one or two across mailing labels and neat copies of all *File-It C* data files. Fast and easy to use, *File-It C* holds over 300 records in 40K.

Price: $19.95 (16K Cassette)

FILE-IT 2 ™ by Jerry White

A deluxe package which contains all of the features and programs in *File-It C* plus five additional especially designed file handling and financial management programs. Six user defined fields are created with up to 5 subfields beneath each main field. The *Financial Entry* and *Financial Report* programs create a powerful personal financial management system recording transaction dates, expense and income entries. Three additional user designated fields give bottom line accounting capability. Two additional *File-It* programs provide random access updating and user controlled record selection. Full field and sub-field sorting is available. Data-files are stored on separate data disks permitting large scale data management. Subfiles may be created, merged and sorted by any field. A *Bar Graph* program generates a screen display of financial data including values and percentages in bar graph form which may optionally be printed. Supports up to four disk drives as well as the *128K Axlon Ramdisk*. Extensive documentation, attractively bound, provides clear user instructions along with a tutorial on computer filing. The *"Alternative"* to more costly data-base management systems. 80 column printer is optional.

Price: $49.95 (24K Diskette)
plus $3.50 for shipping and handling

FAMILY FINANCIER ™ by Paul Thompson

Family Financier, a menu driven cassette based personal finance package, provides a fast and easy means to balance checkbooks, record checks and generate account statements and monthly totals keyed to budget items and/or payees. Stores all data on cassette. Printer optional.

Price: $24.95 (16K Cassette)

DISK MAINTENANCE

SWIFTY TACH-MASTER ™ by Tony Dobre

An accurate *(disk speed diagnostic)* utility program designed specifically for *ATARI 810* Disk Drives. Provides easy-to-read visual indication of the speed of any drive connected to your system. Using machine language, *Tach-Master* displays five RPM readings per second with a working tachometer accurate to ¼ RPM. Allows adjustment of your drive(s) to factory specs easily without special equipment. Comes complete with easy to follow user's manual.

Price: $24.95 (16K Cassette/$29.95 16K Diskette)

DISKETTE INVENTORY SYSTEM ™ by Jerry White

Use this system to gain control of your expanding disk/program inventory. Quickly get locations of single or multiple copies of your programs and all your valuable files. An invaluable tool, this system is easy and convenient to use and to update. Printer optional

Price: $24.95 (24K Diskette)

ACCESSORIES

DISK SENTRY ™

An intelligent digital accessory for your *Atari 810 Disk Drive*, lets you selectively write data to both sides of single sided and write protected disks. *Disk Sentry* cannot harm your drive disks. Installs and removes easily; no soldering required. *Disk Sentry's Led* signals system status, preventing accidental erasure of data. *Disk Sentry* is a convenient push button write-protect override which can pay for itself with your first box of disks.

Price: $39.95 each

VINYL DUST COVERS

Custom sewn upholstery grade vinyl dust covers for *Atari 800* and *400* computers and the *810* and *825* peripherals. Protecting your equipment when not in use, these crisp black covers go well with any decor. Specify model(s) when ordering.

Price: $10.95 each

TUTORIAL

JERRY WHITE'S MUSIC LESSONS ™
INCLUDING Player Piano and Singalong

This two sided diskette or 2 cassette music and sound tutorial by *Jerry White* contains 13 separate programs. It is designed to provide demonstrations, in *Atari Basic*, of the *Atari Home Computer* system's sound capabilities both for music and sound effects. *Music Lessons* comes with extensive, bound documentation complete with program listings. *Player Piano* turns your computer into a twenty note mini-piano that includes a full, graphically accurate, screen display of the piano keyboard. The *Singalong* programs teach and play songs in four part harmony while displaying lyrics on the screen.

Price: $29.95 (16K Cassette/$29.95 32K Diskette)

TELECOMMUNICATIONS

DATALINK ™ by Tony Dobre

Top rated national magazine reviews, purchased by *NASA*, this ultra-sophisticated menu driven multioption smart terminal Communications package is a *must* for all modem users. Features full performance uploading and downloading in full-duplex or simplex modes and supports *ASCII* and *ATASCII* transmission. Automatically off-loads line data to your disk drive upon completion of a transmission with options available to screen and/or printer dump the same data. May be used as a remote terminal compatible with all the commercial services and bulletin boards such as the *Source* and *Compuserve*, etc. Saves significant connect time charges when using commercial services. Requires *Atari 850 Interface* or equivalent and *Atari 830 Modem* or any other 300 Baud modem. Printer optional.

Price: $39.95 (24K Diskette)

ALSO AVAILABLE

PROGRAMMING AIDS PACKAGE ™
SWIFTY UTILITIES ™
MAILIST ™ by Leo Cerruti

— AVAILABLE AT SELECT COMPUTER STORES —

Send check or money order plus $2.50 shipping and handling to

SWIFTY SOFTWARE, INC.
64 BROAD HOLLOW ROAD
MELVILLE, New York 11747

(516) 549-9141
Call to Place Your Order
N.Y. Residents add 7¼ % sales tax

SEND $1.00 FOR CATALOG
(refunded on your first order)

DEALER INQUIRES INVITED

VISA MasterCard

```
50 COLOR I
60 PLOT 31+I,0:DRAWTO 31+I,191
70 PLOT 0,86+I:DRAWTO 79,86+I:REM DRAWS
   CROSS
80 NEXT I
90 FOR I=0 TO 360 STEP 2
100 X(I)=R*COS(I)+34
110 Y(I)=R*SIN(I)+95
120 NEXT I:REM SETS X AND Y VALUES FOR P
    LOTTING ELIPSE
130 FOR I=0 TO 360 STEP 2:REM CALLS ABOV
    E VALUES
140 COLOR A
150 PLOT X(I)+A,Y(I)+A:REM PLOT EACH COL
    ORS' ELIPSE
160 NEXT I
170 A=A+1:REM NEXT COLOR AND NEXT ELIPSE
     POSITION
180 IF A=16 THEN 200:REM END IF ALL COLO
    RS ARE USED
190 GOTO 130:REM DRAW NEXT ELIPSE
200 GOTO 200
```

The final demonstration program draws a landscape and a simple molecular structure floating high above it. This display truly gives the impression of depth and shows what can be done using light and shadow in Graphics Mode 9:

```
10 R=16:X=0:C=15
20 GRAPHICS 9:SETCOLOR 4,13,0
30 FOR I=130 TO 191
40 COLOR C
50 PLOT 0,I:DRAWTO 79,I
60 X=X+1:IF X=4 THEN X=0:C=C-1
70 NEXT I
80 FOR I=0 TO 79 STEP 8
90 COLOR 3:PLOT 59,130:DRAWTO I,191
100 NEXT I
110 COLOR 1:FOR I=0 TO 7:PLOT 2,164:DRAW
    TO 21,158+I:NEXT I
120 COLOR 15:FOR I=0 TO 3:PLOT 21,140:DR
    AWTO 21+I,164-I*2:NEXT I
130 COLOR 4:FOR I=0 TO 4:PLOT 20,140:DRA
    WTO 17+I,160+I:NEXT I
140 FOR Z=1 TO 15
150 FOR I=0 TO 360 STEP 6
160 X=0.25*R*COS(I)+35
170 Y=R*SIN(I)+50
180 COLOR Z
190 PLOT X,Y
200 PLOT X+10,Y+17
210 PLOT X+30,Y-20
220 PLOT X-2,Y+12
230 PLOT X+21,Y+70
240 NEXT I
250 R=R-1
260 NEXT Z
270 FOR I=2 TO 4:COLOR I:PLOT 46,72:DRAW
    TO 51+I,106
280 PLOT 43,62:DRAWTO 39,50+I
290 PLOT 47,62:DRAWTO 60+I,35
300 NEXT I
310 GOTO 310
```

These demos are only an introductory hint of the truly spectacular effects possible via GTIA. You could add more color to Modes 9 and 11 by using Players and Missiles or create dramatic effects by switching between these modes (or with Graphics 8) by POKEing PRIOR with the desired value.

A Monthly Column

Machine Language

Jim Butterfield
Associate Editor

If I May Interrupt

The 6502 goes about its job, executing instructions at lightning speed. As each instruction is completed, the processor checks: should this process be interrupted?

There are two kinds of interrupt, called IRQ and NMI. They have different features and uses, but they share common characteristics. They may only take effect when the current machine language instruction has been completed. At that time, the address of the following machine language instruction is pushed to the stack together with the Status Register. Then the machine gets an interrupt address stored high in memory, and starts to execute instructions from that address. At a later time, when the interrupt job has been serviced, an RTI instruction will cause the previously stored information to be reclaimed from the stack and the interrupted program to continue.

Interrupt is high priority. It tends to be used where fast response is vital. Don't throw it away on some unimportant job which is not time-sensitive; save these big guns for a real time crunch. I tend to recommend the following priorities: if you can, use straight coding; if you need to, use a timer or two; if you must, use interrupt.

Because an interrupt stops the work in progress to handle a special rush job, users often tend to think of it as instantaneous. Not quite. Don't forget that there's a variable wait to complete the instruction under way (up to seven cycles) in addition to the fixed delay of seven cycles while the interrupt does its bookkeeping work. The effect of the variable wait is "timing jitter" – occasionally important even though the time involved is small.

The Big Two

IRQ – Interrupt Request – is the less powerful of the two interrupts, but it's usually easier for the programmer to handle. It may be locked out with an SEI instruction (Set Interrupt Disable) to prevent interruption from striking at an embarrassing moment; the lockout is released with CLI (Clear Interrupt Disable). Using SEI/CLI adds to the possible timing jitter by a substantial amount, of course. When an interrupt takes place, an SEI-type lockout automatically takes effect, so that another IRQ interrupt will have no effect until RTI releases the lockout. This is handy for the programmer – he knows that the code in his IRQ type system will be free from further interrupts.

NMI – Non-Maskable Interrupt – is more powerful and less controllable. It cannot be locked out. As a result, the programmer has to be much more careful in sensitive areas: for example, changing the interrupt vector itself can be a ticklish job since the coding cannot prevent the NMI from striking in mid-change with potentially disastrous results. To add to the complexity: an NMI could cause an interrupt, and while it is being handled, another NMI could interrupt again. Careful coding is needed to avoid data corruption if such a multiple-level interrupt is anticipated.

There's another fundamental difference between IRQ and NMI. IRQ is level-sensitive: when the IRQ pin on the 6502 chip receives a low level, interrupt is being requested. NMI, on the other hand, is edge-sensitive: when the NMI pin on the 6502 chip goes from high level to low, a "latch" is triggered within the chip that will signal that NMI needs attention. Think of it this way: if I held the IRQ pin low permanently, the computer would be continuously interrupting. It would go into interrupt, do the job, and upon completing with RTI, the interrupt would take place again since IRQ is still low. In contrast, if I pulled NMI low permanently, I would have only one interrupt – the one that was triggered when the signal went low. A new "edge" would be needed to trigger NMI again.

IRQ Latches

This gives us two seemingly conflicting requirements for the interrupt signal at the IRQ pin. First, it must remain active until the interrupt takes place; too brief an IRQ signal might be missed entirely. Next, it must be turned off before the interrupt coding completes its activity, or RTI will just cause a new interrupt. This seems difficult – not too fast and not too slow – but, in fact, we accomplish the job very easily with the help of extra chips.

Most of the interface chips (the best known to 6502 users are the 6520 PIA and the 6522 VIA) contain latches that may be set by the external interrupting circuits, and reset by the 6502. For example, if a timer counts down to zero and signals an interrupt, this will be latched and signalled to the 6502. When the 6502 gets around to servicing the interrupt, it can switch off the latch.

This system of latches allows many interrupts to be received and forwarded to the processor

chip. The computer can then interrogate the interface chip and find out what caused the interrupt. There might even be two events calling for service at about the same time. The computer can decide to service one of them, turn that particular latch off, and do the job. The moment it gives RTI the other event (whose latch is still locked in) will re-interrupt and be serviced. It works out remarkably elegantly.

The interface chips may have external ports or built-in devices such as timers and shift registers which are allowed to cause interrupts. Each of these may be logically connected to or disconnected from the interrupt line. It seems complex at first; but a little practice will show the system to be straightforward and logical.

Registers

You may recall that only the instruction address (Program Counter) and Status Register (sometimes called the PSW) are saved on the stack during an interrupt. If you plan to use the A, X or Y registers during your interrupt processing, you must save them by pushing them to the stack. Just before giving RTI, bring them back. Your interrupt must be truly "invisible" to the code that was interrupted.

It's quite easy to implement interrupt. You must be especially careful; debugging is much more difficult for this type of code.

Try to keep your interrupt code short, and let the "background" program pick up and do most of the work. The briefer the interrupt program, the more often you'll be able to service interrupts; that will often yield a more powerful system.

Be very careful that a long interrupt doesn't disturb a critical timing process in background code. More than one thermal printer has had its head "smoked" by a sluggish interrupt that didn't know that the background program was waiting to turn the heat off.

This is a short and very useful addition to VIC. Some PET/CBM computers allow you to stop a LISTing from scrolling and then start it up again. This "Pause" feature adds that capability to VIC (any memory size). Type in the first line very carefully – it's got to be exactly as printed.

VIC Pause

Doug Ferguson
Elida, OH

For VIC owners who have not bought a printer yet, studying a BASIC program by using the LIST command can be tedious. The screen displays only about 20 lines at a time if you hit the STOP key. And then the only way to re-start at the point at which you stopped is to retype LIST – again and again. Even the CONTROL key is not much help; the lines still move by too fast for more than a superficial look. What is needed on the VIC-20 is a PAUSE key.

For months I have been trying a method for the PET from G. H. Watson's "Linelist" program (in **COMPUTE!**, September 1981, #16), but without total success. At least the experience taught me a lot about disassembling the BASIC ROM. Finally, with the help of Eric Brandon's advice (**COMPUTE!**, May 1981, #12, pg. 126) I discovered an alternate method which patches into the hardware interrupt vector. This vector in the VIC is located at 788 and 789 and contains the low-byte/high-byte jump vector $EABF which the VIC visits every 60th of a second to check the STOP key, to update the clock, and to do other chores.

The routine which I have "persuaded" the VIC to jump to every 60th of a second is very similar to Mr. Watson's: to pause a LIST, hold down the SHIFT key. The SHIFT key is the ideal PAUSE key because it can be "locked" with the SHIFT LOCK key. This allows totally hands-free operation so that you can study (or copy in longhand) your BASIC program a few lines at a time.

I have written the machine language loader without DATA statements in case you want to add the PAUSE feature to a program which may already contain conflicting DATA statements. I preferred the cassette buffer as a house for my 26-byte masterpiece because this area stays stable regardless of memory expansion of the VIC.

I used location 888 as a start point because I thought it easier to recall than other traditional cassette buffer addresses. Please note that the first line is a very tight fit; type no space after the line number, and consider the final quotation mark optional.

```
63997 A$="12016913314102000316900314102100308809
    6032159255169001044141002208246076191
    234"
63998 P=1:FOR X=888 TO X+25: POKE X,VAL(MID$(A$,
    P,3)): P=P+3: NEXT
63999 SYS 888
```

When you have finished (carefully) typing the loader, execute with RUN 63997. (With an 8K expander, the VIC may give you an ILLEGAL QUANTITY error. Assuming no errors in your typing of the program, try again several times. The unexpanded VIC does not have this trouble with string variable manipulation.) After a successful RUN, feel free to delete lines 63997 - 63999.

Now it's time to test your new PAUSE key. Try it by freezing the blinking cursor (on or off) with the SHIFT key. Next, press RUN/STOP and hit RESTORE to clear the screen. Note that the SHIFT key does not operate as before. The RUN/STOP-RESTORE technique returns the hardware interrupt vector to its normal contents. If you wish to "freeze" a LIST again, merely SYS 888 to reactivate the PAUSE feature.

Be Sure To Turn It Off Before SAVEing

It is essential to deactivate by the RUN/STOP-RESTORE method when you are not using the PAUSE feature. For one reason, any I/O operations (such as a SAVE of whatever you have been debugging) require that the hardware interrupt vector be returned to normal. (Also remember that a SAVE uses the cassette buffer where the PAUSE routine lives; you may expect to lose the PAUSE feature after a SAVE or a LOAD.)

Another reason to deactivate the PAUSE is that shifted characters such as uppercase and graphics are strange to type, in that they appear *after* you let up on the SHIFT key. Even worse is the loss of repeat keys such as cursor-up and -left and the disappearance of the cursor when it moves rapidly right or down.

I bet you're wondering if there are any other drawbacks to my idea. Well, yes. When you do SYS 888 to activate the PAUSE key and you do try to freeze a LIST, you will occasionally get a kind of "ripple" on the screen which causes a line to be, more or less, repeated twice. I believe that this is a tolerable annoyance, especially if one remembers that the upper half of these "paired" lines is always correct.

By the way, the PAUSE will also work when you RUN a program. However, if you only want a PAUSE feature for this reason, there are much better ways. Lines 132-136 of Amihai Glazer's "Amortize" program (**COMPUTE!**, May 1982, #24) are just perfect in the main routine of any BASIC program in which you want a PAUSE.

The assembly listing that follows was written on Eric Brandon's excellent assembler program (**COMPUTE!**, June 1981, #13). It required memory expansion and a few TAB changes.

```
HIV     = $0314
INTRPT  = $EABF
SCNKEY  = $FF9F
PAUSE   = $0385

1                                *       =    888
2                                HIV     =    788
3                                INTRPT  =    $EABF
4                                SCNKEY  =    $FF9F
5    0378  78                            SEI
6    0379  A9    85                      LDA  #$85
7    037B  8D    14  03                  STA  HIV
8    037E  A9    03                      LDA  #$03
9    0380  8D    15  03                  STA  HIV+
10   0383  58                            CLI
11   0384  60                            RTS
12   0385  20    9F  FF      PAUSE       JSR  SCNKEY
13   0388  A9    01                      LDA  #1
14   038A  2C    8D  02                  BIT  653
15   038D  D0    F6                      BNE  PAUSE
16   038F  4C    BF  EA                  JMP  INTRPT
```

Here are a few notes on the assembly listing. Lines 5-11 change the hardware interrupt vector (HIV) to $0385 (PAUSE); the SEI and CLI are necessary to make the computer wait until both low and high bytes are changed. I used a BIT test because location 653 also determines the CONTROL key, which is often used to slow down a LIST. The whole thing ends with JMP INTRPT to send the VIC where it had intended to go. Lines 12-15 are just sort of a detour.

This has been my first real foray into machine code, and I would appreciate any helpful comments. ©

What is ASCII? How and why is this special code used to communicate between one computer and another? Some computers make slight modifications to the code which must be taken into account when sending messages to a different model. The Atari uses 155 to stand for a carriage return, for example, where most computers use 13. To see what the code number 77 looks like to your machine, you can type ?ASC(77). Whatever character shows up on screen is the meaning of that number. Of course some are special characters. ?ASC(13) will generally perform a carriage return. For a complete table of standard ASCII codes, see **COMPUTE!**, *July 1982, p. 140.*

Telecommunications:

All About ASCII

Michael E. Day
Chief Engineer, Edge Technology
West Linn, OR

ASCII is an acronym for American Standard Code for Information Interchange. More specifically, it is a definition of a code that is used in most computers to store and transmit information.

The computer works with information in *bits* or "on/off conditions" of its memory cells. This means that in order for it to work with numbers and letters it must work with them as a group of on/off bits. The ASCII code defines the representation as seven bits, while most computer systems deal with numbers and letters as an eight-bit code sequence, with the eighth bit being used internally by the computer for its own purposes. The seven bits can be formed into 128 possible combinations, with each combination representing a single character or letter. These 128 characters are broken down into 52 upper and lowercase letters (A-Z & a-z), ten numbers (0-9), 33 special characters (including the space), and 33 special "control" characters. ASCII is a non-shifted code; that is, any of the seven-bit code combinations represent a unique character.

BAUDOT Code

BAUDOT, on the other hand, is an example of a shifted code. BAUDOT code uses only five bits to represent information. As a result, only 32 possible combinations are available. In order to use this method, some decisions had to be made as to what information would be allowed to be represented.

The first thing thrown out is the lowercase letter, as information can be maintained in uppercase only without destroying its meaning. Next the special control functions are reduced to the bare minimum needed – carriage return, line feed, and space. The 26 uppercase letters and the three control functions use up 29 of the available codes. We cannot fit the ten numbers in the three remaining codes, so we have to go about it in a different way.

The numbers and some special characters are provided for by setting aside two of the remaining codes as shift functions. A shift code changes the

The ASCII code standard... permits most computers to talk to one another.

definition of all following codes, until another shift function directs a return to the original code representation. By using shift functions, it is possible to almost double the number of characters that can be represented by the same 32 codes. The final remaining original code is not used and is referred to as a blank or null character.

The null character is used in the BAUDOT code as a non-operational code (it is ignored if it is encountered). The reason for this requirement is the way the machines function. When data is transmitted, it is sent in serial form, or a bit at a time, to the receiving equipment. The old teletype machines which sent and received this code used current flowing through a wire (or not flowing) to represent the on or off bit condition. The problem with this was that, if the wire was broken (which happened quite often), it was the same as sending a continuous stream of off bits, which of course represents one of the possible codes. By ignoring this particular code, at least the machine wouldn't spit out reams of paper if the wire did get broken.

The shifting codes were called the letters shift and the figures shift, which corresponded to the functions they performed. That is, when the letters shift was sent, it meant that all the codes that followed would represent the letters A-Z. The figures shift meant that all the codes following would represent numbers and special characters. The individual codes were assigned to the various letters in a way to ensure a minimum number of errors occurring in transmission, so that when an error did occur it would be noticeable. The letters shift code was chosen to be all on bits; this was to help both in transmission error problems as well as easing editing for transmission.

Erasing And Editing

The all-bits-on code has a special function. Information used to be stored as holes punched in a paper tape. If a mistake was made during creating this tape, then a new one would have to be made. As with most writing, however, the mistakes were often caught as they were made. With the letters shift code being made up of all on bits, all that was needed to wipe out the bad code was to back up to that section of tape and type the letters shift, which punched out all holes on the tape, changing any code on the tape to the letters code. The disadvantage to this was, of course, the possible inadvertent shift to letters case which required a following figures shift (if that was the case that you were in).

When the ASCII code was developed, the paper tape method was still in use, so mistakes were taken care of with a special function code called a *rubout*. The rubout code works the same way as the null code: when it is encountered it is ignored, except that it has the additional editing function of punching all holes in the tape to wipe out the existing code.

Smarter Editing

When computers came along, and it was possible to add some intelligence to the editing process, the rubout was used to tell the computer to back up and erase the last character that was typed. Since we were still using a printer though (one that did not know how to backspace at that), it was still difficult to deal with the information as it was written. Several editing methods were tried to overcome this problem. One of the most common methods was to simply print the character again as it was deleted. This tended to make the result rather confusing, but it did provide a way for the computer to indicate that it had performed its function.

As backspacing printers became available, another method of editing came into being: when the rubout key was depressed, the printer would back up and strike over the previous character with a backslash or some other character. It would then wait for the next character to be typed. If it was another rubout, it would back up and strike over the previous character. If the next character typed was not a rubout, then the printer would move the paper up a line and strike the character on the line below the original character. Another editing method sometimes used because it's easy on a computer was simply to ignore the entire line just typed and start over again. The operator indicated this method by doing a carriage return.

When video terminals came along, the editors took on a slightly different approach to the problem. The video terminal was capable of truly removing the wrong character from the text as it appeared on the screen. Since the rubout key was already being used for the printers, the backspace was used to back up and erase the previous character. Since the error was truly erased from the screen, there was no need to go to the next line, so the backspace simply remained in that character position after it was done. The usual sequence involved was to backspace, space to erase the character that was there, and then backspace again to return to the erased character's position.

Now the printer tends to be relegated to the side to be used only when the text is ready to be printed. The exception is in portable or very low budget operations where the printer must serve as both the editing device and the printing device.

The video terminal has grown from a "glass teletype" to what is now called *full-screen editing*. Full-screen editing is an extension of the backup and erase function, except it is enormously more powerful. Anything that appears on the screen can be directly altered by moving the current position indicator (the cursor) to the position you wish to modify and simply typing over it, deleting it, or inserting new information. The full-screen editor has the advantage of being quite easy to use since what you see is what you get.

Taking things a step further, we end up with a full-blown word processing system where not only can you change the data as it appears on the screen, but you can also move it around, add in whole new sections from storage, format the way it is to be printed, and even have it check your spelling.

Who knows? Maybe next it will start writing things for us, and we can just sit back and enjoy the show.

The Origin Of ASCII

As you can see, the ASCII code was developed as a result of the interaction between the needs of the equipment and the needs of the operator. This is an important aspect of equipment design and is often the dividing line between good and bad machinery. It should be remembered, however, that the coding system used is just that: a coding system. It can be changed, modified, compressed, or expanded, and many other things can be done to it following standard coding theories. The ASCII code standard is simply there because it is the common element that permits most computers to talk to one another. This of course doesn't mean that all computers use the ASCII code standard.

A very early code system used on punched cards was a ten-bit code, later expanded to 12 bits, called the Hollerith code. An outgrowth of the Hollerith code was the BCD (Binary Coded Decimal) code, which was a simplification of the

Hollerith code.

The BCD code uses six bits to represent 64 possible code combinations. Later on, due to the needs of the computer systems, this code was expanded to eight bits and called EBCDIC, for Expanded Binary Coded Decimal Interchange Code.

By this time the code had become quite a mess, what with all the expansions and extensions. The ASCII code was then developed by the American National Standards Institute (ANSI) to simplify code use – to make the code easier for machines to use, as well as easier for humans who had to use it. The main improvement was to place all letters in sequential order, so that it became easier for the computer to sort out information.

There are many other codes in use in computers and other equipment, each having been developed to satisfy a particular need – Selectric code to mechanically control the Selectric typewriter; Addressograph code to control Addressograph machines; seven segment code to drive seven segment readouts; Gray code to provide positional information of rotational devices; and a slew of error correcting and detecting codes designed to provide a more error-free information flow. ©

There's a bit of a problem with the VIC INPUT command. These suggestions will keep you out of difficulties when you are "prompting" questions and expecting to INPUT an answer.

A VIC Bug

Jim Law
Toronto

For Commodore PET users, one of the most talked about commands is the INPUT statement. Specifically, how do you prevent a user from stopping the program by hitting the RETURN key on an empty line? When Commodore designed the VIC-20 they decided to do something about this persistent problem. They decided that if a user presses RETURN and nothing else, the value(s) of the input variable(s) should remain unchanged and the program should continue.

```
10 J=77
20 INPUT "GIVE ME A VALUE";J
30 PRINT J
```

When the above program is RUN on a PET and the user hits only RETURN when asked "give me a value," the program stops and line 30 is never executed. If RUN on a VIC-20, the program will continue to line 30 and the unchanged value, 77, of the variable J will be printed. (If you want to break out of a VIC INPUT statement, press RUN/STOP and RESTORE simultaneously.) But there's a bug in this.

The INPUT Problem

In 40 column PETs of any vintage is an operating system feature called the line-wrap table. This allows the PET to link two screen rows together to get an 80 column line. In the VIC the line-wrap table is able to link up to four 22 column rows together to make one 88 column line. Without it, the VIC's full-screen editor would be unable to handle lines longer than 22 characters.

If a VIC INPUT statement places the input cursor anywhere other than on the first screen row of a multiple-linked line, the INPUT statement will erroneously return the entire multiple-linked line to the input buffer for parsing.

What this means is that if you have some rows which are linked together to make a 44, 66, or 88 column line and the input cursor is not placed on the first screen row of one of these multiple lines, the INPUT statement will "crash" or return a wrong value. If your prompt is on an un-linked line, you'll have no problem.

Suppose we have a very simple program to ask for a person's name and then print it on the screen:

```
10 PRINT "[CLEAR]";
20 INPUT "TYPE YOUR NAME AND PRESS
   RETURN";N$
30 PRINT N$
```

On a PET this works great, and if Fred is using the program he gets back "Fred" on the screen. On the VIC, he gets back "type your name and press return? Fred". The prompt in the INPUT statement in line 20 ("type your...") is longer than 22 characters and is printed over the end of the first row, creating a 44 column line. When Fred pressed RETURN, the entire 44 characters were returned to the input buffer, leading and trailing blanks dropped and the result assigned to the variable N$. This works even if Fred did nothing other than press RETURN.

The fun is just beginning.

```
10 PRINT "[CLEAR]";
20 INPUT "HOW OLD IS YOUR GRANDMOTHER"
   ;AG
30 PRINT AG
```

No matter what Fred types in response to this prying question, all he gets back is: "?redo from start". If he types "83"[RETURN], then: "how old is your grandmother? 83" gets moved to the input buffer; leading and trailing blanks are removed; and the computer attempts to convert "how old..." into a number for variable AG. Forget it, Fred. Won't work.

If your prompt is shorter than one line, everything gets sorted out correctly and the statement works as expected. Usually. Long INPUT prompts are not the only way to get linked screen rows:

```
10 PRINT "[CLEAR]LINE #1            "
20 INPUT "[HOME][DOWN] FROM LINE #2";X$
30 PRINT X$
```

Be sure to put a bunch of spaces followed by a quote mark after "line #1" on line 10. This sets the trap.

If line 10 were not there, this program would work fine, as the value of x$ would be assigned from a non-linked line. But line 10 creates a 44 character line at the top of the screen. The INPUT statement then types over the second row of the screen, which is also the second screen row in the 44 character line created in line 10. The value returned to x$ and printed in line 30 could be: "line #1   from line #2? hello".

The One Sure Cure

The only sure ways to make sure the rows from which you will input are not linked are: 1) clear the screen – this clears the line-wrap table to all unlinked rows; and 2) scroll off the bottom of the screen – any new rows placed at the bottom of the screen are always single. In addition, if you have any long prompts, put them in separate PRINT statements before your INPUT and do not use a semicolon or a comma at the end of the PRINT.

```
10 INPUT "[CLEAR] WHAT'S YOUR NAME";N$
20 PRINT "THE INPUT IS ON A SEPARATE LINE"
   : INPUT X$
25 :
30 REM NEXT LINE LEAVES THE CURSOR AT
   THE START OF A SCREEN LINE
40 INPUT "ABCDEFGHIJKLMNOPQRSTUV";X$
50 INPUT "[CLEAR]ABCDEFGHIJKLMNOPQRSTU
   [RIGHT]ABCDEF";X$
```

Line 10 should work unconditionally. Line 20 will work if the row from which the actual input is done is not linked. Lines 40 and 50 work because the character which sends the cursor to the second row is not a printable character, and so does not link the two rows together. Lines 40 and 50 will not work if the rows were already linked. ©

Have you ever wanted to use text and graphics modes on screen at the same time? *Among the techniques in this article is a method which allows you to display both character sets (uppercase/graphics and upper/lowercase) simultaneously. This IRQ-driven routine uses streamlined, carefully calculated timing delays to synchronize the 6502 with the video scan lines on the CRT. This technique, called a* kernal *on some machines, permits previously unheard-of displays. If you want to play around with the routine a little, try POKEing various values into location 684. POKE 684,64 will change the third display line into a "hybrid line" with all shifted characters made of half of each character (shift-Q would be the top of the Q and the bottom of the "ball" character). It's designed for the 8032, with suggestions for modifying it for 40-column screens.*

Three PET Innovations

Timothy Stryker
Samurai Software
Pompano Beach, FL

Have you ever written a program in which you needed to mimic the flashing cursor that appears on the screen when an INPUT statement is in progress? Sometimes it's useful to be able to GET user keystrokes, but at the same time to display a flashing cursor to the user so that he thinks he's in INPUT mode. Here are a couple of little routines that can come in handy for this:

```
            CRSON  EQU *       TURN CURSOR ON
A0 00              LDY #0      SET BASIC CURSOR-ON FLAG
84 A7              STY $A7
C8                 INY         START FLASHING
                               IMMEDIATELY
84 A8              STY $A8
60                 RTS
            CRSOFF EQU *       TURN CURSOR OFF
78                 SEI         MASK INTERRUPTS
A5 AA              LDA $AA     CHECK CURSOR-RVS-FLD
                               FLAG
F0 0D              BEQ ALROFF  IF OFF, OK, PROCEED
A9 01              LDA #1      SET BLINK-COUNTER TO 1
85 A8              STA $A8
58                 CLI         UNMASK INTERRUPTS
A5 A8  WAIT        LDA $A8     WAIT FOR BLINK-COUNTER
C9 01              CMP #1        TO CHANGE
F0 FA              BEQ WAIT
D0 EE              BNE CRSOFF  GO MAKE SURE IT'S OK NOW
A9 01  ALROFF      LDA #1      SAFE TO KILL IT FOR GOOD
85 A7              STA $A7
58                 CLI         ALL SET
60                 RTS
```

Most people are able to figure out how to turn the thing on (usually via the BASIC command "POKE 167,0"), but the problem is that if you try to move the cursor around while in this state, little renditions of it are left behind as you go. This can be distinctly annoying, so you then experiment around with things like POKEing 167 with a one just before moving the cursor again, which also doesn't work. Eventually either you go crazy, or you write a routine like CRSOFF above, which is guaranteed to turn the cursor completely off so that you can move it someplace else and turn it on again.

The two above routines are completely relocatable, so you can stick them anyplace you like: in the tape buffers, in between BASIC lines – even on the screen. Here are the decimal POKEs from BASIC to put them in the second cassette buffer:

```
100 FOR I = 900 TO 931: READ P : POKE I,P : NE
    XT
110 DATA 160,0,132,167,200,132,168,96
120 DATA 120,165,170,240,13,169,1,133,168,88,1
    65,168
130 DATA 201,1,240,250,208,238,168,1,133,167,8
    8,96
```

Having done this, just SYS to 900 to turn the cursor on and SYS to 908 to turn it back off. Remember not to try to PRINT anything with the cursor turned on this way, or you'll find that it leaves the same little residues as before. Turn it off, do your PRINT, and then turn it back on again. Incidentally, the above will work only under the Upgrade and V4.0 ROMs. Change all the $A7's to $0224's, the $A8's to $0225's, and the $AA to a $0227 to make this work on an Original ROM machine (don't forget to change the addressing modes of the opcodes and the branch offsets too).

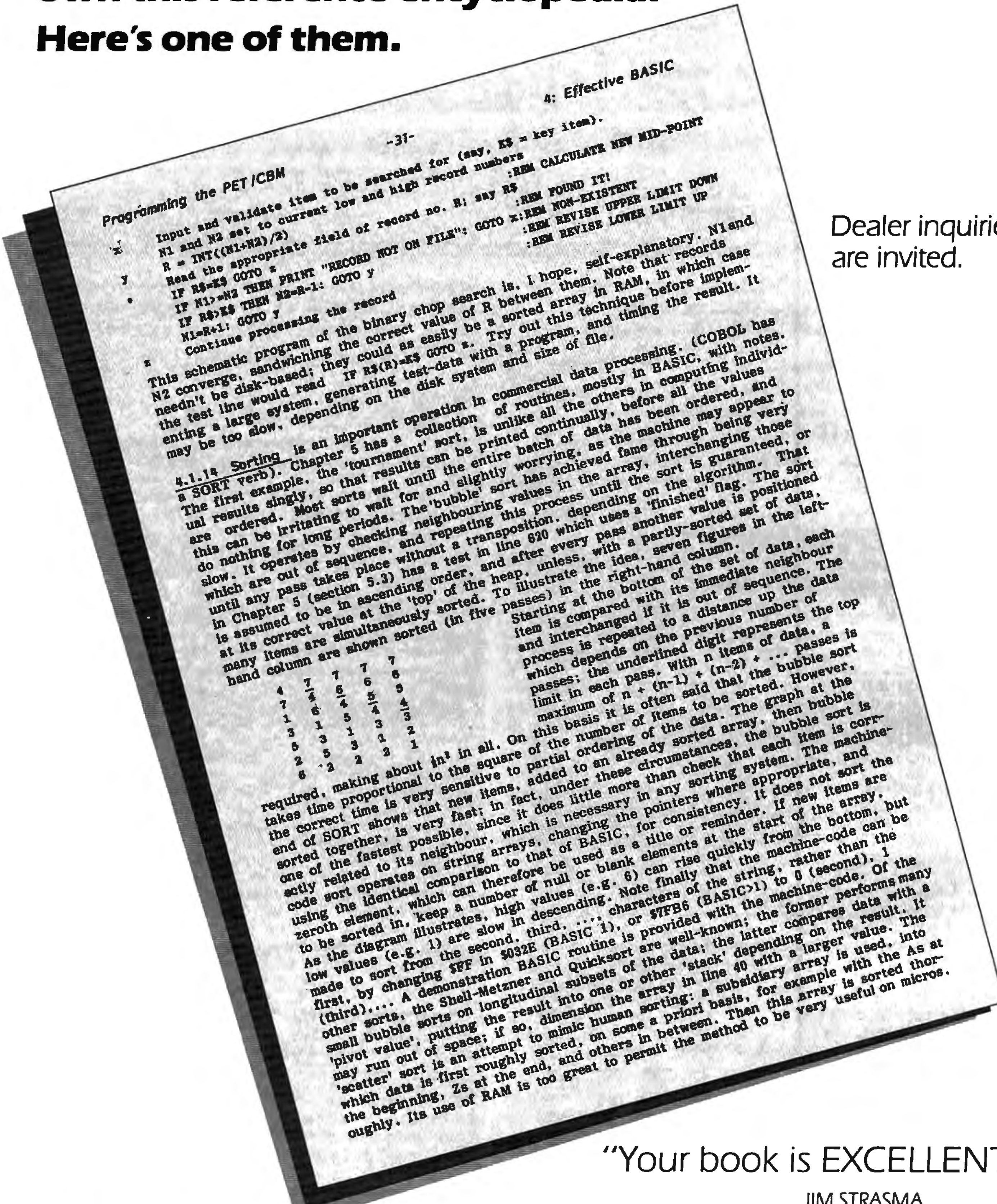
Hidden PRINTs

Another little item I've found useful doesn't involve machine code at all. Have you ever put debugging PRINT statements into a program, removed them once you had the thing debugged, and then later wished you had them back in again? This can be particularly likely if your program does any modem communications or that sort of thing, where the program might be working fine, but the line is getting garbled for some reason and you need to find out why. How about this: leave the debugging PRINT statements in, but make them conditional on the value of the PEEK of 152 (516 on the Original ROMs). The PEEK of 152 is a 1 if the SHIFT key is being held down, and a 0 otherwise. This way, if you want to see the debugging PRINTs, you can just hold down the SHIFT key; otherwise, the program operates normally. The reason for using the SHIFT key rather than some other key is that this way no extraneous characters will get stuck in the keyboard input buffer.

The last little item I have here is neither as

There are over 500 reasons to own this reference encyclopedia.

Here's one of them.



Programming the PET/CBM — -31- — 4: Effective BASIC

```
      Input and validate item to be searched for (say, K$ = key item).
      N1 and N2 set to current low and high record numbers
x     R = INT((N1+N2)/2)                        :REM CALCULATE NEW MID-POINT
      Read the appropriate field of record no. R; say R$
      IF R$=K$ GOTO z                           :REM FOUND IT!
y     IF N1>=N2 THEN PRINT "RECORD NOT ON FILE": GOTO x:REM NON-EXISTENT
      IF R$>K$ THEN N2=R-1: GOTO y              :REM REVISE UPPER LIMIT DOWN
      N1=R+1: GOTO y                            :REM REVISE LOWER LIMIT UP
z     Continue processing the record
```

This schematic program of the binary chop search is, I hope, self-explanatory. N1 and N2 converge, sandwiching the correct value of R between them. Note that records needn't be disk-based; they could as easily be a sorted array in RAM, in which case the test line would read IF R$(R)=K$ GOTO z. Try out this technique before implementing a large system, generating test-data with a program, and timing the result. It may be too slow, depending on the disk system and size of file.

4.1.14 Sorting is an important operation in commercial data processing. (COBOL has a SORT verb). Chapter 5 has a collection of routines, mostly in BASIC, with notes. The first example, the 'tournament' sort, is unlike all the others in computing individual results singly, so that results can be printed continually, before all the values are ordered. Most sorts wait until the entire batch of data has been ordered, and this can be irritating to wait for and slightly worrying, as the machine may appear to do nothing for long periods. The 'bubble' sort has achieved fame through being very slow. It operates by checking neighbouring values in the array, interchanging those which are out of sequence, and repeating this process until the sort is guaranteed, or until any pass takes place without a transposition, depending on the algorithm. That in Chapter 5 (section 5.3) has a test in line 620 which uses a 'finished' flag. The sort is assumed to be in ascending order, and after every pass another value is positioned at its correct value at the 'top' of the heap, unless, with a partly-sorted set of data, many items are simultaneously sorted. To illustrate the idea, seven figures in the left-hand column are shown sorted (in five passes) in the right-hand column.

4	7	7	7	7	7
7	4	6	6	6	6
1	6	4	5	5	5
3	1	5	4	4	4
5	3	1	3	3	3
2	5	3	1	2	2
6	2	2	2	1	1

Starting at the bottom of the set of data, each item is compared with its immediate neighbour and interchanged if it is out of sequence. The process is repeated to a distance up the data which depends on the previous number of passes; the underlined digit represents the top limit in each pass. With n items of data, a maximum of n + (n-1) + (n-2) + ... passes is required, making about ½n² in all. On this basis it is often said that the bubble sort takes time proportional to the square of the number of items to be sorted. However, the correct time is very sensitive to partial ordering of the data. The graph at the end of SORT shows that new items, added to an already sorted array, then bubble sorted together, is very fast; in fact, under these circumstances, the bubble sort is correctly one of the fastest possible, since it does little more than check that each item is correctly related to its neighbour, which is necessary in any sorting system. The machine-code sort operates on string arrays, changing the pointers where appropriate, and using the identical comparison to that of BASIC, for consistency. It does not sort the zeroth element, which can therefore be used as a title or reminder. If new items are to be sorted in, keep a number of null or blank elements at the start of the array. As the diagram illustrates, high values (e.g. 6) can rise quickly from the bottom, but low values (e.g. 1) are slow in descending. Note finally that the machine-code can be made to sort from the second, third, ... characters of the string, rather than the first, by changing $FF in $032E (BASIC 1), or $7FB6 (BASIC>1) to 0 (second), 1 (third),... A demonstration BASIC routine is provided with the machine-code. Of the other sorts, the Shell-Metzner and Quicksort are well-known; the former performs many small bubble sorts on longitudinal subsets of the data; the latter compares data with a 'pivot value', putting the result into one or other 'stack' depending on the result. It may run out of space; if so, dimension the array in line 40 with a larger value. The 'scatter' sort is an attempt to mimic human sorting: a subsidiary array is used, into which data is first roughly sorted, on some a priori basis, for example with the As at the beginning, Zs at the end, and others in between. Then this array is sorted thoroughly. Its use of RAM is too great to permit the method to be very useful on micros.

Dealer inquiries are invited.

"Your book is EXCELLENT!"

JIM STRASMA,
From a letter to the author.

A **COMPUTE! Books** Publication
In conjunction with Level Limited.

simple nor as clean as either of the above, but it can be very useful in certain circumstances. Suppose you plan to make a display using the lowercase character set, but you find that you also need certain characters from the graphics set. You can have only one set or the other enabled at one time, so what do you do? You can (a) change out the character-generator ROM; (b) change your mind about what you really wanted to do to begin with; or (c) give up computers and start a farm. Well, consider this: the PET IRQ's are exactly synchronized with the vertical retrace interval of the CRT. You could make use of this fact to write a little routine which grabs the IRQ's, enables one character set for a while, and then enables the other character set and allows the PET to go about its business. In this way, the top N lines of the CRT can be made to display characters from a different set from what the rest of the screen displays.

Phantom Windows

I have worked out the numbers on this for the 8032. The screen is updated 60 times a second, and each update takes exactly 12 ½ milliseconds. There are 250 scan lines, so each line takes exactly 50 microseconds. Of this, ten microseconds are consumed in the horizontal retrace, so 40 microseconds of character output are provided for. This works out to exactly 500 nanoseconds per character-width of beam scan (if these numbers sound awfully exact, it's because they are – this is one *stable* display).

The difference between 12½ milliseconds and 1/60th of a second is a little less than 4.2 milliseconds. The PET apparently takes its IRQ in the dead center of the vertical retrace, so you have to blow away a couple of milliseconds right off the bat before the beam even turns on again at the top of the screen. From that point on, each additional 50-microsecond interval that you wait before switching the character set back gives you one additional display scan line using the other set. Here are the kinds of routines you need:

Twin-Screen Disassembly

```
ADDR  OBJECT     ASSEMBLY SOURCE

027A                       ORG  634          START OF TAPE-1 BUFFER
027A              DUALON   EQU  *            TURN ON DUAL CHAR MODE
027A  A5 90                LDA  $90          SAVE CURRENT IRQ VEC
027C  8D CA 02             STA  HIVEC          IN HOLDING REGISTER
027F  A5 91                LDA  $91
0281  8D CB 02             STA  HIVEC+1
0284  78                   SEI               MASK INTERRUPTS
0285  A9 95                LDA  #<DUAIRQ     POINT IRQ VEC AT DUAIRQ
0287  85 90                STA  $90
0289  A9 02                LDA  #>DUAIRQ
028B  85 91                STA  $91
0289D 58                   CLI               UNMASK INTERRUPTS
028E  60                   RTS
028F              DUAOFF   EQU  *            TURN OFF DUAL CHAR MODE
028F  78                   SEI               MASK INTERRUPTS
0290  20 BF 02             JSR  RSIRQV       RESTORE ORIG IRQ VEC
0293  58                   CLI               UNMASK INTERRUPTS
0294  60                   RTS

0295              DUAIRQ   EQU  *            DUAL CHAR MODE IRQ RTN
0295  A5 97                LDA  $97          STOP KEY PRESSED?
0297  C9 03                CMP  #3
0299  D0 06                BNE  NOSTOP       NO, CONTINUE
029B  20 BF 02             JSR  RSIRQV       YES, RESTORE ORIG VEC
029E  6C 90 00             JMP  ($90)        GO ABOUT BUSINESS
02A1  A9 0E       NOSTOP   LDA  #14          USE LOWER CASE CHAR SET
02A3  8D 4C E8             STA  59468
02A6  A0 07                LDY  #7           DELAY 36 USEC TO GET
02A8  88          IDELAY   DEY                 IN SCAN-LINE SYNC
02A9  D0 FD                BNE  IDELAY
02AB  A2 47                LDX  #41+30       THREE FULL LINES
02AD  A0 08       NXTSCN   LDY  #8           DELAY 50 USEC PER SCAN
02AF  EA                   NOP
02B0  EA                   NOP
02B1  88          SDELAY   DEY
02B2  D0 FD                BNE  SDELAY
02B4  CA                   DEX               DELAYED LONG ENOUGH?
```

continued on page 171

```
02B5   D0 F6               BNE  NXTSCN       NO, DELAY ANOTHER SCAN
02B7   A9 0C               LDA  #12          YES, USE GRAPHICS NOW
02B9   8D 4C E8            STA  59468
02BC   6C CA 02            JMP  (HIVEC)      DO REGULAR IRQ THING
02BF               RSIRQV  EQU  *            RESTORE IRQ VEC ROUTINE
02BF   AD CA 02            LDA  HIVEC        DO IT TO IT
02C2   85 90               STA  $90
02C4   AD CB 02            LDA  HIVEC+1
02C7   85 91               STA  $91
02C9   60                  RTS
02CA               HIVEC   EQU  *            HOLDING REG FOR IRQ VEC
```

Unfortunately, it is impossible for a facility like this to be made position-independent on the 6502. In fact, because of the critical nature of the timing loops, you must take care if you do relocate it to ensure that none of these loops cross memory page boundaries; otherwise, they will take longer than they are supposed to and the routines will not work properly.

As given here, a SYS to 634 will cause the top three print lines of an 8032 display to appear in the lowercase character set, while the rest of the screen appears in the graphics set. The argument to the LDX at $02AB determines the number of video scan lines of delay used: setting this to (decimal) 42 will give you exactly one scan line of lowercase if the display is in the uncompressed mode, 43 will give you two scans, etc. Use 41 as your delay base in the uncompressed mode, and add ten scan lines per print line of delay. In the compressed display mode, the delay base is 67, and you should add only eight scan lines per print line.

If you would rather have graphics in the upper part of the screen and lowercase toward the bottom, simply interchange the 14 at $02A2 with the 12 at $02B8. The STOP-key check from $0295 to $029E is there just so that if you are calling this thing from BASIC or RPL, you don't have to worry about the display staying in this weird mode if you abort execution via the STOP key. Normally, your program should SYS to DUAOFF at 655 to restore the display to normal when exiting.

Naturally, the above madness cannot be used verbatim on a 40-column machine, but with a little effort you should be able to coerce it into working. Keep in mind that Upgrade machines, and some (but not all) 4.0 40-column machines stick a four into location $97 when the STOP key is hit, instead of a three.

Program 1. LOADER For Dual Screen Routine

```
634 DATA 165, 144, 141, 202, 2, 165
640 DATA 145, 141, 203, 2, 120, 169
646 DATA 149, 133, 144, 169, 2, 133
652 DATA 145, 88, 96, 120, 32, 191
658 DATA 2, 88, 96, 165, 151, 201
664 DATA 3, 208, 6, 32, 191, 2
670 DATA 108, 144, 0, 169, 14, 141
676 DATA 76, 232, 160, 7, 136, 208
682 DATA 253, 162, 71, 160, 8, 234
688 DATA 234, 136, 208, 253, 202, 208
694 DATA 246, 169, 12, 141, 76, 232
700 DATA 108, 202, 2, 173, 202, 2
706 DATA 133, 144, 173, 203, 2, 133
712 DATA 145, 96
800 FOR ADRES=634TO713:READ DATTA:X=X+DATTA:PO
    KE ADRES,DATTA:NEXT ADRES
801 IFX<>10264THENPRINT"ERROR IN DATA STATEMEN
    TS"
```

©

GTIA Demonstration

Jerry White
Levittown, NY

This GTIADEMO program* won't work properly on a machine that has the CTIA chip. However, if you have the GTIA, this demo will provide a colorful and animated display you won't soon forget.

At first glance, you may be reluctant to enter such a large program just for a demo. For this reason, I have separated the three independent modules with REMarks. The first two modules do not require very much typing at all. I suggest you enter each module separately, then LIST them onto disk or cassette. When you RUN the first module, you just might find the incentive to continue typing.

The third module is, in my opinion, the most impressive to watch. Admittedly, it does take about a minute and a half before the animation begins. I'm quite sure you will find the effort required to enter this program, and the brief delay, to be justly rewarded.

Module three of the demo program uses Graphics 10. In this mode, we have nine hues and luminances. In the demo, a small assembler subroutine is used to shift values in the color registers. This provides a hypnotic, animation effect. To end the program, simply press any key.

**Software adapted by permission of Atari, Inc.*

```
100 GOSUB 770:REM IDENTIFICATION DISPLAY
    (VERSION DATE 4/2/82 JERRY WHITE)
110 REM GTIADEMO MODULE ONE
120 JW=15:GRAPHICS 9:SETCOLOR 4,JW,0
130 FOR Y=55 TO 0 STEP -10:FOR X=0 TO 24
    :C=X:IF X>11 THEN C=24-X
140 C=C+3:Z=Y+(X):D=INT(SQR(144-(X-12)*(
    X-12))/2:COLOR 15-C:PLOT Z,Y+7-D:DR
    AWTO Z,Y+7+D:COLOR C:DRAWTO Z,180-Y+
    D
150 NEXT X:NEXT Y
160 JW=JW-1:IF JW=0 THEN 250
170 SETCOLOR 4,JW,0:READ P
180 FOR ME=8 TO 0 STEP -0.5:SOUND 0,P,10
    ,ME:SOUND 1,P-1,10,ME:SOUND 2,P+1,10
    ,ME:SOUND 3,P+2,10,ME:NEXT ME
190 POKE 540,30
200 IF PEEK(540)<>0 THEN 200
210 GOTO 160
220 REM DATA FOR SOUND ROUTINE
230 DATA 243,230,217,204,193,182,173,162
    ,153,144,136,128,121,60
240 REM GTIADEMO MODULE TWO
250 GRAPHICS 11:DIM ML$(21):FOR ME=1 TO
    21:READ IT:ML$(ME,ME)=CHR$(IT):NEXT
    ME
260 REM DATA FOR MACHINE LANGUAGE SHIFT
    COLOR ROUTINE
270 DATA 104,162,0,172,193,2,189,194,2,1
    57,193,2,232,224,8,144,245,140,200,2
    ,96
280 FOR I=1 TO 8:READ A:POKE 704+I,A+224
    :NEXT I
290 REM DATA FOR COLOR REGISTER POKES
300 DATA 2,4,6,8,6,4,2,2
310 FOR I=0 TO 38:COLOR Q:X=I:Y=I*2:PLOT
     X,Y
320 DRAWTO 79-X,Y:PLOT X,Y+1:DRAWTO 79-X
    ,Y+1:DRAWTO 79-X,190-Y
330 DRAWTO X,190-Y:PLOT 79-X,190-Y+1:DRA
    WTO X,190-Y+1:DRAWTO X,Y
340 Q=Q+1:IF Q>8 THEN Q=1
350 NEXT I:JW=0:ME=JW
360 X=USR(ADR(ML$)):SOUND 0,JW,2,2:SOUND
     1,16-JW+10,12,2:SOUND 2,JW,0,2
370 JW=JW+1:IF JW>15 THEN JW=0:ME=ME+1
380 SOUND 0,0,0,0:SOUND 1,0,0,0:SOUND 2,
    0,0,0
390 IF ME>12 THEN 420
400 SETCOLOR 1,JW,6:GOTO 360
410 REM GTIADEMO MODULE THREE
420 GRAPHICS 10:JW=-2:FOR ME=705 TO 712:
    JW=JW+18:POKE ME,JW:NEXT ME

430 NUM=1:FOR Y=0 TO 191:COLOR NUM:PLOT
    0,Y:DRAWTO 79,191-Y:NUM=NUM+0.4167:I
    F NUM>8 THEN NUM=1
440 NUM=NUM+1:NEXT Y
450 FOR X=79 TO 0 STEP -1:COLOR NUM:PLOT
     X,0:DRAWTO 79-X,191:NUM=NUM+1:IF NU
    M>8 THEN NUM=1
460 NEXT X
470 JW=-2:FOR ME=705 TO 712:JW=JW+18:POK
    E ME,JW:NEXT ME
480 FOR ME=0 TO 359 STEP 2:NUM=8:READ X,
    Y:COLOR 0:PLOT X,Y:IF ME<181 THEN 52
    0
490 FOR JW=1 TO 45:LOCATE X,Y+JW,IT:IF I
    T=0 THEN POP :GOTO 520
500 COLOR NUM:PLOT X,Y+JW:NUM=NUM-1:IF N
    UM<1 THEN NUM=8
510 NEXT JW
520 NEXT ME:KULR=10
530 JW=USR(ADR(ML$)):SOUND 0,0,0,0:SOUND
     1,0,0,0:SOUND 2,0,0,0:SOUND 3,0,0,0
540 IF PEEK(764)<>255 THEN 580
550 P=P+5:IF P<249 THEN SOUND 0,P,10,2:S
    OUND 1,P+2,10,2:SOUND 2,P+4,10,2:SOU
    ND 3,P+6,10,2:GOTO 530
560 P=0:KULR=KULR+1:IF KULR>15 THEN KULR
    =1
570 SETCOLOR 1,KULR,6:GOTO 530
580 GRAPHICS 0:POKE 764,255:? :? "END OF
     GTIA DEMONSTRATION":? :END
590 DATA 60,96,60,97,60,98,60,99,60,99,6
    0,100,60,101,60,101,60,102,60,103
600 DATA 59,103,59,104,59,105,58,105,58,
    106,58,106,57,107,57,108,57,108,56,1
    09
610 DATA 56,109,55,110,55,110,54,111,54,
    111,53,112,53,112,52,113,52,113,51,1
    13
620 DATA 50,114,50,114,49,114,49,115,48,
    115,47,115,47,116,46,116,45,116,45,1
    16
```

```
630 DATA 44,116,43,116,43,116,42,116,41,
    116,40,116,40,116,39,116,38,116,38,1
    16
640 DATA 37,116,36,116,36,116,35,116,34,
    116,34,115,33,115,32,115,32,114,31,1
    14
650 DATA 31,114,30,113,29,113,29,113,28,
    112,28,112,27,111,27,111,26,110,26,1
    10
660 DATA 25,109,25,109,24,108,24,108,24,
    107,23,106,23,106,23,105,22,105,22,1
    04
670 DATA 22,103,21,103,21,102,21,101,21,
    101,21,100,21,99,21,99,21,98,21,97
680 DATA 20,96,21,96,21,95,21,94,21,94,2
    1,93,21,92,21,92,21,91,21,90
690 DATA 22,90,22,89,22,88,23,88,23,87,2
    3,87,24,86,24,85,24,85,25,84
700 DATA 25,84,26,83,26,83,27,82,27,82,2
    8,81,28,81,29,80,29,80,30,80
710 DATA 31,79,31,79,32,79,32,78,33,78,3
    4,78,34,77,35,77,36,77,36,77
720 DATA 37,77,38,77,38,77,39,77,40,77,4
    0,76,41,77,42,77,43,77,43,77
730 DATA 44,77,45,77,45,77,46,77,47,77,4
    7,78,48,78,49,78,49,79,50,79
740 DATA 50,79,51,80,52,80,52,80,53,81,5
    3,81,54,82,54,82,55,83,55,83
750 DATA 56,84,56,84,57,85,57,85,57,86,5
    8,87,58,87,58,88,59,88,59,89
760 DATA 59,90,60,90,60,91,60,92,60,92,6
    0,93,60,94,60,94,60,95,60,96
770 GRAPHICS 17:POSITION 6,6:? #6;"GUAR 
    DEXC"
780 ? #6:? #6:? #6;"{3 SPACES}BY JERRY W
    HITE":POKE 764,255:POKE 710,154:POKE
     708,14:POKE 711,74:FOR JW=1 TO 100:
    NEXT JW
790 ? #6:? #6:? #6;"{4 SPACES}press any
    key":? #6:? #6:? #6;"{6 SPACES}to be
    gin":BLINK=0
800 BLINK=BLINK+1:IF BLINK>20 THEN POKE
    709,0:BLINK=-20:POKE 53279,0
810 IF BLINK=0 THEN POKE 709,234:POKE 53
    279,0
820 IF PEEK(764)=255 AND PEEK(53279)>6 T
    HEN 800
830 POKE 764,255:RETURN
```

©

For Upgrade and 4.0 BASIC PET/CBM's of any memory size, this program can be entered and used whether or not you know machine language. It's a handy utility which instantly deletes any line ranges from within a BASIC program.

PET Machine Language Delete

Roger Burrows
Ottawa, Ontario

If you never make mistakes or revise programs, you won't need this tool. However, if you're like me you'll find this fast line delete program very useful. With it, you can delete a range of lines (or a single line) by a few keystrokes. And it won't take more than a second.

The "BASIC loader" program here will enter the program into the computer wherever you wish. If you RUN it as presented, it loads into the upper memory of an 8K PET. To "protect" it from being overwritten by a BASIC program, you should then type: POKE 53,31. If you have 16K of memory, substitute the following in line 20: FOR I = 16208 TO 16381 (etc...) and then use POKE 53,63 to protect. Finally, for 32K machines, substitute in line 20: FOR I = 32592 TO 32765 and use POKE 53,127.

The program is *relocatable* because the machine language code has no internal JMPs or JSRs or other self-references to particular addresses within its boundaries. By changing the POKE loop in line 20, you can send it wherever you want to within RAM memory. Program 1 is for Upgrade BASIC PET/CBM's. Substitute the lines in Program 2 if your computer uses 4.0 BASIC. Also note the checksum change in line 10 for 4.0 BASIC.

When you want it to delete lines from a BASIC program, type SYS 8016 (or SYS 16208 or 32592, whatever your starting address is in line 20) plus the start and end line numbers. For example:

SYS 8016,120,180

would delete lines 120 through 180. To delete a single line, you can use SYS 8016,120.

Program 1. Upgrade Version

```
10 CK = 21339: REM CHANGE TO 22051 FOR 4.0
     BASIC
20 FOR I = 8016 TO 8189: READ X: Y=Y+X: PO
     KEI,X: NEXTI
100 DATA 32,118,0,240,109,32,112,0,240,104,
     176,103,32,115,200
110 DATA 165,17,133,178,133,180,165,18,133,
     179,133,181,32,118,0
120 DATA 240,18,32,112,0,240,13,176,76,32,1
     15,200,165,17,133
130 DATA 180,165,18,133,181,165,178,133,17,
     165,179,133,18,32,44
140 DATA 197,165,92,133,182,165,93,133,183,
     165,180,133,17,165,181
150 DATA 133,18,32,44,197,144,12,160,0,177,
     92,133,184,200,177
160 DATA 92,24,144,6,165,92,133,184,165,93,
     133,185,197,183,144
170 DATA 8,208,10,165,182,197,184,144,4,96,
     76,3,206,162,1
180 DATA 160,0,177,184,145,182,208,7,232,22
     4,3,176,13,144,2
190 DATA 162,0,200,208,238,230,185,230,183,
     208,232,152,101,182,133
200 DATA 42,165,183,105,0,133,43,160,171,32
     ,86,241,160,174,32
210 DATA 86,241,32,66,196,76,57,196,0,0,0,0
     ,0,0,0
220 IF CK <> Y THEN PRINT "ERROR IN DATA ST
     ATEMENTS"
```

Program 2. Substitutions For 4.0 BASIC

```
100 DATA 32,118,0,240,109,32,112,0,240,104,
     176,103,32,246,184
120 DATA 240,18,32,112,0,240,13,176,76,32,2
     46,184,165,17,133
130 DATA 180,165,18,133,181,165,178,133,17,
     165,179,133,18,32,163
140 DATA 181,165,92,133,182,165,93,133,183,
     165,180,133,17,165,181
150 DATA 133,18,32,163,181,144,12,160,0,177
     ,92,133,184,200,177
170 DATA 8,208,10,165,182,197,184,144,4,96,
     76,0,191,162,1
200 DATA 42,165,183,105,0,133,43,160,171,32
     ,133,241,160,174,32
210 DATA 133,241,32,182,180,76,173,180,0,0,
     0,0,0,0,0
```

©

To thoroughly document your computer's BASIC or operating system (or any significant machine language program), you need to create a commented map of the routines. "Resource" is a collection of BASIC programs which, working together, help you to produce annotated disassemblies.

Last month **COMPUTE!** *published explanatory text and the first program. "Resource" now concludes with the rest of the programs and some example results. The author used "Resource" to aid in generating annotated cross-reference lists for the OSI version of Microsoft's BASIC.*

Resource: Part II

Mapping Machine Language Code

T. R. Berger, Coon Rapids, MN

The tables which accompany these final programs comprising "Resource" are selections from annotated cross reference lists for OSI-Microsoft 8K disk BASIC from 0S65D V3.2 NMHZ disks. The tables were produced by using "Resource" and the annotations derive from both Jim Butterfield's memory maps (**COMPUTE! II**, June/July 1980) and my maps of OS65D (**COMPUTE!**, January-March 1981).

All addresses within the example tables are in hex and the first address on any line is the called address. Thereafter, the addresses refer to the place where the *calling* code resides. In addition, many of the addresses have preceding letters. These letters mean different things in different tables. In a JMP or JSR table, an M means the calling code is a JUMP instruction.

An S means the calling code is a JUMP TO SUBROUTINE instruction. In the MEMORY table, the letter is always the first letter of the calling opcode. For example,

1DF3 STA $0100,Y

is referenced in the table beside 0100 as S1DF3. The Zpage table has no leading letters. This table was produced by an early version of "Resource," before the extra information was added.

Table 1. Keyword Action Addresses

Word	Token	Address						
END	80	082A	TO	9D		TAN	B9	1FF2
FOR	81	0748	FN	9E		ATN	BA	2056
NEXT	82	0C4B	SPC	9F		PEEK	BB	1688
DATA	83	08F9	THEN	A0		LEN	BC	15F6
INPUT	84	0B2C	NOT	A1	1E88	STR$	BD	12E9
DIM	85	0F24	STEP	A2		VAL	BE	1627
READ	86	0B58	+	A3	16D9	ASC	BF	1605
LET	87	09A6	−	A4	16C2	CHR$	C0	1566
GOTO	88	08A6	*	A5	18F4	LEFT$	C1	157A
RUN	89	087E	/	A6	1A0D	RIGHT$	C2	15A6
IF	8A	0929	^	A7	1E4F	MID$	C3	15B1
RESTORE	8B	080A	AND	A8	0E8C	NF	C4	ERROR
GOSUB	8C	0889	OR	A9	0E89	SN	C5	ERROR
RETURN	8D	08D3	>	AA		RG	C6	ERROR
REM	8E	093C	=	AB		OD	C7	ERROR
STOP	6F	0828	<	AC		FC	C8	ERROR
ON	90	094C	SGN	AD	1B34	OV	C9	ERROR
NULL	91	086D	INT	AE	1BC7	OM	CA	ERROR
WAIT	92	169C	ABS	AF	1B53	US	CB	ERROR
EXIT	93	223C	USR	B0	22F7	BS	CC	ERROR
DISK	94	2253	FRE	B1	1204	DD	CD	ERROR
DEF	95	1235	POS	B2	1225	/O	CE	ERROR
POKE	96	1693	SQR	B3	1E45	ID	CF	ERROR
PRINT	97	21AA	RND	B4	1F66	TM	D0	ERROR
CONT	98	0853	LOG	B5	18B3	LS	D1	ERROR
LIST	99	06B9	EXP	B6	1EC1	ST	D2	ERROR
CLEAR	9A	067C	COS	B7	1FA2	CN	D3	ERROR
NEW	9B	0662	SIN	B8	1FA9	UF	D4	ERROR
TAB	9C							

Table 2. Memory Table

```
.       Zpage
0001    L173A
0002    L1733
0003    L172C
0004    L1725
0016    S05EF L05F2 S0612
0017    L0512
0018    S062B
00A0    B0E8B
00A2    B0909
00FF    S1CF6 S1D67 S1D70 S1DB4 S1DBE L1DD1 S1E
       0F
.
.       Stack
0100    S1DF3 S1E14
0101    L03A6 L107B S1086 S1DEE
0102    L03B1 C03C2 L0FC9 L1077 S1081 S1E05
0103    L03B6 C03BB S1E01
0104    S1E0A
0109    L0C75 S0C8B
010F    L0C90
0110    L0C95
0111    L0C9F
0112    L0C9A
01DE    L0E79
01DF    L0E7E
.
.       Start of keyword address table
0200    L07F9
0201    L07F5
.
.       Start of operator hierarchy and address
       table
0266    C0D20 C0D48 L0D64
0267    L0D53
0268    L0D4F
.
.       Table of BASIC keywords (Start $0284)
0283    L061D
0284    S05E0 L0622 L0736 L073E
.
.       Error messages
0364    L0456
0365    L045C
.
.       BIT hiding code
07A9    B057C
08A2    B10CF
0EA2    B08E3
1410    B19BE
.
.       Constants
1E21    A1D91
1E22    A1D8A
1E23    A1D83
1E24    A1D7C
.
.       Operand pointing to IO flags
21D5    S2104
21DA    S210A
.
.       Stack pointer
226F    S211F
.
.       Table index for OS buffer write routine
228A    S217F
.
.       Buffer read write data for OS
22C8    L22E2
22C9    L22D6
22CA    L22DC
.
.       USR pointer to OS and disk
22F2    S22D9
22F3    S22DF
.
.       OS Input flag
2321    S20F5 L2101 S21D6 S2201 L2215
.
.       OS Output flag
2322    S20F8 L2107 S215D S21DB L21FE S2208
.
.       OS Passed char. (Control C)
2325    L0819 S0823
.
.       OS Disk sector number
265E    S22AC
.
.       BIT hiding code
28A9    B0E0F
.
.       OS Default IO flag
2AC6    L20F2
.
.       BIT hiding code
2CA9    B0E12
.
.       OS Read buffer pointer
2CE5    S2142
.
.       OS End of buffer on read
2CED    S2113
.
.       Transient GET and PUT pointer
2E7A    L22A6
.
.       OS Swapped value ($E1,$E2) Start pointer
      for buffer read
305A    S2116
305B    S2119
.
.       Pointer to SOURCE File header
3178    S2126 L2273
.
.       Number of tracks in SOURCE File
317D    S2136
.
.       BIT hiding code
3FA9    B0AEB
A4A2    B1AC4
```

Table 3. Zpage Table

```
. ;Index for Zpage, Jump vectors for BASIC
00   05AD 05D9 0609 0627 1357
01   135B 170D 173D 1832 1838 1848 184C 184E
   1850
02   135F 1736 182E 1834 185A 185C 185E
03   172F 182A 1830 1866 1868 186A
04   1728 1826 182C 1872 1874 1876
.
```

(Table 3. continued)

```
.  ;Search Character
0A  1320 1BDF 1E70 1EE2
0A  090C 0914 0916 0976 0998 0B95 0B9D 0E97
    0EB2 130D
.
.  ;Scan between quotes FLAG
0B  130F 1324
0B  05B3 0607 060D 0910 0912 0918 091E 0BA2
    0E9D 0EA9
.
.  ;POINTER: Input Buffer, # of subscripts
0C  0EAB 0EB0 0EB4 1028 1091 10DE 1108 112F
    1175 11AD
0C  049C 04F7 050F 05D1 05E9 061A 0E8E 0E95
    0E9B 0EA7
.
.  ;Default DIM FLAG
0D  0F33 1059 109E 10D7 1113 116E
.
.  ;Type: FF=string 00=numeric
0E  09B5 0A4D 0B91 0CBF 0D10 0D36 0DB4 0E31
    0ED3 0F44
0E  0F63 105E 1097 1204 121A 1368 1601
.
.  ;
0F  09B2 0BBC 0E36 0F46 0F71 105B 109A
.
.  ;FLAG: DATA scan; LIST quote; memory
10  05AB 05B9 0600 1372 1396 139F
.
.  ;Subscript FLAG; FNx FLAG
11  06A8 074A 0F6B 0F81 0F8C 1240 126A
.
.  ;0=Input; $40=GET; $98=READ
12  0B0D 0B5E 0B80 0C12
.
```

Table 4. JMP and JSR Table

```
.     Jump vector for commands
0003  S047A
.
.     Jump vector for evaluation
006F  M0D84
.
.     Jump vector for functions
00A1  S0E83
.
.     CHRGET subroutine:  get BAISC character
00C0  S1615 S1C05 S1C12 S1C35 S2163 M220B
      S2259
00C0  S0484 S06D0 S079F S07E0 M07FD S0960
      S09A0 S0AC1 S0B8E
00C0  S0CB3 S0D00 S0DB6 M0E1B S0E4D S0F48
      S0F53 S0F7B S103D
.
.     Subentry: get previous character
00C6  S1652 S16A3 M2160 S21B0 S21EB
00C6  S0CAC S0CE7 S0F28 S0F30 S0F37 S108A
      S12C5 S15B5 M1624
00C6  S06C7 S0798 S089D S092C S0941 S0A32
      S0B7B S0BC1 S0BDD
.
.     Search stack for FOR and GOSUB activity
03A1  S074F S08D9 S0C58
.
.     Open space in memory
03CF  S0504 S0FF1
03D6  S14A2
.
.     Test stack depth
0412  S075D S088B S0CDD
.
.     Check available memory
041F  S03CF S10EC S1142
.
.     Send error message then:
044C  M1194
044E  M1A87
044E  M085B M08E6 M0CCA M0E20 M10D2 M1232
      M1352 M14D4 M1821
.
.     Warm start BASIC
```

continued on p. 178

```
.PASS 2
.       RESOURCE 2 ** SOURCE AND EQUATE
        FILE BUILDER **
.
.               TWO BUFFERS REQUIRED
.
.
.
.
.
.
.
10 REM *** RESOURCE 2 - SOURCE AND EQUATE
   FILE BUILDER ***
20 REM T.R. BERGER 11/80
.
.  ** REMOVE COMMA AND SEMICOLON **
50 POKE2972,13:POKE2976,13
.
60 PRINT:PRINT"RESOURCE ** STEP 2 - BUILD
   SOURCE & EQUATE FILE **"
70 PRINT:PRINT
80 INPUT"SCRATCH FILE";SF$
90 INPUT"RESOURCE FILE";OF$
100 PRINT:INPUT"SYMBOL FILE";FS$
110 INPUT"EQUATE FILE";EF$
.
140 SP$="        "
.
.  ** COUNT SYMBOLS **
.
180 DISK OPEN,6,FS$
.
.  * SYMBOL COUNTER *
200 SN=-1
210 INPUT #6,IN$
220 IF IN$="XIT" THEN250
230 SN=SN+1
240 GOTO210
250 DISK CLOSE, 6
.
.  ** LOAD SYMBOLS **
.
260 DISK OPEN,6,FS$
.
.  * DIMENSION STRING AND MARKER ARRAYS *
280 DIM SS$(SN), SS(SN)
300 FOR I=0 TO SN
310 INPUT #6, SS$(I)
320 NEXT I
330 DISK CLOSE,6
.
.  ** MAIN PROGRAM **
.
.  * LINE NUMBERS AND INCREMENT *
380 CL=10000: IN=10
390 DISK OPEN,6,SF$
400 DISK OPEN,7,OF$
.
.  * LOOP BACK HERE *
410 INPUT #6,IN$
420 IFIN$="XIT"THEN640
.
.  * GET ADDRESS OF LINE *
440 A1$=LEFT$(IN$,4)
.
.  ** BINARY SEARCH FOR SYMBOL **
.
   * SEARCH *
470 L=0: R=SN
480 M=INT((L+R)/2)
.
.  * EXIT HERE IF NOT FOUND *
490 IF L>R THEN OU$=SP$+MID$(IN$,5): GOTO
    580
.
.  * EXIT HERE IF FOUND *
500 IF A1$=SS$(M) THEN 550
510 IF A1$>SS$(M) THEN L=M+1: GOTO480
520 R=M-1: GOTO480
.
.  * END OF SEARCH *
.
.  * CREATE SYMBOL AND MARK ADDRESS USED
   *
550 SS(M)=1: OU$="HH"+IN$
.
.  * CREATE RESOURCE LINE *
580 OU$=STR$(CL)+" "+OU$
.
.  * INCREMENT LINE NUMBER *
600 CL=CL+IN
.
.  * PRINT LINE *
610 PRINT #7,OU$
620 PRINTOU$
630 GOTO410
.  * LOOP BACK FROM HERE *
.
.  * CLOSE FILES *
640 PRINT #7,IN$
650 DISK CLOSE,7
660 DISK CLOSE,6
.
.  * END OF MAIN PROGRAM *
.
.  * WRITE TWO BYTE EQUATES *
710 DISK OPEN,7,EF$
.
.  * FIRST LINE NUMBER *
720 CL=5000
.
.  * TITLE *
730 PRINT #7,STR$(CL)+" ;EQUATES"
740 CL=CL+IN
.
.  * COUNTER FOR EQUATES *
750 K=0
.
.  * PRINT EQUATES *
760 FORI=0TOSN
.
.  * SKIP SYMBOLS WHICH ARE LABELS *
770 IF SS(I)=1 THEN 830
780 A1$=STR$(CL)+" HH"+SS$(I)+" = $"+SS$(
    I)
790 PRINT #7, A1$
800 PRINT A1$
.
.  * NEXT LINE NUMBER *
810 CL=CL+IN
.
.  * INCREMENT EQUATES COUNT *
820 K=K+1
830 NEXTI
```

```
840 PRINT #7,"XIT"
850 DISK CLOSE,7
.
.  * FINISHED WITH EQUATES *
870 PRINT:PRINT,
880 PRINT"CODE SOURCE FILE REGENERATED":
    PRINT
890 PRINT TAB(10) "RESOURCE FILE:  "OF$
900 PRINTTAB(10) "EQUATES FILE: ";EF$
910 PRINT:PRINTTAB(10)"SCRATCH FILE: "SF$
920 PRINTTAB(10) "SYMBOL FILE:  "FS$
930 PRINTTAB(9) SN+1" SYMBOLS"
940 PRINTTAB(9) K" EQUATES"
950 PRINT:PRINT"PASS 2 COMPLETE"
960 PRINT:PRINT:END
```

```
.PASS 3
.
.     RESOURCE 3 ** CROSS REFERENCE FILES
      **
.
.           TWO BUFFERS REQUIRED
.
.
.
.
.
.
100 REM *** RESOURCE 3 - CROSS REFERENCE
    BUILDER ***
110 REM *** T.R. BERGER 11/80
.
.  * DELETE COMMA AND SEMICOLON *
120 POKE2972,13: POKE2976,13
.
130 PRINT:PRINT"RESOURCE ** STEP 3 -
    CROSS REFERENCE GENERATOR"
140 PRINT:PRINT
150 PRINTTAB(20)"TYPES OF REFERENCES"
160 PRINT:PRINT"B"TAB(10)"BRANCH"
170 PRINT"J"TAB(10)"JSR AND JMP"
180 PRINT"M"TAB(10)"MEMORY"
190 PRINT"Z"TAB(10)"ZPAGE"
.
200 PRINT:INPUT"YOUR CHOICE (J/B/M/Z)";
    CR$
210 IFCR$<>"B"ANDCR$<>"J"ANDCR$<>"M"AND
    CR$<>"Z"THEN200
.
220 PRINT:INPUT"SCRATCH FILE";SF$
230 INPUT"REFERENCE FILE";RF$
240 PRINT:INPUT"NUMBER OF REFERENCES";NR
.
.  * DIMENSION ARRAYS *
250 DIM SS$(NR), SA$(NR), V(NR)
.
.  * SET SYMBOL NUMBER AND TYPE *
260 T=1: SN=-1
270 IF CR$="M" THEN T=2
280 IF CR$="Z" THEN T=3
.
.  * SYMBOL PLUCKER *
290 S=13:NL=4
300 IF CR$="Z" THEN S=15: NL=2
.
.  ** MAIN PROGRAM **
.
310 DISK OPEN,6,SF$
320 DISK OPEN,7,RF$
.
.  * LOOP BACK HERE *
330 INPUT #6,IN$
340 IF IN$="XIT" THEN600
.
.  * TOO SHORT, NO SYMBOL *
350 IF LEN(IN$)<16 THEN330
.
.  * CHECK FOR NO SYMBOL *
360 IF MID$(IN$,11,2)<>"HH" THEN330
.
.  * DISPLAY LINE WITH SYMBOL *
370 PRINT IN$
.
.  * DETERMINE SYMBOL TYPE *
380 ON T GOSUB740 ,790,860
.
.  * CHECK FOR RELEVANT SYMBOL *
390 IF FL=0 THEN330
.
.  * GET ADDRESS OF LINE *
400 A1$=M$+LEFT$(IN$,4)
.
.  * GET SYMBOL *
410 A2$=MID$(IN$,S,NL)
.
.  * SEARCH SYMBOL TABLE *
.  * BINARY SEARCH *
.
420 L=0: R=SN
.
.  * SYMBOL NOT FOUND, INSERT IT *
430 IF L>R THEN480
440 M=INT((L+R)/2)
.
.  * SYMBOL IN TABLE *
450 IF A2$=SS$(V(M)) THEN540
460 IF A2$>SS$(V(M)) THEN L=M+1: GOTO430
470 R=M-1: GOTO430
.
.  * ADD A SYMBOL *
480 SN=SN+1: SS$(SN)=A2$
.
.  * POINT TO ITS PROPER POSITION IN
   ORDERING *
490 IF L=SN THEN530
500 FOR I=SN-1 TO L STEP -1
510 V(I+1)=V(I)
520 NEXT I
530 V(L)=SN: M=L
.
.  * ADD A CROSS REFERENCE *
540 SA$(V(M))=SA$(V(M))+" "+A1$
.
.  *  CHECK IF CROSS REF LINE TOO LONG *
550 IF LEN(SA$(V(M)))<50 THEN330
.
.  * PRINT CROSS REF LINE *
560 PRINT #7, SS$(V(M))+" "+SA$(V(M))
570 PRINT SS$(V(M))+" "+SA$(V(M))
580 SA$(V(M))=""
590 GOTO330
.  * LOOP BACK FROM HERE *
.
.  * CLOSE SCRATCH FILE *
```

```
600 DISK CLOSE,6
.
.  * PRINT REMAINING CROSS REF LINES *
610 FOR I=0 TO SN
620 IF SA$(V(I))="" THEN650
630 PRINT #7, SS$(V(I))+" "+SA$(V(I))
640 PRINT SS$(V(I))+" "+SA$(V(I))
650 NEXT I
660 PRINT #7,"XIT"
670 DISK CLOSE,7
.
.  * END OF MAIN PROGRAM *
.
680 PRINT:PRINT
690 PRINTTAB(10)CR$" REFERENCES COMPLETE"
700 PRINTTAB(10)"SYMBOLS FOUND:  "SN+1
710 PRINTTAB(10)"REFERENCE FILE:  "RF$
730 PRINT:PRINT:END
.
. ** SUBROUTINES **
.
.  * BRANCH AND J (T=1) *
740 IF MID$(IN$,6,1)<>CR$ THEN FL=0: GOTO7
    80
.
.  * SIFT OUT BIT INSTRUCTIONS *
750 IF MID$(IN$,6,3)="BIT" THEN FL=0: GOTO
    780
.
.  * LABEL FOR TYPE *
760 M$=MID$(IN$,7,1)
770 FL=1
780 RETURN
.
. * MEMORY (T=2) *
790 M $=MID$(IN$,6,1)
800 A2$=MID$(IN$,13,1)
810 IF M $="J"ORA2$="Z" THEN FL=0: GOTO850
.
.  * SIFT OUT BRANCHES *
820 IF M $="B"ANDMID$(IN$,6,3)<>"BIT" THEN
    FL=0: GOTO850
.
.  * LABEL TYPE *

840 FL=1
850 RETURN
.
.  * Z PAGE REFERENCES (T=3) *
860 IF MID$(IN$,13,1)<>CR$ THEN FL=0: GOTO
    900
870 M$=" "
.
.  * LABEL FOR INDEXING *
880 IF LEN(IN$)>16 THEN M$=RIGHT$(IN$,1)
890 FL=1
900 RETURN
```

```
.PASS 4
.
.     RESOURCE 4 ** Z PAGE EQUATE FILE **
.
.          ONE BUFFER REQUIRED
.
.
.
.
.
100 REM *** RESOURCE 4 - ZPAGE EQUATES ***
110 REM T.R. BERGER 11/80
120 PRINT:PRINT
.
130 PRINT"RESOURCE ** STEP 4 - ZPAGE
    EQUATE FILE **"
.
140 PRINT:PRINT:INPUT"ZPAGE CROSS
    REFERENCE FILE";ZF$
150 INPUT"ZPAGE EQUATE FILE";ZE$
.
160 PRINT:INPUT"NUMBER OF SYMBOLS";NS
.
.  * LINE NUMBER AND INCREMENT *
170 FL=1000: IN=10
.
.  * DIMENSION ARRAYS *
180 DIM SS$(NS), V(NS)
.
.  * SYMBOL COUNTER *
190 SN=-1
.
.  ** LOAD SYMBOLS **
.
200 DISK OPEN,6,ZF$
210 PRINT:PRINT"LOADING SYMBOLS"
.
.  * LOOP BACK HERE *
220 INPUT #6,IN$
230 IF IN$="XIT" THEN380
.
.  * JUST THE ZPAGE REFERENCE *
240 IN$=LEFT$(IN$,2)
.
.  * PUT SYMBOLS IN ORDER *
.
.  * SEARCH FOR SYMBOL *
.  * BINARY SEARCH *
.
250 L=0:R=SN
.
.  * GO ADD NEW SYMBOL *
260 IF L>R THEN310
270 M=INT((L+R)/2)
.
.  * HAVE THIS ONE, GET ANOTHER *
280 IF IN$=SS$(V(M)) THEN220
290 IF IN$>SS$(V(M)) THEN L=M+1: GOTO260
300 R=M-1: GOTO260
.
.  * ADD SYMBOL TO LIST *
310 SN=SN+1: SS$(SN)=IN$
.
.  * POINT TO ITS PROPER POSITION IN
   ORDERING *
320 IF L=SN THEN360
330 FOR I=SN-1 TO L STEP -1
340 V(I+1)=V(I)
350 NEXT I
360 V(L)=SN
.
370 GOTO220
.  * LOOP BACK FROM HERE *
.
380 DISK CLOSE,6
```

```
.
.  * SYMBOLS ALL LOADED *
.
.  * PRINT EQUATES *
.
390 DISK OPEN,6,ZE$
.
.  * TITLE *
400 PRINT #6,STR$(FL)+" ;ZPAGE EQUATES"
.
.  * PRINT EQUATES NOW *
410 FOR I=0 TO SN
420 FL=FL+IN
430 IN$=STR$(FL)+"  HHZZ"+SS$(V(I))+" = $
    "+SS$(V(I))
440 PRINT #6,IN$
450 PRINT IN$
460 NEXT I
.
470 PRINT #6,"XIT"
.
.  * BUFFER 6 REQUIRES A PUT *
480 DISK PUT
490 DISK CLOSE,6
500 PRINT:PRINT
.
.  * OUTPUT DATA *
510 PRINTTAB(9) SN+1" SYMBOLS"
520 PRINTTAB(10) "ZPAGE CROSS REF FILE:
    "ZF$
530 PRINTTAB(10) "ZPAGE EQUATE FILE:
    "ZE$
540 PRINT:PRINT:END
```

```
.ONE PASS RESOURCE
.
.     RESOURCE S ** ONE PASS PROGRAM **
.
.        TWO BUFFERS REQUIRED
.
.
.
.
.
100 REM *** RESOURCE S ***
110 REM T.R. BERGER 2/81
120 PRINT
130 PRINTTAB(10)"RESOURCE - SINGLE PASS"
140 PRINT
.
.  ** REMOVE COMMA AND SEMICOLON **
150 POKE2972,13:POKE2976,13
.
160 INPUT"SOURCE FILE";SF$
170 INPUT"RESOURCE FILE";RF$
180 INPUT"EQUATE FILE";EF$
190 INPUT"CROSS REF FILE";CF$
200 INPUT"SCRATCH FILE";JF$
.
210 PRINT:INPUT"NUMBER OF SYMBOLS";NS
220 INPUT"NUMBER OF ZPAGE SYMBOLS";NZ
.
.  ** DIMENSION SYMBOL AND POINTER ARRAYS
   **
230 DIM SS$(NS),SB$(NS),SJ$(NS),SM$(NS),V(
    NS),SS(NS)
240 DIM ZS$(NZ),ZA$(NZ),U(NZ)
.
.  ** SYMBOL COUNTERS **
250 SN=-1:ZN=-1:SP$="      "
.
.  ** FIRST PASS **
.
260 DISK OPEN,6,SF$
270 DISK OPEN,7,JF$
.
.  ** LOOP BACK HERE **
280 INPUT #6,IN$
.
290 IF IN$="XIT" THEN790
300 IF LEN(IN$)<15 THEN280
.
.
.  ** ADJUST SOURCE, PICK UP SYMBOLS **
.
.  A1$=XXXX  ADDRESS
.  A2$=OPCODE +
.  A3$=OPERAND (SYMBOL)
.  A4$=ADDR MODE
.  OU$=A1$+A2$+A3$+A4$
.  IN$=INPUT FROM OS1 DISASSEMBLER
.
.
.
310 A3$="":A4$=""
.
.  ** GET ADDRESS **
320 A1$=LEFT$(IN$,4)
.
.  ** DO ERRORS **
330 IFMID$(IN$,13,1)="?"THENA2$=" .BYTE $
    "+MID$(IN$,6,2):GOTO760
.
.  ** DO REFORMATTING **
.
.  ** ELIMINATE END SPACES **
340 IN$=MID$(IN$,12): L=LEN(IN$)
350 IF MID$(IN$,L,1)=" " THEN L=L-1:GOTO
    350
360 IN$=LEFT$(IN$,L)
.
.  ** DO IMPLIED, ACCUMULATOR, IMMEDIATE
   ADDRESSING **
370 IF L<7ORMID$(IN$,6,1)="#" THEN A2$=IN
    $: GOTO760
.
.  ** ADJUST OPERAND POSITION **
380 IFMID$(IN$,6,1)="$"THENK=7:A2$=LEFT$(
    IN$,5)+" HH":GOTO400
390 K=8:A2$=LEFT$(IN$,6)+"HH"
.
.  ** ZPAGE CHECK **
400 M=K+2
.
.  ** DO ZPAGE OPERANDS **
410 IFM>LTHENA3$=RIGHT$(IN$,2):A2$=A2$+"Z
    Z":GOTO500
420 IF MID$(IN$,M,1)>"/"THEN440
430 A3$=MID$(IN$,K,2):A2$=A2$+"ZZ":A4$=
    MID$(IN$,M): GOTO500
.
.  ** TWO BYTE OPERAND CHECK **
440 M=K+4
```

```
.
.  ** DO TWO BYTE OPERANDS **
450 IFM>LTHENA3$=RIGHT$(IN$,4):GOTO470
460 A3$=MID$(IN$,K,4): A4$=MID$(IN$,M)
.
.  ** SEARCH FOR SYMBOL **
470 GOSUB1670
.
.  ** SYMBOL NOT FOUND, INSERT IT **
480 IF L>R THEN660
.
.  ** SYMBOL FOUND, ADD CROSS REF **
490 GOTO720
.
.  ** SEARCH FOR ZPAGE REF **
500 L=0: R=ZN
.
.  ** SYMBOL NOT FOUND, INSERT IT **
510 IF L>R THEN560
520 M=INT((L+R)/2)
.
.  ** SYMBOL FOUND, ADD CROSS REF **
530 IF A3$=ZS$(U(M)) THEN620
540 IF A3$>ZS$(U(M)) THEN L=M+1: GOTO510
550 R=M-1: GOTO510
.
.  ** ADD SYMBOL **
560 ZN=ZN+1: ZS$(ZN)=A3$
.
.  ** POINT TO PROPER POSITION IN ORDERING
   **
570 IF L=ZN THEN610
580 FOR I=ZN-1 TO L STEP-1
590 U(I+1)=U(I)
600 NEXT I
610 U(L)=ZN: M=L
.
.  ** GET ADDRESSING MODE **
620 A5$=" "
630 IF A4$<>"" THEN A5$=RIGHT$(IN$,1)
.
.  ** ADD CROSS REF TO STRING **
640 ZA$(U(M))=ZA$(U(M))+" "+A5$+A1$
650 GOTO760
.
.  ** ADD SYMBOL **
660 SN=SN+1:SS$(SN)=A3$
.
.  ** POINT TO PROPER POSITION IN ORDERING
   **
670 IF L=SN THEN710
680 FOR I=SN-1 TO L STEP-1
690 V(I+1)=V(I)
700 NEXT I
710 V(L)=SN: M=L
.
.  ** FIND CORRECT CROSS REF TABLE **
720 A5$=MID$(A2$,2,1): A0=1
730 IF A5$="B"ANDMID$(A2$,2,3)<>"BIT" THEN
    A0=2
740 IF A5$="J" THEN A0=3
.
.  ** ADD CROSS REF TO TABLE **
750 ON A0 GOSUB1640 ,1650 ,1660
.
.  ** GENERATE LINE FOR SCRATCH FILE **
760 OU$=A1$+A2$+A3$+A4$
770 PRINT #7,OU$: PRINT OU$
780 GOTO280
.  ** LOOP BACK HERE **
.
.  ** CLOSE SOURCE AND SCRATCH FILES **
790 PRINT #7, IN$
810 DISK CLOSE,6
820 DISK CLOSE,7
.
.  ** END FIRST PASS **
.
.  ** PASS 2, WRITE CROSS REF FILES **
.
830 DISK OPEN,7,CF$
840 PRINT #7,".  CROSS REFERENCES"
850 PRINT #7,"."
860 PRINT #7,".  ZPAGE"
870 PRINT #7,"."
.
.  ** DO ZPAGE REFS **
880 FOR I=0 TO ZN
890 A0$=ZA$(U(I)): ZA$(U(I))="": A2$=ZS$(
    U(I))
.
.  ** BREAK UP LONG LINES, PRINT FILE **
900 GOSUB1730
910 NEXT I
920 PRINT #7,".":PRINT #7,"."
930 PRINT #7,".  JMP & JSR"
940 PRINT #7,"."
.
.  ** DO JMP & JSR REFS **
950 FOR I=0 TO SN
960 A0$=SJ$(V(I)): SJ$(V(I))="": A2$=SS$(
    V(I))
.
.  ** BREAK UP LONG LINES, PRINT FILE **
970 GOSUB1730
980 NEXT I
990 PRINT #7,".": PRINT #7,"."
1000 PRINT #7,".  MEMORY": PRINT #7,"."
.
.  ** DO MEMORY REFS **
1010 FOR I=0 TO SN
1020 A0$=SM$(V(I)): SM$(V(I))="": A2$=SS$
     (V(I))
.
.  ** BREAK UP LONG LINES, PRINT FILE **
1030 GOSUB1730
1040 NEXT I
1050 PRINT #7,".": PRINT #7,"."
1060 PRINT #7,".  BRANCH": PRINT #7,"."
.
.  ** DO BRANCH REFS **
1070 FOR I=0 TO SN
1080 A0$=SB$(V(I)): SB$(V(I))="": A2$=SS$
     (V(I))
.
.  ** BREAK UP LONG LINES, PRINT FILE **
1090 GOSUB1730
1100 NEXT I
1110 PRINT #7,"XIT"
1120 DISK CLOSE,7
.
.  ** END REF FILES **
.
.  ** GENERATE RESOURCE FILE **
```

```
.
1130 DISK OPEN,6,JF$
1140 DISK OPEN,7,RF$
.
.  ** LINE NUMBER AND INCREMENT **
1150 CL=10000: IN=10
.
.  ** LOOP BACK HERE **
1160 INPUT #6,IN$
1170 IF IN$="XIT" THEN1260
.
.  ** GET ADDRESS OF LINE **
1180 A3$=LEFT$(IN$,4)
.
.  ** SEARCH FOR SYMBOL **
1190 GOSUB1670
.
.  ** SYMBOL FOUND, MARK IT, ENTER LABEL
   **
1200 IF L<=R THEN SS(M)=1: OU$="HH"+IN$:
     GOTO1220
.
.  ** SYMBOL NOT FOUND, DELETE ADDRESS **
1210 OU$=SP$+MID$(IN$,5)
.
.  ** ADD LINE NUMBER AND OUTPUT **
1220 OU$=STR$(CL)+" "+OU$
1230 CL=CL+IN
1240 PRINT #7,OU$: PRINT OU$
1250 GOTO1160
.  ** LOOP BACK FROM HERE **
.
.  ** CLOSE SCRATCH AND RESOURCE FILES **
1260 PRINT #7,IN$
1280 DISK CLOSE,6
1290 DISK CLOSE,7
.
.  ** RESOURCE FILE DONE **
.
.  ** DO EQUATE FILES **
.
1300 DISK OPEN,7,EF$
.
.  ** LINE NUMBER **
1310 CL=1000
1320 PRINT #7,STR$(CL)+"  ;EQUATE FILE"
1330 CL=CL+IN: PRINT #7,STR$(CL)+" ;"
1340 CL=CL+IN: PRINT #7,STR$(CL)+"  ;ZPAGE
     "
1350 CL=CL+IN: PRINT #7,STR$(CL)+" ;"
.
.  ** DO ZPAGE EQUATES **
1360 FOR I=0 TO ZN
1370 CL=CL+IN
1380 PRINT #7,STR$(CL)"  HHZZ"ZS$(U(I))"
     = $"ZS$(U(I))
1390 PRINT STR$(CL)"  HHZZ"ZS$(U(I))" = $"
     ZS$(U(I))
1400 NEXT I
.
1410 CL=CL+IN
1420 PRINT #7, STR$(CL)+" ;"
1430 CL=CL+IN: PRINT #7,STR$(CL)+" ;"
1440 CL=CL+IN: PRINT #7,STR$(CL)+"  ;TWO
     BYTE"
1450 CL=CL+IN: PRINT #7,STR$(CL)+" ;"
.
.  ** DO TWO BYTE EQUATES **
1460 FOR I=0 TO SN
1470 IF SS(I)=1 THEN1510
1480 CL=CL+IN
1490 PRINT #7,STR$(CL)"  HH"SS$(V(I))" =
     $"SS$(V(I))
1500 PRINT STR$(CL)"  HH"SS$(V(I))" = $"S
     S$(V(I))
1510 NEXT I
.
1520 PRINT #7,"XIT"
1530 PRINT #7,"E": PRINT #7,"E"
1540 DISK CLOSE,7
.
.  ** END OF EQUATES **
.
.  ** FINAL DATA **
.
1550 PRINT:PRINTTAB(10)"RESOURCE COMPLETE
     "
1560 PRINTTAB(7)SN+1" SYMBOLS"
1570 PRINTTAB(7)ZN+1" ZPAGE LOCATIONS"
1580 PRINTTAB(8)"SOURCE FILE:       ";SF$
1590 PRINTTAB(8)"SCRATCH FILE:      ";JF$
1600 PRINTTAB(8)"EQUATE FILE:       ";EF$
1610 PRINTTAB(8)"RESOURCE FILE:     ";RF$
1620 PRINTTAB(8)"CROSS REF FILE:    ";CF$
1630 PRINT:PRINT:END
.
.  ** END OF PROGRAM **
.  ** SUBROUTINES **
.
.  ** MEMORY CROSS REFS **
1640 SM$(V(M))=SM$(V(M))+" "+A5$+A1$:
     RETURN
.
.  ** BRANCH CROSS REFS **
1650 SB$(V(M))=SB$(V(M))+" "+MID$(A2$,3,1)
     +A1$: RETURN
.
.  ** JMP & JSR CROSS REFS **
1660 SJ$(V(M))=SJ$(V(M))+" "+MID$(A2$,3,1)
     +A1$: RETURN
.
. ** SEARCH FOR SYMBOL **
1670 L=0: R=SN
.
.  ** SYMBOL NOT FOUND **
1680 IF L>R THEN RETURN
1690 M=INT((L+R)/2)
.
.  ** SYMBOL FOUND **
1700 IF A3$=SS$(V(M)) THEN RETURN
1710 IF A3$>SS$(V(M)) THEN L=M+1: GOTO1680
1720 R=M-1: GOTO1680
.
.  ** BREAK UP LONG LINES, PRINT CROSS REF
   FILE **
1730 L=LEN(A0$)
1740 IF L=0 THEN RETURN
1750 IF L<49 THEN A1$=A0$: A0$="": GOTO
     1770
1760 A1$=LEFT$(A0$,48): A0$=MID$(A0$,49)
1770 PRINT #7,A2$" "A1$: PRINT A2$" "A1$
1780 GOTO1730
```

©

"Message Board" allows you to put up to 20 pages of messages on a Graphics 2 screen and display them repeatedly for as long as you like. It makes a fine message board for any kind of meeting and can even be a title maker for home video recorders.

Atari Message Board

Dennis J. Harkins
Hatfield, PA

We use an expanded version of this program to generate messages for a cable TV channel here, and we leave our Atari 800 running for hours, flipping through page after page of program schedules and community announcements. We also make titles and credits for our student video productions with the program, since the Atari has a nice "clean" video output through the DIN socket on the side.

String With Dynamic Keyboard

The program stores the information in DATA statements through Atari's "dynamic" keyboard technique (see Bruce Frumker's article in **COMPUTE!**, August 1981, #15, for a good example of this). This keeps the program short and simple and makes it easy to save messages on either disk or cassette.

The program first asks whether you'd like to write messages or run your previously stored messages. If you choose to write messages, it then asks for a color combination for your message and background. You then enter your message line by line. Hitting RETURN leaves a blank line. After the 10th (or 20th) line is entered, you can see it displayed and have the option to make any changes. After you are satisfied with the "page," you are asked if you want to write another page. When you are through writing messages, you can then either RUN them or save the program to disk or tape to RUN later.

Choosing colors and luminance values can be time consuming at first, so try the following values when asked for color information, if you don't have any favorite color combinations:

1, 14, 7, 8	(Gold on Blue)
7, 8, 1, 14	(Blue on Gold)
2, 14, 4, 6	(Gold on Red)
14, 12, 11, 6	(Gold on Green)
7, 2, 3, 6	(Blue on Red)
7, 12, 0, 6	(Blue on Gray)

You will learn a lot about the Atari's colors by playing a bit with the combinations. We have a little color selector program that lets you preview the colors, but I'll save that for another time. Remember to avoid using commas in your message, since commas in a DATA statement tell the computer to end that particular "read." You may notice that the "window" is still at the bottom of each page: it's just the same color and luminance as the background. We use this for our logo and the date.

The loop in line 2620 controls the amount of time each page is displayed, and you will probably want to change this to suit your messages.

I'm sure you can think of some refinements (I've already added dozens of them). How about messages in four colors? It's a simple matter to add the extra SETCOLOR information and then type, in lowercase and inverse characters, some of the lines of your message. Experiment a bit. You can come up with some pretty fancy looking messages with a little work.

Program Details

Line 2010 dimensions an Answer string and a Message string (note its length) and sets the Line Number (LN) variable to zero.

Line 2130 sets the value of A to correspond (inversely!) to the graphics mode wanted. This allows it to be used in line 2310 to input 10 or 20 lines when you are writing a message.

Line 2210 inputs B, C, D, E as your color values. To see what's happening here, look at line 2550 where the values are read into SETCOLOR

```
2010 DIM A$(1),M$(20):LN=0
2020 ? CHR$(125):POKE 752,1:POSITION 11,
     8:? "MESSAGE GENERATOR"
2030 ? :? :? "{9 SPACES}BY DENNIS HARKIN
     S"
2040 FOR DELAY=1 TO 500:NEXT DELAY
2050 ? CHR$(125):POKE 752,0:POSITION 2,8
2060 ? "TO WRITE MESSAGES, TYPE W"
2070 ? :? "TO RUN YOUR MESSAGES,TYPE R"
2080 ? :? :? "THEN HIT RETURN"
2090 ? :? :INPUT A$:IF A$="R" THEN 2500
2100 IF A$<>"W" THEN 2050
2110 ? CHR$(125):POSITION 2,8
2120 ? "LARGE OR SMALL LETTERS?"
2130 ? :? :? "TYPE 1 FOR LARGE, 2 FOR SM
     ALL":? :? :INPUT A
2140 IF A<1 OR A>2 THEN GOTO 2110
2150 ? CHR$(125):POSITION 2,4
2160 ? "NOW ENTER YOUR COLOR CHOICES"
2170 ? :? :? "REMEMBER TO ENTER FOUR NUM
     BERS{8 SPACES}SEPARATED BY COMMAS"
2180 ? :? :? "EXAMPLES:"
2190 ? :? "{10 SPACES}1,14,7,8"
2200 ? :? "{5 SPACES}OR{3 SPACES}7,8,1,1
     4"
2210 ? :? :? "THEN HIT RETURN":? :? :INP
     UT B,C,D,E
2220 LN=LN+1
2230 ? CHR$(125)
```

COMPUTE!'s Second Book Of Atari

After only three years on the market, the Atari 400/800 microcomputers have become among the most popular personal computers ever made. So it was no surprise when *COMPUTE!'s First Book of Atari*, a collection of the best Atari articles published during 1980-81 in **COMPUTE!** Magazine, also became a "bestseller" with Atari enthusiasts. The first printing sold out in just a few months.

That's why we've followed up with *COMPUTE!'s Second Book of Atari*. Available immediately, the *Second Book of Atari* continues **COMPUTE!**'s tradition for personal computer users.

But the *Second Book of Atari* differs from the *First Book* in one important respect – all the articles are totally new and previously unpublished. The *Second Book of Atari* includes such interesting articles as "Page Flipping," "Fun With Scrolling," "Perfect Pitch," "Player-Missile Drawing Editor," and "TextPlot Makes a Game." Whole chapters are devoted to subjects such as "Advanced Graphics and Game Utilities," "Programming Techniques," and "Beyond BASIC." With 250 pages – more than 25 percent thicker than the *First Book* at the same price – the *Second Book of Atari* is crammed with information and ready-to-type program listings. And the book is spiral-bound to lie flat and is fully indexed for quick reference.

Best of all, *COMPUTE!'s Second Book of Atari*, like **COMPUTE!** Magazine itself, is written and edited to appeal to all computer enthusiasts – beginners and experts alike. Priced at only $12.95.

Available at computer dealers and bookstores nationwide. To order directly call TOLL FREE 800-334-0868. In North Carolina call 919-275-9809. Or send check or money order to **COMPUTE! Books**, P.O. Box 5406, Greensboro, NC 27403.

Add $2.00 shipping and handling. Outside the U.S., add $4.00 for air mail, $2.00 for surface mail. All orders prepaid, U.S. funds only.

```
2240 ? "{DOWN}";LN;" DATA ";A;",";B;",";
     C;",";D;",";E
2250 ? :? :? "CONT"
2260 POSITION 0,0
2270 POKE 842,13:STOP
2280 POKE 842,12
2290 IF A=1 THEN LN=LN+9
2300 IF A=2 THEN LN=LN+4
2310 FOR G=1 TO 10*A
2320 ? CHR$(125):POSITION 6,4:? "TYPE LI
     NE ";G;" OF MESSAGE HERE"
2330 POSITION 10,8:? "------------------
     --"
2340 POSITION 2,10:? "ONLY 20 CHARACTERS
      WILL BE ACCEPTED"
2345 POSITION 2,13:? "REMEMBER THAT YOU
     CANNOT USE COMMAS{3 SPACES}IN YOUR
     MESSAGE"
2350 POSITION 9,7:POKE 752,0:INPUT M$:SE
     TCOLOR 1,7,4:POKE 752,1:? "
     {CLEAR}"
2360 ? "{DOWN}";LN;" DATA ";M$
2370 ? :? :? "CONT"
2380 POSITION 0,0
2390 POKE 842,13:STOP
2400 POKE 842,12:? "{CLEAR}":SETCOLOR 1,
     7,10
2410 ON A GOTO 2420,2425
2420 IF G<>10*A THEN LN=LN+10:GOTO 2430
2425 IF G<>10*A THEN LN=LN+5
2430 NEXT G
2440 ? CHR$(125):POSITION 13,8:? "MESSAG
     E ENTERED"
2450 POSITION 1,14:? "DO YOU WANT TO SEE
      THAT PAGE (Y OR N)";:INPUT A$
2460 IF A$="Y" THEN 2700
2480 ? :? :? :? "DO YOU HAVE ANOTHER PAG
     E (Y OR N)";:INPUT A$
2485 IF A$="Y" THEN GOTO 2110
2490 IF A$<>"N" THEN 2480
2495 GOTO 2050
2500 ? CHR$(125):? :? :? "PAGES TO BE RU
     N";:INPUT Z
2510 RESTORE :FOR W=1 TO Z
2520 READ A,B,C,D,E
2530 IF A=1 THEN GRAPHICS 2
2540 IF A=2 THEN GRAPHICS 1
2550 SETCOLOR 0,B,C:SETCOLOR 2,D,E:SETCO
     LOR 4,D,E
2560 FOR N=0 TO 10*A-1
2570 READ M$
2580 LET P=(20-LEN(M$))/2
2590 POSITION P,N
2600 PRINT #6;M$
2610 NEXT N
2620 FOR T=1 TO 3000:NEXT T
2630 POKE 77,0:NEXT W
2640 GOTO 2510
2700 RESTORE LN-99
2710 READ A,B,C,D,E
2720 IF A=1 THEN GRAPHICS 2
2730 IF A=2 THEN GRAPHICS 1
2740 SETCOLOR 0,B,C:SETCOLOR 2,D,E:SETCO
     LOR 4,D,E
2750 FOR N=0 TO 10*A-1
2760 READ M$
2770 LET P=(20-LEN(M$))/2
2780 POSITION P,N
2790 PRINT #6;M$
2800 NEXT N
2810 ? "THIS IS PAGE ";LN/100
2820 ? "HIT RETURN TO CONTINUE":INPUT A$
2830 GRAPHICS 0:POSITION 2,10:? "DO YOU
     WANT TO RETYPE THAT PAGE{7 SPACES}(
     Y OR N)";:INPUT A$
2840 IF A$="Y" THEN LN=LN-100:GOTO 2110
2850 IF A$<>"N" THEN 2830
2860 GOTO 2480
```

statements.

Lines 2230-2280 use the "dynamic keyboard" routine to enter the first data line.

Lines 2310-2340 set up the loop to enter each line of the message, again using the "dynamic keyboard" routine. Notice the way the LN variable increments depending on the number of lines in your message. This makes a lot of little options easier. LIST 201-300, for example will always LIST page 3, no matter what length each page is.

Lines 2500-2640 are the actual display loop. Note that line 2580 centers the data string in a line of 20 spaces.

Lines 2700-2820 are the same display loop modified for the "PREVIEW" option.

To use the program as a titler for your home video productions, change line 2620 to:

2620 GET#1,KEY

and line 2500 to:

2500 PRINT CHR$(125):? :? :? "PAGES TO BE RUN" ;:INPUT Z:OPEN #1,4,0,"K:"

You'll be able to prepare your titles in the same way as you create messages and hit any key to change to the next credits. ©

Mass-delete lines, renumber, and search – many programming aids become available when you put the Assembler Cartridge to work on a BASIC program.

Editing BASIC Programs With The Atari Assembler/Editor Cartridge

Dennis Allen
San Jose, CA

It's probably a safe bet that the majority of Atari owners do most if not all of their programming in BASIC. However, this article will show you how to put the Atari Assembler/Editor Cartridge to work providing some BASIC programming aids.

The Editor/Assembler consists of three programs: the Assembler, the Debugger, and the Editor. The Assembler converts source programs into object code which is executable by the computer. The Debugger is a programming aid which does just that – debugs machine language programs. The Editor is a fairly simple line-oriented text editor through which you can enter and correct or modify source programs – *or any other ASCII text files.* It is the Editor which will allow you to modify your BASIC programs.

Apart from the I/O commands which allow you to SAVE or LOAD source and object code files to and from the cassette or disk, there are four major commands of interest to the BASIC programmer: Renumber, Delete, Replace, and Find. *Renumber* is a simple renumbering utility. It allows you to specify the starting line number and the increment. *Delete* lets you erase any line or group of lines. *Find* will show you all of the statement lines where a certain string of characters appears. *Replace* will allow you to substitute any string of characters for any other string – a line at a time or throughout the entire program at once.

A Major Typing Task Avoided

Program 1 is a simple demonstration of the use of these editing features. Suppose you've just CLOADed this program into your computer. The first thing you will notice is the line number increment. You would like to insert some graphics routines in the middle of the sound routines, but you can't because a thoughtless programmer used a line number increment of one. Now pretend this is a very large program and the routines you want to insert will extend the overall program beyond your memory limit. You now have two problems which would require many hours of retyping to solve. Not with the Editor!

Step one is to get the program in a form the Editor can LOAD. To do this you type LIST"C" [*For disk, use LIST"D:Name"*]. At this point, it is a good idea to make a hardcopy listing of the program you are editing, especially if it's a long one. Once the program has been listed to cassette, you simply replace the Atari BASIC Language cartridge with the Editor/Assembler, power up and type ENTER# C: (Note the #,NOT". [*For disk:* ENTER#D:Name]). Once the cassette stops, type *LIST*, and the program is LISTed to the screen, just as if you were still in BASIC.

Step two is to get rid of all the REM statements, since we are concerned with saving memory. Referring to Program 1, the first 11 REMs are easy to find. To get rid of these, we type the command: *DEL1,11*. Now do a LIST and you will find that the first 11 lines have vanished. This is much easier than typing 11 separate line numbers followed by carriage returns. Assuming this were a much longer program (with many REM statements embedded throughout), we would want a convenient way to list all the lines with the word REM. Simple. Type: *FIND/REM/,A*. The computer responds:

```
16 REM SOUND VALUES
23 REM SOUND SUBROUTINE
```

Now type: *DEL16*, followed by *DEL23*, or use the BASIC method – simply type the line number followed by a RETURN. The list will show the deletion of these two lines as well. The ",A" following the FIND command is a qualifier meaning "all occurrences of." Thus "FIND/REM/,A" means: find *all* lines which contain the letters REM and list them to the screen. It does not matter where in the statement the string is to be found. If the program you are editing has a line such as:

```
1010 PRINT "THIS IS A REMINDER":
     GOTO 3050
```

the FIND command will find the REM in "REMINDER" and list that line as well. Examine the results of the FIND listing before you delete any lines.

Using REPlace

Like the FIND command, REP has an "all occurrences" option. REP also has a "query" option which allows you to step through each line containing the string to be replaced. The computer will pause, then "ask" you whether or not you want the REP command to be executed on that line. This option is very convenient when you have to change, say, a variable name which may also appear in a PRINT statement.

To scan through the program for all of the occurrences of TIMER, and to optionally replace them with the shorter variable T, type: *REP/TIMER/T/1,1000,Q*. (The "1,1000" determines the range of line numbers for which the command is valid; choosing a very large number will insure scanning of the entire program.) The computer responds:

```
13 FOR TIMER=1 TO 100:NEXT TIMER
?■
```

To exercise the REP command for this line, type *Y* followed by a return. The computer responds:

```
13 FOR T=1 TO 100:NEXT TIMER
?■
```

The first occurrence of TIMER in line 13 has been changed to T. The computer is now querying you about the second occurrence. To change this one and all of the following occurrences, just type Y after every prompt. When the Editor has scanned through the entire program, it will display the EDIT prompt, meaning the function is complete. Use the same procedure to change WAIT to W and LOOP to L. Since the variables T and D are already used, we will change TONE to A and DURATION to X. When through, do another list to the screen just to make sure you didn't make any mistakes.

One of the reasons for using the query option is that the Editor signals the completion of a command with the EDIT prompt whether or not the function was actually performed. In the above example, had you tried to do a *REP/TIMER/T/,A*, and typed *TIMET* by mistake, the Editor would respond with its usual prompt, although, since there is no TIMET string in the program, nothing was REPlaced. If there *were* another variable named TIMET – you get the picture.

Having shortened all of our variable names, we are ready for step four: renumbering the lines to make room for our patches. This is where the hardcopy of the original program comes in handy. If you have one, fine. If not, you will have to struggle along with screen lists (like I do). The RENumber command will effortlessly renumber all statement numbers, but unfortunately will do nothing with all your GOTOs and GOSUBs. These will have to be handled separately.

First, identify where all of your GOTOs and GOSUBs are GOing to. Make a note of the target statements; i.e., GOSUB 17 points to a statement which starts "A=81", which follows a GOTO statement. Your "target notes" can be anything which uniquely places the location in the program of the targets; there can be no confusion. My notes looked like this:

17 A=81 (after GOTO)
24 T=2 (after RET)
32 FOR T=1 (after RET)
34 SO.0 (after L=1)

Having fixed the relative position of our target lines, we can execute the RENumber option. Type: *REN100,10*. The "100" tells the Editor what the new number of the first line of our program will be; the "10" specifies the increment. Now we can fix our GOTOs and GOSUBs. To do this, you can either do a screen list or use the FIND command. Referring to the target notes, do a *FIND/A=81/,A*, to determine old line number 17's new number.

Now simply REPlace all GOxx references to line 17 with GOxx 140. (You don't need to worry about the leading zeros in the Editor's listing. BASIC will ignore them.) Execute the following, responding with Y to all of the queries:

REP/GOSUB 17/GOSUB 140/1,1000,Q
(Don't forget the space after the GOxx!)
REP/GOSUB 24/GOSUB 200/1,1000,Q
REP/GOTO 32/GOTO 280/1,1000,Q
REP/THEN 34/THEN 300/1,1000,Q

Now do one last screen list, checking it carefully for errors. Your new program should look like Program 2, ready for your additions or patches. If there are no errors, type: LIST#C:, to produce a cassette file, which you can *ENTER"C:* using the BASIC cartridge. Once the modified program is ENTERED into BASIC, it can be RUN, CSAVEd and treated exactly as if it were written using the BASIC cartridge. If you have followed this procedure using Program 1 and have RUN both versions, note the great difference that shortening the variable names causes in the sound routines.

Program 1.

```
1 REM DEMONSTRATION PROGRAM
2 REM
3 REM by
4 REM
5 REM Dennis Allen
6 REM
7 REM
8 REM
9 REM
10 REM
11 REM 1981
12 GOSUB 17
13 FOR TIMER=1 TO 100:NEXT TIMER
```

```
14 GOSUB 17
15 GOTO 32
16 REM SOUND VALUES
17 TONE=81:D=20:GOSUB 24
18 TONE=64:D=20:GOSUB 24
19 TONE=50:D=20:GOSUB 24
20 TONE=43:D=95:GOSUB 24
21 TONE=50:D=20:GOSUB 24
22 RETURN
23 REM SOUND SUBROUTINE
24 TIMER=2
25 SOUND 0,TONE,10,15
26 SOUND 1,TONE,12,4
27 FOR DURATION=1 TO D:NEXT DURATION
28 SOUND 0,0,0,0
29 SOUND 1,0,0,0
30 FOR WAIT=1 TO 10:NEXT WAIT
31 RETURN
32 FOR TIMER=1 TO 50:NEXT TIMER
33 LOOP=1
34 SOUND 0,162,10,8
35 SOUND 1,162,2,4
36 FOR TIMER=1 TO 15:NEXT TIMER
37 SOUND 0,128,10,8
38 SOUND 1,128,2,8
39 FOR TIMER=1 TO 15:NEXT TIMER
40 LOOP=LOOP+1:IF LOOP<10 THEN 34
41 END
```

Program 2.

```
100 GOSUB 140
110 FOR T=1 TO 100:NEXT T
120 GOSUB 140
130 GOTO 280
140 A=81:D=20:GOSUB 200
150 A=64:D=20:GOSUB 200
160 A=50:D=20:GOSUB 200
170 A=43:D=95:GOSUB 200
180 A=50:D=20:GOSUB 200
190 RETURN
200 T=2
210 SOUND 0,A,10,15
220 SOUND 1,A,12,4
230 FOR X=1 TO D:NEXT X
240 SOUND 0,0,0,0
250 SOUND 1,0,0,0
260 FOR W=1 TO 10:NEXT W
270 RETURN
280 FOR T=1 TO 50:NEXT T
290 L=1
300 SOUND 0,162,10,8
310 SOUND 1,162,2,4
320 FOR T=1 TO 15:NEXT T
330 SOUND 0,128,10,8
340 SOUND 1,128,2,8
350 FOR T=1 TO 15:NEXT T
360 L=L+1:IF L<10 THEN 300
370 END
```

©

This program can be used to ring school bells, to scare crows, and to do many other timing tasks.

Perform A Task At Equally Spaced Intervals

Marvin L. De Jong
Department of Mathematics-Physics
The School of the Ozarks
Pt. Lookout, MO

A short time ago we required an observation of a certain experiment in our laboratory to be made approximately every 30 minutes over a period of 30 hours. Not wishing to stay up night and day for that period of time, we designed a timing program to do the observation with the aid of a movie camera. The movie camera had a remote switch, so all we had to do was close the switch for a period of three seconds every 30 minutes, over a time interval of 30 hours.

Perhaps you will encounter a similar timing problem sometime. This program should handle most such problems. It is designed to perform some task at equally spaced programmable intervals ranging from 0.01s to 99 days, 23 hours, 59 minutes, and 59.99 seconds. The task is performed for the first time at 0.01s after the program is initiated. It is then performed at equally spaced intervals until execution is terminated. The desired time interval between tasks is determined by the contents of locations $0000 (tenths and hundredths of seconds), $0001 (tens and units of seconds), $0002 (minutes), $0003 (hours), and $0004 (days). The appropriate *decimal* quantities must be entered into these locations.

Whatever task the computer is to perform is accomplished by subroutine TASK, which we located at $OF16. We include our subroutine in the program to illustrate the use of the T2 timer on the 6522 VIA. The task must take less time than the interval between tasks, by about 200 microseconds.

Any microcomputer system with a 6522 Versatile Interface Adapter can be used to execute the program. The program itself is easily relocated. The comments should explain most of the details regarding its operation, which is very similar to a clock routine. It is important to note that the T1 timer is operating in its free-running mode with interrupts. In order for the program to work, the IRQ interrupt vector *must* be loaded to point to the interrupt routine at $0F4D in Program 1.

Although our task consisted of simply switching a device on for a few seconds, the task might be more complex, such as performing an analog-to-digital conversion, measuring the activity of a radioactive source, making a temperature or voltage measurement with T/F or V/F converter, starting the coffee pot in the morning, firing a gun to scare the crows from the corn, or running the school buzzers or bells to let the kids out of class. Let your imagination go wild.

Program 1: A program to perform a task at equally-spaced programmable intervals.

$0000-$0004 = TIMTBL; five locations whose contents determine the time interval in hundredths of seconds, seconds, minutes, hours, and days.

$0005-$0009 = CNTTBL; five locations used to count until [CNTTBL] = [TIMTBL].

$000A = FLAG; a location that is decremented to zero when [CNTTBL] = [TIMTBL].

IRQ VECTOR = $0F4D

```
$ØE98 A9 Ø1      START   LDA #$Ø1       Initialize Port B for the task.
 ØE9A 8D ØØ AØ           STA PBD        Port B, pin PBØ starts at logic one.
 ØE9D 8D Ø2 AØ           STA PBDD       Data Direction Register set to
 ØEAØ A9 4Ø              LDA #$4Ø       make PBØ an output pin.
 ØEA2 8D ØB AØ           STA ACR        Put T1 in free-running mode.
 ØEA5 A9 CØ              LDA #$CØ       Enable interrupts by setting
 ØEA7 8D ØE AØ           STA IER        IER, bit six.
 ØEAA A9 ØE              LDA #$ØE       Set up T1 timer to run free with
 ØEAC 8D Ø4 AØ           STA T1LL       a period of ($27ØE + 2) TC = Ø.Ø1s.
 ØEAF A9 27              LDA #$27
 ØEB1 8D Ø5 AØ           STA T1LH       Start timing.
 ØEB4 58                 CLI            Clear interrupt flag to allow interrupts.
 ØEB5 A9 Ø1              LDA #$Ø1       Set FLAG to be non-zero.
 ØEB7 85 ØA              STA FLAG
 ØEB9 F8                 SED            All arithmetic functions done in
 ØEBA 38                 SEC            decimal mode.
 ØEBB A2 FB              LDX #$FB       Start by setting CNTTBL = TIMTBL.
 ØEBD B5 Ø5      BACK    LDA TIMTBL,X
 ØEBF 95 ØA              STA CNTTBL,X
 ØEC1 E8                 INX
 ØEC2 DØ F9              BNE BACK
 ØEC4 A5 Ø5              LDA CHSEC      Now make CNTTBL = TIMTBL - Ø.Ø1.
 ØEC6 FØ Ø6              BEQ ONE        Then task will be performed for
 ØEC8 E9 Ø1              SBC #$Ø1       the first time after the first
 ØECA 85 Ø5              STA CHSEC      time out of the T1 timer
 ØECC BØ 34              BCS ONWRD
 ØECE A9 99      ONE     LDA #99
 ØEDØ 85 Ø5              STA CHSEC
 ØED2 A5 Ø6              LDA CSEC       Get seconds from CNTTBL.
 ØED4 FØ Ø6              BEQ TWO        If it is not zero, subtract one.
 ØED6 E9 Ø1              SBC #$Ø1
 ØED8 85 Ø6              STA CSEC       Result into seconds place.
 ØEDA BØ 26              BCS ONWARD     No borrow, so get out.
 ØEDC A9 59      TWO     LDA #59        If it is zero, borrow one from
 ØEDE 85 Ø6              STA CSEC       the seconds plce.
 ØEEØ A5 Ø7              LDA CMIN       Get minutes.
 ØEE2 FØ Ø6              BEQ THREE      Is it zero?
 ØEE4 E9 Ø1              SBC #$Ø1       No. Subtract one and get out.
 ØEE6 85 Ø7              STA CMIN
 ØEE8 BØ 18              BCS ONWRD
 ØEEA A9 59      THREE   LDA #59        Yes. There was a borrow from
 ØEEC 85 Ø7              STA CMIN       the minutes place.
 ØEEE A5 Ø8              LDA CHRS       Get hours.
 ØEFØ FØ Ø6              BEQ FOUR       Is it zero, then borrow from days.
 ØEF2 E9 Ø1              SBC #$Ø1       Otherwise, subtract one.
 ØEF4 85 Ø8              STA CHRS
 ØEF6 BØ ØA              BCS ONWARD
 ØEF8 A9 23      FOUR    LDA #23
 ØEFA 85 Ø8              STA CHRS
 ØEFC A5 Ø9              LDA CDAYS
 ØEFE E9 Ø1              SBC #$Ø1
 ØFØØ 85 Ø9              STA CDAYS      Finished doing [CNTTBL]=[TIMTBL]-0.01s.
 ØFØ2 A5 ØA      ONWRD   LDA FLAG       Wait in this loop until FLAG is zero.
 ØFØ4 DØ FC              BNE ONWRD
 ØFØ6 A2 FB              LDX #$FB       Clear CNTTBL, then do TASK.
 ØFØ8 A9 ØØ              LDA #$ØØ
 ØFØA 95 ØA      UPWRD   STA CNTTBL,X
 ØFØC E8                 INX
 ØFØD DØ FB              BNE UPWRD
 ØFØF 2Ø 16 ØF           JSR TASK       Perform TASK.
 ØF12 C6 ØA              DEC FLAG       Set FLAG to non-zero number.
 ØF14 DØ EC              BNE ONWRD      Wait for the next interval to elapse.
 ØF16 A9 3C      TASK    LDA #$3C       Count $3C = 6Ø intervals of .Ø5s
 ØF18 85 3Ø              STA COUNT      each.
```

```
0F1A CE 00 A0            DEC PBD          PB0 to logic zero to start device.
0F1D A9 4F     HERE      LDA #$4F         Load T2 for 0.05s.
0F1F 8D 08 A0            STA T2LL
0F22 A9 C3               LDA #$C3         $C34F + 1 = 50000
0F24 8D 09 A0            STA T2CH
0F27 AD 0D A0  WAIT      LDA IFR          Read interrupt flag register
0F2A 29 20               AND #$20         to see if T2 has timed out.
0F2C F0 F9               BEQ WAIT
0F2E C6 30               DEC COUNT
0F30 D0 EB               BNE HERE
0F32 EE 00 A0            INC PBD          PB0 to logic one to turn device
0F35 60                  RTS              off.
  .                       .
  .                       .
  .                       .
0F4D 48        INTERRUPT PHA              Save accumulator.
0F4E 8A                  TXA              Save X.
0F4F 48                  PHA
0F50 AD 04 A0            LDA T1CL         Clear T1 interrupt flag, IFR6.
0F53 A9 05               LDA #$05         Set FLAG to $05.
0F55 85 0A               STA FLAG
0F57 F8                  SED              Set decimal mode for subsequent
0F58 38                  SEC              BCD arithmetic. Set carry to
0F59 A2 FE               LDX #$FE         add one to the least significant
0F5B B5 07     ADD1      LDA CNTTBL,X     location of the CNTTBL.
0F5D 69 00               ADC #$00         Add carry.
0F5F 95 07               STA CNTTBL,X
0F61 E8                  INX              The two lowest locations of CNTTBL
0F62 D0 F7               BNE ADD1         have now been incremented by 0.01s.
0F64 C9 60               CMP #$60         Have seconds reached 60 yet?
0F66 D0 26               BNE OUT          No. Branch to see if CNTTBL=TIMTBL.
0F68 A9 00               LDA #$00         Yes. Clear seconds counter and
0F6A 85 06               STA SEC          increment minutes counter.
0F6C A5 07               LDA MIN          Carry is set from CMP instruction,
0F6E 69 00               ADC #$00         so minutes are incremented with ADC.
0F70 85 07               STA MIN
0F72 C9 60               CMP #$60         Have minutes reached 60 yet?
0F74 D0 18               BNE OUT          No. Branch to see if CNTTBL=TIMTBL.
0F76 A9 00               LDA #$00         Yes. Clear minutes counter and
0F78 85 07               STA MIN          increment hours counter.
0F7A A5 08               LDA HRS
0F7C 69 00               ADC #$00
0F7E 85 08               STA HRS
0F80 C9 24               CMP #$24         Have hours reached 24 yet?
0F82 D0 0A               BNE OUT          No. Branch to see if CNTTBL=TIMTBL.
0F84 A9 00               LDA #$00         Yes. Clear hours counter.
0F86 85 08               STA HRS
0F88 A5 09               LDA DAYS         Increment days counter using
0F8A 69 00               ADC #$00         set carry flag.
0F8C 85 09               STA DAYS
0F8E A2 FB     OUT       LDX #$FB         Subtract the five bytes of CNTTBL
0F90 38                  SEC              from the five bytes of TIMTBL.
0F91 B5 05     HERE      LDA TIMTBL,X     Get a byte from TIMTBL.
0F93 F5 0A               SBC CNTTBL,X     Subtract a byte from CNTTBL
0F95 D0 02               BNE THERE        If result is not zero, leave FLAG alone.
0F97 C6 0A               DEC FLAG         Otherwise decrement FLAG.
0F99 E8        THERE     INX              Repeat for the remaining bytes of
0F9A D0 F5               BNE HERE         TIMTBL and CNTTBL. If all five bytes
0F9C D8                  CLD              are equal, FLAG contains zero to
0F9D 68                  PLA              flag main program to do task.
0F9E AA                  TAX              Restore registers.
0F9F 68                  PLA
0FA0 40                  RTI              Return.
```

©

These screens enter the FORTH machine and bring back two lists: what words are and are not being used in a particular program. These lists can be used to streamline an application by removing (metacompiling) all unreferenced words. Or you could analyze a program to discover areas where large numbers of words were being called and then speed things up by replacing those sections with machine language.

The FORTH Page

REFS

Richard Mansfield and Lou Cargile

The question seems simple at first. What words are *not* used in a FORTH program? Say you've written a simple game and you decide to find out what words in the dictionary are not necessary to run this game. Since these words are never referenced in this game, they could be left out of this FORTH application entirely. (See "Headless Metacompilation," **COMPUTE!**, July 1982, p. 174.) Unfortunately, you can't just look at the screen which holds the game and then make a list of all the words on it.

The words on the game screen are the highest level definitions. Within each of these definitions is a group of other words, and within those words are yet others below. Imagine that you defined : SCOREFORMAT ." SCORE IS –" ; At first glance, it might seem that you could eliminate most FORTH words since this definition uses only the word ." and that's all. But what words are necessary to support ." itself?

FORTH has hidden levels, words within words, radiating below what you can see. The simple ." uses many other words. It uses EXPECT, for one. What's more, EXPECT itself uses several words. Within EXPECT is +ORIGIN. And +ORIGIN uses + which is finally an end point. We can come to a full stop with +, a code definition, complete within itself, referencing nothing. Higher level words, however, all depend upon a hidden support network. What's visible reveals only a fraction of the words actually being used. Beneath, like a desert plant, there is a root system many times the size of what appears on the surface.

List Management

The function of the REFS screens is to analyze a particular FORTH program and report back to you with a list of words representing that hidden root system. Any word which is referenced within an application is added to the list. When you load the REFS screens, a buffer area is created (REF.LIST) with room for 512 CFAs. This buffer is cleared out whenever you execute XREF. To accumulate a list of references you execute +REFS. Accumulation continues during whatever FORTH operations take place up to and including the execution of -REFS.

Accumulation occurs in the order in which each word was first referenced. The list will not duplicate references: a word goes onto the list only the first time it is called.

After the accumulation is stopped with -REFS, you can see the list of words by executing the word REFS. Perhaps equally useful is the list of words which were not needed and never made the REFS list. NONREFS will list out the words which are unnecessary for your application, beginning with the top of the context vocabulary. These words would be candidates for omission in a compacted, headless, dedicated FORTH program.

They would only be candidates, though. The words VARIABLE, CONSTANT, :, etc., will *always* appear in the NONREFS list, but they must be included so that the address of metacompiler utilities DOCOLON (or DOVARIABLE or DOCONSTANT, etc.) will be that of the compacted FORTH, not the host FORTH. And the words intended to be executed in the application will in general be on this list. This is because the execution of a word from the input stream does not of itself put the word on the REFS list. Only when a word is called by another executing word does it become a part of the REFS list. For example, compiling

```
: TEST2 TEST1 ;
```

would not put either TEST2 or TEST1 on the REFS list. Executing TEST2, however, would put TEST1 on the REFS list.

Other Uses

The listing of NONREFS is somewhat slowed because each word printed has to first be compared with the entire REFS list. If the list were maintained in sorted order, this listing could be made faster. In any case, it is worthwhile having the REFS listed out in the order in which the words were first interpreted.

The approach used in these screens intercepts the action of the inner interpreter. It is possible to create a debugging "trace" routine in a similar way and watch FORTH at work during execution.

Program 1. REFS Screens

```
SCR # 244
 ( REFS --1 )  FORTH DEFINITIONS HEX
Ø VARIABLE REF.LIST 3FE ALLOT ( SPACE FOR 512 WORD CFA'S)
 ' REF.LIST 400 FF FILL ( INITIALIZE REF.LIST ARRAY )
CODE STORE.REF  ( CODE TO BE PATCHED TO INNER INTERPRETER )
  T REF.LIST 2 - 100 /MOD
  # LDA, N 1+ STA, # LDA, N STA,
  Ø # LDY,
  BEGIN, INY, INY, Ø=  IF, N 1+ INC, THEN,
    N )Y  LDA, W CMP,
      Ø= IF, INY, N )Y LDA, W 1+ CMP,
        Ø= IF, FF # LDA, THEN, DEY,
      THEN,  FF # CMP,
  Ø= END, N )Y LDA, FF # CMP,
  Ø= IF, W LDA, N )Y STA, W 1+ LDA, INY, N )Y STA, THEN,
  Ø # LDY,
  CLC, IP LDA, ( OVERWRITTEN INTERPRETER CODE ) RTS, -->

SCR # 245
 ( REFS --2 )
 : REFS 8Ø OUT ! ( PRINT A LIST OF REFERENCED WORDS )
    CR REF.LIST BEGIN OUT @ C/L > IF CR Ø OUT ! ENDIF
             DUP @ FFFF = Ø=
              WHILE DUP @ 2 + NFA ID. SPACE 2 + REPEAT
 DROP CR ;
 : NONREFS  ( PRINT A LIST OF NON-REFERENCED WORDS )
 8Ø OUT ! CONTEXT @ @ >R
 BEGIN  OUT @ C/L > IF CR Ø OUT ! ENDIF
    REF.LIST BEGIN DUP @ FFFF = OVER @ R PFA CFA = OR Ø=
                  WHILE 2 + REPEAT
    @ FFFF =
      IF R ID. SPACE SPACE ENDIF
    R> PFA LFA @ DUP >R Ø= ?TERMINAL OR
 UNTIL R> DROP ;  -->

SCR # 246
 ( REFS --3 )
CODE +REFS  ( PATCH STORE.REF CODE TO INNER INTERPRETER )
                    2Ø # LDA, NEXT ØB + STA,
  'T STORE.REF 1ØØ /MOD # LDA, NEXT ØD + STA,
                         # LDA, NEXT ØC + STA,
   NEXT JMP,
CODE -REFS  ( RESTORE INNER INTERPRETER )
        18 # LDA, NEXT ØB + STA,
        A5 # LDA, NEXT ØC + STA,
        Ø6 # LDA, NEXT ØD + STA,
        NEXT JMP,
 : XREFS    ( EMPTY REFERENCE BUFFER )
   ASSEMBLER NEXT ØB + C@ 18 =
     IF REF.LIST 4ØØ FF FILL
     ELSE ."   CANT--IN +REFS MODE" ENDIF ;
 FORTH DECIMAL  ;S
```

REFS could also be used to help pinpoint areas which are retarding a FORTH application. You could insert the sequence XREFS +REFS at some key point in a colon definition (perhaps just before a loop that appears to be operating slowly) and further in the definition place the word -REFS (perhaps at the completion of the loop). The REFS list would then contain the candidates which could be coded in machine language to speed up the loop.

Try executing the following:

XREFS +REFS -REFS REFS

and see how little it takes for FORTH just to keep body and soul together.

The FORTH Interest Group 6502 Fig-FORTH inner interpreter (its address is given by the Assembler constant NEXT) is shown disassembled in Program 2. In this particular implementation of FORTH, NEXT is $0642 and the code that is overwritten by a JSR is at $064D-$064F. The three bytes overwritten form the last three bytes of the patch jumped to (line 15, Screen 244). To use REFS on other FORTH implementations, changes would be made here and in lines two through four and lines seven through nine of Screen 246.

REFS Functions

```
+REFS    begin adding to accumu-
                lated list of
                referenced words.
-REFS    stop adding to the list.
XREFS    clear out the list.
REFS     print the list.
NONREFS  print all other words in
                the dictionary.
```

Program 2. 6502 Inner Interpreter

```
0642 A0 01      LDY #$01
0644 B1 06      LDA (IP),Y
0646 85 02      STA W+1
0648 88         DEY
0649 B1 06      LDA (IP),Y
064B 85 01      STA W
064D 18         CLC
064E A5 06      LDA IP
0650 69 02      ADC #$02
0652 85 06      STA IP
0654 90 02      BCC $0658
0656 E6 07      INC IP+1
0658 4C 00 00   JMP $0000
```

CAPUTE!:

Modifications Or Corrections To Previous Articles

PET Compactor

The following lines were inadvertently left out of "Machine Language Compactor," July 1982, p. 159.

```
0508 52 0C A9 50 9D 51 0C A9
0608 88 0B C9 16 B0 48 AD 87
0708 20 63 F5 20 F7 09 20 CE
0808 20 F0 DB C9 8F D0 0D 20
0908 FF 20 CC FF EE 7F 0B 4C
0A08 E8 4C FE 09 CA 8E 30 0A
0B08 00 0D 0D 4F 55 54 50 55
```

The author sent in the following modification to Compactor which allows a BASIC program to be reduced even further in size by altering the way that ON GOTO is compacted. To add this patch, change the following two lines:

```
07E9   4C 63 03
089C   4C 8A 03
```

and add these lines:

```
0363 AD 79 0B C9 91 D0 05 A9
036B 01 8D 90 0B AD 79 0B C9
0373 89 F0 03 4C F0 07 AD 90
037B 0B D0 06 AD 79 0B 4C EC
0383 07 AD 79 0B 4C F0 07 A9
038B 00 8D 90 0B AD 82 0B 4C
0393 9F 08 00 00 00 00 00 00
```

Gold Miner Game, Atari Version

(also see note in box below)

Two RESTORE commands should be changed in the Atari version of this game (see p. 28, **COMPUTE!**, July 1982). Change line 840 to RESTORE 850 and the RESTORE in line 1130 should read RESTORE 1190.

Our July issue featured an action game and accompanying article entitled "Gold Rush!" This game should not be confused with an arcade-graphics game for both the Apple and Atari of the same name produced and marketed by Sentient Software of Aspen, Colorado. No comparison or confusion was intended regarding the products of Sentient Software, and readers should be aware that these two games are entirely different products. In any future use of this article and action game, we will refer to it as "Gold Miner."

COMPUTE!'s Listing Conventions

Many of the programs which are listed in **COMPUTE!** use special keys (cursor control keys, color keys, etc.). To make it easy to tell *exactly* what should be typed in when copying a program into the computer, we have established the following listing conventions.

For The Atari

In order to make special characters, inverse video, and cursor characters easy to type in, **COMPUTE!** magazine's Atari listing conventions are used in all the program listings in this magazine.

Please refer to the following tables and explanations if you come across an unusual symbol in a program listing.

Atari Conventions

Characters in inverse video will appear like: INVERSE VIDEO
Enter these characters with the Atari logo key, {⅄}.

When you see	Type	See	
(CLEAR)	ESC SHIFT <	↖	Clear Screen
(UP)	ESC CTRL -	↑	Cursor Up
(DOWN)	ESC CTRL =	↓	Cursor Down
(LEFT)	ESC CTRL +	←	Cursor Left
(RIGHT)	ESC CTRL *	→	Cursor Right
(BACK S)	ESC DELETE	◀	Backspace
(DELETE)	ESC CTRL DELETE	◧	Delete character
(INSERT)	ESC CTRL INSERT	◨	Insert character
(DEL LINE)	ESC SHIFT DELETE	⬒	Delete line
(INS LINE)	ESC SHIFT INSERT	⬓	Insert line
(TAB)	ESC TAB	▶	TAB key
(CLR TAB)	ESC CTRL TAB	◧	Clear tab
(SET TAB)	ESC SHIFT TAB	◨	Set tab stop
(BELL)	ESC CTRL 2	◳	Ring buzzer
(ESC)	ESC ESC	⇐	ESCape key

Graphics characters, such as CTRL-T, the ball character • will appear as the "normal" letter enclosed in braces, e.g. {T}.

A series of identical control characters, such as 10 spaces, three cursor-lefts, or 20 CTRL-R's, will appear as {10 SPACES}, {3 LEFT}, {20 R}, etc. If the character in braces is in inverse video, that character or characters should be entered with the Atari logo key. For example, {♥} means to enter a reverse-field heart with CTRL-comma, {5 ▥} means to enter five inverse-video CTRL-U's.

For PET/CBM/VIC

Generally, any PET/CBM/VIC program listings will contain bracketed words which spell out any special characters: {DOWN} would mean to press the cursor-down key; {3DOWN} would mean to press the cursor-down key three times.

To indicate that a key should be *shifted* (hold down the SHIFT key while pressing the other key), the key would be underlined in our listing. For example, <u>S</u> would mean to type the S key while holding the shift key. This would result in the "heart" graphics symbol appearing on your screen.

Sometimes in a program listing, especially within quoted text when a line runs over into the next line, it is difficult to tell where the first line ends. How many times should you type the SPACE bar? In our convention, when a line breaks in this way, the ~ symbol shows exactly where it broke. For example:

```
100 PRINT "TO START THE GAME  ~
    YOU MAY HIT ANY OF THE KEYS
    ON YOUR KEYBOARD."
```

shows that the program's author intended for you to type two spaces after the word *GAME.*

For The Apple

Programs listed as "Microsoft" are written for the PET/CBM, Apple, OSI, etc. Although the programs are general in nature, you may need to make a few changes for them to run correctly on your Apple. Microsoft BASIC programs written for the PET/CBM sometimes contain special cursor control characters. The following table shows equivalent Apple words. Notice that these Apple commands are *outside* quotations (and even separate from a PRINT statement). PRINT"[RVS]YOU WON" becomes INVERSE: PRINT"YOU WON":NORMAL

[CLEAR[(Clear Screen) HOME

[DOWN] (Cursor down)
Apple II +: Call -922
POKE 37,PEEK(37)+(PEEK(37)<23)

[UP] (Cursor up)
POKE 37,PEEK(37)-(PEEK(37)>0))

[LEFT] (Cursor left) PRINT CHR$(8);

[RIGHT] (Cursor right)
PRINT CHR$(21)

[RVS] (Inverse video on. Turns off automatically after a carriage return. To be safe, turn off inverse video after the print statement with NORMAL unless the PRINT statement ends with a semicolon.)
INVERSE

[OFF] (Inverse video off) NORMAL

Shifted characters can represent either graphics characters or uppercase letters. If within text, just use the non-shifted character, otherwise substitute a space. Some "generalized" programs contain a POKE such as POKE 59468,14. Omit these from the program when typing it in. One final note: you will probably want to insert a question mark or colon within an INPUT prompt. PET/CBM and many other BASICs automatically print a question mark:

INPUT "WHAT IS YOUR NAME";N$
becomes
INPUT "WHAT IS YOUR NAME?";N$

All Commodore Machines

Clear Screen	{CLEAR}	**Cursor Left**	{LEFT}
Home Cursor	{HOME}	**Insert Character**	{INST}
Cursor Up	{UP}	**Delete Character**	{DEL}
Cursor Down	{DOWN}	**Reverse Field On**	{RVS}
Cursor Right	{RIGHT}	**Reverse Field Off**	{OFF}

VIC Conventions

Set Color To Black	{BLK}	**Function Two**	{F2}
Set Color To White	{WHT}	**Function Three**	{F3}
Set Color To Red	{RED}	**Function Four**	{F4}
Set Color To Cyan	{CYN}	**Function Five**	{F5}
Set Color To Purple	{PUR}	**Function Six**	{F6}
Set Color To Green	{GRN}	**Function Seven**	{F7}
Set Color To Blue	{BLU}	**Function Eight**	{F8}
Set Color To Yellow	{YEL}	**Any Non-implemented Function**	{NIM}
Function One	{F1}		

8032/Fat 40 Conventions

Set Window Top	{SET TOP}	**Erase To Beginning**	{ERASE BEG}
Set Window Bottom	{SET BOT}	**Erase To End**	{ERASE END}
Scroll Up	{SCR UP}	**Toggle Tab**	{TGL TAB}
Scroll Down	{SCR DOWN}	**Tab**	{TAB}
Insert Line	{INST LINE}	**Escape Key**	{ESC}
Delete Line	{DEL LINE}		

©

Turn To The Future With COMPUTE! Publications

The Beginner's Guide To Buying A Personal Computer

A Novice's handbook of useful, helpful information designed to teach you the basics of evaluating and selecting a personal computer. Written in plain English for the interested beginner. Complete with personal computer specification charts and buyer's guide. Applicable to home, educational, and small business buyers. ISBN 0-942386-03-5. Paperback. $3.95.

Inside Atari DOS

From the authors of the Atari Disk Operating System, an exciting step-by-step guide to the DOS software. Complete with listings of commented source code and detailed explanations of each module of code. Author: Bill Wilkinson, Optimized Systems Software, Inc. Spiral bound for ease of access to listings. For intermediate to advanced Atari users. ISBN 0-942386-02-7. Paperback. $19.95.

COMPUTE!'s First Book Of Atari

192 pages of useful, informative applications and programs from **COMPUTE!** Magazine issues now out of print. Includes previously unpublished information including Memory Map. Contents include such articles and programs as "Adding A Voice Track to Atari Programs," "Designing Your Own Atari Graphics Modes," and "Inside Atari BASIC." Spiral bound for ease of access to listings. For beginner level to advanced Atari users. ISBN 0-942386-00-0. Paperback. $12.95.

COMPUTE!'s Second Book Of Atari

With over 200 pages of unpublished Atari information specially selected by the editors of **COMPUTE!** Magazine, this new release is an excellent resource for Atari owners and users. Spiral bound. ISBN 0-942386-06-X. Paperback. $12.95.

COMPUTE! Books invites dealer inquiries. Call the Toll Free Number below for Dealer Information.

To Order COMPUTE! Books

Send coupon to
COMPUTE! Books, P.O. Box 5406, Greensboro, NC 27403 USA

For fastest service, in the US call
Toll Free 800-334-0868
In NC call 919-275-9809.

All orders must be prepaid (money order, check or charge). All payments must be in US funds. (Outside the US add $4.00 shipping and handling for air mail, $2.00 for surface mail.) NC residents add 4% sales tax.

☐ Payment Enclosed
Please charge my ☐ VISA ☐ MasterCard ☐ American Express

Account No. ____________ Expires ____/____

Please send me:

Quan.		Price		Shipping/Handling	
____	**Beginner's Guide**	$ 3.95 ea.	+	$1.00 ea.	______
____	**First Book of Atari**	12.95 ea.	+	2.00 ea.	______
____	**Inside Atari DOS**	19.95 ea.	+	2.00 ea.	______
____	**Second Book of Atari**	12.95 ea.	+	2.00 ea.	______
				Total	______

Name ____________
Address ____________
City ________ State ______ Zip ______
Country ____________

Allow 4-5 weeks for delivery. Foreign surface delivery allow 2-3 months. Dept. A

COMPUTE! Back Issues

Here are some of the applications, tutorials, and games from available back issues of **COMPUTE!**. Each issue contains much, much more than there's space here to list, but here are some highlights:

January 1981: Load PET Programs Into The Apple II, Player-Missile Graphics for Atari, The Atari DOS, The Kernel of the OSI Operating System, Fixing LOADing Problems on the PET, Spooling with the PET Disk, Expanding KIM.

February 1981: Simulating PRINT USING, Using the Atari as a Terminal for Telecommunications, Attach a Printer to the Atari, Double Density Graphing on C1P, Commodore Disk Systems, PET Crash Prevention, A 25¢ Apple II Clock.

May 1981: Named GOSUB/GOTO in Applesoft, Generating Lower Case Text on Apple II, Copy Atari Screens to the Printer, Disk Directory Printer for Atari, Realtime Clock on Atari, PET BASIC Delete Utility, PET Calculated Bar Graphs, Running 40 Column Programs on a CBM 8032.

June 1981: Computer Using Educators (CUE) on Software Pricing, Apple II Hires Character Generator, Ever- expanding Apple Power, Color Burst for Atari, Mixing Atari Graphics Modes 0 and 8, Relocating PET BASIC Programs, An Assembler In BASIC for PET, QuadraPET: Multitasking?

July 1981: Home Heating and Cooling, Animating Integer BASIC Lores Graphics, The Apple Hires Shape Writer, Adding a Voice Track to Atari Programs, Machine Language Atari Joystick Driver, Four Screen Utilities for the PET, Saving Machine Language Programs on PET Tape Headers, Commodore ROM Systems, The Voracious Butterfly on OSI.

August 1981: Minimize Code and Maximize Speed, Apple Disk Motor Control, A Cassette Tape Monitor for the Apple, Easy Reading of the Atari Joystick, Blockade Game for the Atari, Atari Sound Utility, The CBM "Fat 40," Keyword for PET, CBM/PET Loading, Chaining, and Overlaying.

October 1981: Automatic DATA Statements for CBM and Atari, VIC News, Undeletable Lines on Apple, PET, VIC, Budgeting on the Apple, Switching Cleanly from Text to Graphics on Apple, Atari Cassette Boot-tapes, Atari Variable Name Utility, Atari Program Library, Train your PET to Run VIC Programs, Interface a BSR Remote Control System to PET, A General Purpose BCD to Binary Routine, Converting to Fat-40 PET.

November 1981: SuperPet: A Preview, Japanese Micros: A First Look, Introduction to Binary Numbers, An Apple Primer, Page Flipper for Apple, An Atari Database System, A Program for Writing Programs on the Atari, Atari Textplot, OSI Relocation, The PET Speaks, Inversion Partitioning, A Personal News Service on PET, Bits, Bytes, and Basic Boole.

December 1981: Saving Fuel $$ (Multiple Computers: versions for Apple, PET, and Atari), Unscramble Game (multiple computers), Maze Generator (multiple computers), Animating Applesoft Graphics, A Simple Printer Interface for the Apple II, A Simple Atari Wordprocessor, Adding High Speed Vertical Positioning to Atari P/M Graphics, OSI Supercursor, A Look At SuperPET, Supermon for PET/CBM, PET Mine Maze Game.

January 1982: Invest (multiple computers), Developing a Business Algorithm (multiple computers), Apple Addresses, Lowercase with Unmodified Apple, Cryptogram Game for Atari, Superfont: Design Special Character Sets on Atari, PET Repairs for the Amateur, Micromon for PET, Self-modifying Programs in PET BASIC, Tinymon: a VIC Monitor, Vic Color Tips, VIC Memory Map, ZAP: A VIC Game.

February 1982: Insurance Inventory (multiple computers), Musical Transposition (multiple computers), Multitasking Emulator (multiple computers), Disassemble Apple Programs from BASIC, Plotting Polar Graphs on Apple, Atari P/M Graphics Made Easy, Atari PILOT, Put A Rainbow in your Atari, Marquee for PET, PET Disk Disassembler, VIC Paddles and Keyboard, VIC Timekeeping.

March 1982: Word Hunt Game (multiple computers), Infinite Precision Multiply (multiple computers), Atari Concentration Game, VIC Starfight Game, CBM BASIC 4.0 To Upgrade Conversion Kit, Apple Addresses, VIC Maps, EPROM Reliability, Atari Ghost Programming, Atari Machine Language Sort, Random Music Composition on PET, Comment Your Apple II Catalog.

April 1982: Track Down Those Memory Bugs (multiple computers), Shooting Stars Game (multiple computers), Intelligent Input Subroutines (multiple computers), Ultracube for Atari, Customizing Apple's Copy Program, Using PET/CBM In The High School Physics Lab, Grading Exams on a Microcomputer (multiple computers), Atari Mailing List, Renumber VIC Programs The Easy Way, Browsing the VIC Chip, Disk Checkout for PET/CBM.

May 1982: VIC Meteor Maze Game, Atari Disk Drive Speed Check, Modifying Apple's Floating Point BASIC, Fast Sort For PET/CBM, Extra Atari Colors Through Artifacting, Life Insurance Estimator (multiple computers), PET Screen Input, Getting The Most Out Of VIC's 5000 Bytes.

Home and Educational COMPUTING! (Fall 1981 and Summer 1981 – count as one back issue): Exploring The Rainbow Machine, VIC As Super Calculator, Custom Characters, Alternate Screens, Automatic Line Numbers, Using The Joystick (Spacewar Game), Fast Tape Locater, Window, VIC Memory Map.

Back issues are $3.00 each or six for $15.00. Price includes freight in the US. Outside the US add $1.00 per magazine ordered for surface postage. $3.00 per magazine for air mail postage. All back issues subject to availability.

In the Continental US call
TOLL FREE 800-334-0868
(In NC Call 919-275-9809)

Or write to **COMPUTE!** Back Issues, P.O. Box 5406, Greensboro, NC 27403 USA. Prepayment required in US funds. MasterCard, Visa and American Express accepted. North Carolina Residents add 4% sales tax.

New Products

BIT 3 Computer Corporation Introduces Two Cards For The Atari

BIT 3 Computer Corporation introduced new word processing capabilities for the Atari 800 with the announcement of two new products – Full-View 80 Display Card and the 32K Memory Plus – which combine to make the Atari 800 a 48K 80-column computer.

The Full-View 80 gives the Atari 80-column capability with upper- and lowercase characters, while retaining the normal Atari 40-column graphics mode. Switching between the two modes is via an on-board softswitch.

The lowercase characters have full descenders, making them easy to read on the monitor screen. The Full-View 80 is fully compatible with BASIC and machine language. 80-column word processing is available through the 80-column version of Letter Perfect.

The 32K Memory Plus card operates with the Atari 400 as well as the Atari 800. It doubles the memory capacity of the Atari 400. On the Atari 800 it allows 48K capacity, while freeing up a slot for the Full-View 80, so that both 80-column display and 48K of memory are available together.

The Full-View 80 is independent of the 32K Memory Plus and will operate in a 32K environment.

See the Full-View 80 Display Card and the 32K Memory Plus Card at your Atari dealer or contact BIT 3 for information.

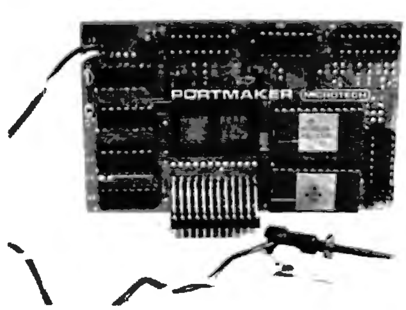
The price of Full-View 80 is $349.00; of 32K Memory Plus, $179.00.

BIT 3 Computer Corp.
8120 Penn Ave. South
Minneapolis, MN 55431
(612)881-6955

Portmaker From Microtech

Portmaker is a small dual serial port I/O board that plugs into any 2532 type ROM socket to add serial RS232 capability to its host computer. Portmaker contains two 6850 ACIA communication parts complete with buffer circuits and a simple baud rate clock. A ROM socket is also included allowing use of the original ROM for all but its highest 16 locations. Portmaker is designed to operate with 650X and 680X computers including the Commodore PET/CBM and the Rockwell AIM.

A special intelligent communications software package to use with Portmaker is available for the PET/CBM. The Standard Terminal Communications Package (STCP), from Eastern House Software, converts the PET/CBM into a data management center. Disk files can be serially transmitted. Received data can be recorded on disk or printer (or both) with complete control of all peripherals. A realtime clock with display and alarm capability is also included. All STCP routines can be controlled from BASIC, allowing automated telemetry. XON and XOFF protocall control codes are provided.



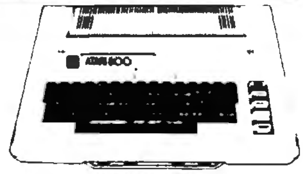
Portmaker is available in vertical or horizontal mount for $69.95. A complete package, including a special Portmaker, cable, and the STCP software called COMPACK, is available for $129.95.

CGRS Microtech
P.O. Box 102
Langhorne, PA 19047
(215)757-0284

Computerware Announces Three Releases For Radio Shack Color Computer

Mazerace is a board-type game that involves both chance and computer mail order west

ATARI



400

16K ... $269
32K ... $389
48K ... $489

410 Recorder	$76.00
810 Disc Drive	$449.00
822 Printer	$269.00
825 Printer	$589.00
830 Modem	$159.00
820 Printer	$259.00
850 Interface	$169.00
CX40 Joy Stick	$18.00
CX853 16K RAM	$77.95

800

16K ... $649
32K ... $724
48K ... $769

Microtek 16K RAM	$74.95
Microtek 32K RAM	$119.95
Ramdisk (128K)	$429.95
Intec 48K Board	$219.95
Intec 32K	$119.95
One year extended warranty	$70.00
481 Entertainer	$69.00
482 Educator	$130.00
483 Programmer	$49.00
484 Communicator	$344.00

FRANKLIN ACE 1000

64K Personal Computer
Hardware, software and peripheral compatable with the Apple II and even has some features not found on the Apple.

Call now for best price!!

HOT ATARI GAMES

PAC-MAN	$35.00
Centipede	$35.00
Caverns of Mars	$32.00
Asteroids	$29.00
Missile Command	$29.00
Star Raiders	$39.00
Canyon Climber	$25.00
Protector	$24.00
Mouskattack	$31.00
Jawbreaker	$27.00
Ghost Hunter	$24.00

K-BYTE
ROM CARTRIDGE GAMES FOR YOUR ATARI

Krazy Shoot Out	$39.00
K-razy Kritters	$39.00
K-razy Antics	$39.00
K-star Patrol	$39.00

STICK STAND
$6.99

ARCADE ACTION FROM YOUR ATARI JOYSTICK

PERCOM

Finally, a dual disk drive for the Atari 800/400. Both single and dual drive models read both sides of the disk! Will read all disks written for Atari 810. *CALL FOR INTRODUCTORY PRICE!*

Telecommunications Modems

Hayes	
Smart	$239.00
Chronograph	$199.00
Micromodem II	$279.00
Micromodem 100	$309.00
Novation Auto	$239.00
D Cat	$169.00
Cat	$159.00
Anchor Modem	$79.00

AMDEK MONITORS

300G	$169.00
Color I	$339.00
Color II	$699.00
Color III	$429.00

OTHERS

Zenith 9" (Green)	$119.00
BMC 12" Green	$85.00

NEC

COMPUTERS

8001-A	$749.00
8031	$749.00
8012	$549.00

PRINTERS

8023	$549.00
7710/7730	$2399.00
3510/3530	$1789.00

MONITORS

JB-1201	$179.00
JC-1201	$349.00
JC-1202	$899.00

VISICORP

VISICALC	
Apple II +	$189.00
Atari	$189.00
Commodore	$189.00
IBM	$189.00

Also available are:

VISIDEX — VISIPLOT
VISIFILE — VISITERM
VISIPACK — VISITREND

SOFTWARE

We stock manufacturer's and third party software for most all computers on the market! Call today for a copy of our new

CATALOG

You'll find programs by Atari, APX, Data Soft Crystal Software, EPYX, Synapse, OnLine, Arcade Plus, K-BYTE, Magis, Canadian Micro Professional Software, Creative Software, BPI, VISICORP, Commodore, NEC and of course our own!

Maxell Disks

MD I (box of 10)	$36.00
MD II (box of 10)	$46.00
MFD I (8")	$44.00
MFD II (8" Double Density)	$54.00
Syncom (box of 10)	$29.00

Computer Covers

Atari 400	$6.99	Commodore VIC-20	$6.99
Atari 800	$6.99	Commodore 8032	$14.99
Atari 810	$6.99	Commodore 8050/4040	$10.99

west 800-648-3311

In Nevada CALL (702)588-5654

P.O. Box 6689
Stateline,
NV. 89449

strategy. The playing field is an 18 by 18 hexagon matrix that is partially filled with obstacles. Either one person against the computer or two people can play, with the computer or the players randomly scrambling the playing field to keep the action exciting. Mazerace uses high resolution graphics and requires joysticks and Extended BASIC.

Mazerace costs $17.95 on cassette or $22.95 on disk (plus $2.00 for shipping and handling).

****

3D Drawing Board is a tool for education, entertainment, or serious projects. It helps you draw objects in three dimensions and then rotate, change elevation, size, and distance. The drawings can be saved to tape or disk for future use. A complete instruction manual accompanies the program to make it easy to use.

3D Drawing Board costs $24.95 on cassette or $29.95 on disk (plus $2.00 for shipping and handling).

****

Storm brings the arcade game to your home. It has 135 levels of play, fast action, graphics, and sound.

Storm costs $24.95 on cassette or $29.95 on disk (plus $2.00 for shipping and handling).

All three packages are available directly from:

Computerware
1472 Encinitas Blvd., Box 668
Encinitas, CA 92024
(714)436-3512

Mailgram Messages From Personal Computers

Over 18,000 subscribers to The Source are now able to compose and send Mailgram Messages direct from their personal computers.

Messages may be sent anywhere in the United States, including Alaska and Hawaii. If entered in The Source by 4 p.m., EST, they are virtually guaranteed next business day delivery.

The Mailgram Messages are electronically forwarded by The Source to Western Union's largest Mailgram processing center. They are then routed by zip code to the serving post office nearest the addressee, where they are printed, enveloped and given preferential delivery treatment.

Only senders and recipients are permitted to read the Mailgram Messages.

The cost of sending Mailgram Messages is billed to subscribers monthly, along with other charges from The Source.

To use the new service, subscribers sign on to The Source, using a personal computer, data terminal, or communicating word processor. They then type MGRAM and are prompted step-by-step to enter names, addresses and messages. Once messages are composed, subscribers can review them on their terminal screens exactly as they will be reproduced as Mailgrams. Changes in the text can then be made, if necessary, before transmission.

Subscribers may send one Mailgram Message to a single address, or they can instruct that the same, or different, messages be sent to multiple addresses. Discount rates apply to quantity mailings of the same message. There is no limit to the number of Mailgram Messages that can be sent.

Special features of the new service include:

- Being able to receive one confirmation copy of the Mailgram Message, or to receive duplicate copies that include the names and addresses of all Mailgram Messages sent.
- Saving the message in a personal file on The Source for later reference.
- Creating "text files" on The Source for frequently used messages, to facilitate their re-sending.
- Arranging for the Mailgram Message Processing Service to enter names and addresses of large lists, at a cost of 35¢ each.
- Storing lengthy address files with the Message Processing Service at a nominal charge.

The rate for sending a single Mailgram Message on The Source is $5.15 for messages up to 100 words, with an additional $1.00 charge for each additional 100 words.

The discounted prices for multiple Mailgram Messages with identical text are: 2-25, $4.25 each; 26-100, $3.50 each, and 101-200, $3.00 each. In every case, another $1.00 is charged for each additional 100 words. Reduced rate quotations are available from the Message Processing Center for quantities above 200.

The cost for confirmation copies, which are delivered at the same time as the Mailgram Messages, is $2.00 each, with another $1.00 charged for each additional 100 words.

The Source can be reached by a local telephone call from more than 360 metropolitan areas, or via a nationwide WATS link-up for subscribers in outlying regions.

Subscriber costs include a one-time $100 registration fee, and standard hourly usage charges of $18 weekdays, $5.75 evenings and weekends, and $4.25 after midnight. The costs of sending Mailgram Messages are in addition to standard usage fees.

Source Telecomputing Corp.
1616 Anderson Road
McLean, VA 22102
(703)734-7500

HUNTINGTON COMPUTING

NEW ITEMS

#8800	Adam & Eve Paddles (Tech Designs)	$33.89
#3259	Apple Mechanic (Beagle Bros.)	$24.99
#8660	Assembler Teacher (Compu Works)	$38.19
#1077	Bandits (Sirius)	$29.69
#3109	Business Bookkeeping System (Dakin 5)	$335.69
#1118	Cannonball Blitz (On-Line)	$29.69
#1457	Firebug (Muse)	$21.19
#2563	First Class Mail (Continental)	$63.69
#745	Firmware Rom (Novation)	$24.59
#8207	Format Rom (Soft Control)	$42.39
#1456	Frazzle (Muse)	$21.19
#1207	Global Program Line Editor (Synergis)	$55.19
#4403	Gold Rush (Sentient)	$29.69
#8002	Graftrax Plus (Epson)	$60.00
#1409	Graphics Combo Pkg (Sublogic)	$101.79
#1078	Jellyfish (Sirius)	$25.39
#1079	Kabul Spy (Sirius)	$29.69
#1148	Lafpak (On-Line)	$29.69
#703	The Manager (Omega)	$29.69
#1111	Mouskattack (On-Line)	$35.09
#3302	P F S Graph (Software)	$106.19
#7558	Personal Finance Master (Spectrum)	$63.69
#1018	Pursuit of Graf Spee (Strategic Sim.)	$50.89
#1164	Pie Writer - Multi (Hayden)	$127.39
#1163	Pie Writer - Standard (Hayden)	$127.39
#9487	The Programmer (Advanced Operating)	$420.69
#224	Pseudo Disk - Pascal (Saturn Sys.)	$33.09
#2183	Queen of Phobos (Phoenix)	$29.69
#221	64K Ramboard (Saturn Systems)	$361.00
#1681	S A G A #1-Adventureland (Adventure International)	$25.39
#9581	Slide Show (C & H Video)	$42.39
#8208	Sort Rom (Soft Control)	$33.89
#8700	Space Tablet 3-Axis (MCS)	$335.69
#8701	Space Tablet 4-Axis (MCS)	$446.19
#8640	Speed Read Plus (Optimized)	$50.89
#4657	Swordthrust #6-Eternal Curse (CE Soft)	$25.39
#4658	Swordthrust #7-Hall of Alchemie (CE)	$25.39
#2265	Visifactory (Micro Lab)	$63.69

SPECIAL
With any order over $49.00 (COD's excluded) receive FREE a copy of the program Mail Ordering (on disk for Apple) This program will allow you to make up mail orders with just a few keystrokes Exclusively available from Huntington Computing Ask for stock number 998

ADVENTURE INTERNATIONAL

#1664	Adventures # 1-2-3	$34.99
#1665	Adventures # 4-5-6	$34.99
#1666	Adventures # 7-8-9	$34.99
#1677	Adventures # 10-11-12	$33.89
#1668	Apple Spice	$26.89
#1671	Poker Tournament	$17.79
#1681	S A G A #1 Adventureland	$25.39
#1682	S A G A #2 Pirate Adventure	$25.39
#1678	Stone of Sisyphus	$25.39

AVALON HILL

#2050	B-1 Nuclear Bomber (Cassette)	$12.99
#2057	Computer Acquire (Cassette)	$16.99
#2063	Computer Stocks & Bonds	$13.59
#2056	Conflict 2500 (Cassette)	$12.69
#2060	Empire of the Overmind (Cassette)	$25.49
#2059	Empire of the Overmind	$24.99
#2055	Lords of Karma (Cassette)	$16.99
#2058	Major League Baseball	$25.49
#2051	Midway Campaign (Cassette)	$12.69
#2052	North Atlantic Convoy (Cassette)	$12.69
#2053	Nuke War (Cassette)	$12.69
#2054	Planet Miners (Cassette)	$12.69
#2062	Tanktics (Cassette)	$20.39
#2061	Tanktics	$24.59

AVANT-GARDE

#7003	Chambers of Xenobia	$13.49
#7000	Creativity Tool Box	$38.19
#7005	Five Great Games	$25.39
#7004	Five More Great Games	$25.39
#7008	Race for Midnight	$25.39
#7006	Sentence Diagramming	$21.19
#7010	Sounds & Scrolling	$16.09
#7011	Super Draw & Write	$15.79
#7009	Super Shape Draw	$16.89
#7015	Word Scrambler	$16.89
#7016	Zero Gravity Pinball	$25.39

Softlights

By Fred Huntington

It's an exciting time around the Huntington household this month. We're very proud to announce the birth of our seven-pound five-ounce baby boy, Dale, born on June 6, 1982 in Visalia, California. Baby and Mama are doing just great and Melody (our three-year-old) loves him and calls him "My baby."

The other big news is that I have resigned my position as school principal so that I may devote full time to Huntington Computing.

We're instituting lots of changes to improve efficiency and speed. Our goal is to get everything out of the door within twenty-four hours with no backorders.

To celebrate the excitement, we're offering the following specials: (Always an excuse to run a sale.)

#9010 **Wurst of Huntington**
List $19.99, **now $9.99**

#4403 **Gold Rush (Sentient)**
List $34.95, **now $25.69**

#1967 **Arcade Machine (Broderbund)**
List $44.95, **now $34.49**

#1970 **Chop Lifter (Broderbund)**
List $34.95, **now $25.69**

#4002 **Verbatim Datalife Disks**
(w/plastic box & hub rings)
List $45.00, **now $25.99**

#9140 **Great Grandma Huntington T-Shirt - $5.00**

GREAT GRANDMA SAYS

Great Grandma Huntington once told me about a computer the Russians invented that was so smart it defected to the West!

Great Grandma also said, "Please buy little Freddy's Wurst of Huntington, because he personally gets two dollars for every one he sells. And, it's good. Just read the review in October 1981 issue of Softalk.

Watch next month's Softlights for the winners of the Great Grandma Huntington contest. There were some fantastic entries.

BRODERBUND

#1960	Alien Typhoon	$21.19
#1963	Apple Panic	$25.39
#1967	Arcade Machine	$39.49
#1966	David's Midnight Magic	$30.69
#1952	Galactic Empire	$22.99
#1954	Galactic Revolution	$22.99
#1953	Galactic Trader	$22.99
#1951	Galaxy Wars	$22.99
#2000	Genetic Drift	$25.39
#1957	Golden Mountain	$16.89
#1950	Hyper Head-on	$21.99
#1962	Labyrinth	$25.39
#1961	Payroll	$335.69
#1968	Star Blazer	$27.09
#1956	Tank Command	$12.99
#1955	Tawala's Last Redoubt	$25.39
#1959	Track Attack	$26.89

MISCELLANEOUS

#8800	Adam & Eve Paddles (Tech Designs)	$33.89
#7890	Apple-cillin II (XPS)	$33.89
#3000	Bookkeeper (Delta)	$76.39
#9700	Castles of Darkness (Logical)	$30.69
#3001	Checkwriter (Delta)	$33.99
#9640	Colorblind (Energy)	$30.69
#9014	Computer Almanac (Huntington Computing)	$24.99
#6380	Crossword Magic	$39.99
#4401	Cyborg (Sentient)	$28.99
#9880	Deadline (Infocom)	$42.39
#9742	Electric Semicolon	$97.69
#9900	Financial Facts (Hanson)	$22.99
#9600	The Game Show	$34.29
#6870	Handwriting Analysis (Micro Lipp)	$16.89
#9840	Jabbertalky (Mind Toys)	$26.29
#9580	The Menu II (C & H Video)	$33.89
#9380	The Menu Generator	$35.09
#4400	OO-Topos (Sentient)	$27.99
#6240	Paddle-Adapple (Southern California)	$26.89
#7650	Pornopoly (CCI)	$25.39
#7920	Property Management	$292.49
#3400	Raster Blaster (Budgeco)	$25.29
#9500	Recipe Handler (Soft Touch)	$33.09
#9841	Ricochet (Mind Toys)	$16.89
#6600	Rubik's Cube (Software Alternatives)	$16.89
#9620	Shadow Hawk One (Horizon)	$43.89
#3380	Space War I (Galaxy)	$33.89
#4252	Star Blaster (Piccadilly)	$25.39
#9180	Starship Commander (Voyager)	$35.89
#4251	Suicide (Piccadilly)	$26.29
#9680	Taxman (Hal Labs)	$19.49
#4850	Time Lord (Ramware)	$25.39
#9012	Understand Yourself (Huntington)	$24.99
#9740	Volcanoes (Earthware)	$43.49
#9741	Volcanoes - Educational Version (Earthware)	$44.49
#9880	Zork I (Infocom)	$33.89
#9881	Zork II (Infocom)	$33.89

ON-LINE

#1120	Cranston Manor	$29.69
#1104	Crossfire	$26.29
#1126	The Dictionary	$87.89
#1125	M M S II	$43.89
#1115	Marauder	$29.69
#1101	Mission Asteroid	$17.99
#1113	Missile Defense	$21.19
#1102	Mystery House	$21.19
#1149	Pegasus II	$25.39
#1117	Sabotage	$21.19
#1105	Screenwriter II	$99.99
#1121	Soft Porn Adventure	$25.39
#1122	Threshold	$33.89
#1114	Ultima II	$46.69
#1112	Ulysses	$29.69
#1100	The Wizard and the Princess	$28.89

SIRIUS

#1058	Autobahn	$25.39
#1077	Bandits	$29.69
#1073	Borg	$26.89
#1065	Copts & Robbers	$29.69
#1053	Cyclod	$25.39
#1062	Epoch	$25.39
#1070	Joyport	$67.39
#1052	Minotaur	$29.69
#1066	Outpost	$26.29
#1057	Pulsar II	$25.39
#1071	Snake Byte	$25.39
#1064	Sneakers	$25.39
#1056	Space Eggs	$19.99
#1072	Twerps	$26.89

Call Toll-Free **800-344-5106** (outside California)

HUNTINGTON COMPUTING
Post Office Box 1297
Corcoran, California 93212

Foreign Orders 209-992-4481
In California 800-692-4146

Outside Calif. 800-344-5106

We take **MasterCard, American Express or VISA** (Include card # and expiration date). California residents add 6% tax. Include $2.00 for postage. Foreign and hardware extra Foreign (excluding Canada) remit U.S currency, checks on U.S. banks, use listed charge cards, or make direct wire transfers through Security Pacific Bank, Corcoran, for a $6.00 charge All overseas orders shipped by air Send for free catalog Prices subject to change without notice.

Whole Brain Spelling For The Apple II

Whole Brain Spelling is the first in a series of educational software packages from SubLOGIC. The program has been designed to help the user develop internal visualization skills for improving spelling in a manner as entertaining as it is educationally sound. It effectively utilizes the graphic and color capabilities of the Apple II Plus computer to provide positive user feedback and to emphasize visual aspects of the learning process.

The program itself is extremely user friendly. You can move to any lesson section as desired, choose your own word lists to study, and proceed at your own rate. A main Spelling Menu is accessible from any portion of the program. Lesson instructions are also always available at the touch of a key.

A 2000-word list of practice spelling words is included with Whole Brain Spelling, organized in order of increasing spelling difficulty. Study words can be printed in upper- or lowercase, in any color you choose. And each correct spelling rewards you with a rainbow of varying, multicolored letters until the next word is selected.

The standard Whole Brain Spelling program comes complete with 200 ten-word lists, systematically chosen and organized in order of increasing spelling difficulty. Whole Brain Spelling is also available with supplementary word lists in the following categories: Medical, Scientific, Secretarial, Fairy Tale, and A Child's Garden of Words.

The Medical word lists are helpful to anyone involved in the health sciences or the medical profession, and include categories in general medical terms, diagnosis, anatomy, and drugs.

The Scientific word lists are divided into the categories of general science, earth science, life science, and physical science. Each category is graded into four levels of difficulty.

The Secretarial word lists include commonly used real estate, insurance, legal, commercial, and accounting terms organized into four difficulty levels.

For anyone who is a child at heart, the list of Fairy Tale words should prove irresistible. Taken from Grimm's and other fantasy tales, these words will pique your curiosity as well as enrich your spelling skills.

A Child's Garden of Words gives the preschooler through third grader (ages 5-9) a head start in the development of word visualization skills. These are the words most often encountered in beginning reading lessons, and these lists offer a primer to the main word lists found on the standard Whole Brain Spelling program.

A2-ED1 Whole Brain Spelling is available at most computer stores or from SubLOGIC direct. The A2-ED1 requires 48K memory and either an Apple II Plus or an Apple II with Applesoft in ROM (a color monitor is also recommended). Unless you request otherwise, the main word list will be included with each program ordered. The disk price is $34.95. For direct orders add $1.50 for shipping and specify UPS or first class mail. Illinois residents add 5% sales tax.

SubLOGIC Communications Corp.
713 Edgebrook Drive
Champaign, IL 61820
(217)359-8482

Scholastic Aptitude Test Package For Atari

Program Design has released *Preparing for the SAT* for Atari 400 and 800 computers. The package teaches students how to take the Scholastic Aptitude Test (SAT) and other aptitude and intelligence tests, develops problem-solving skills, and provides practice in answering questions typically found on such tests.

Preparing for the SAT consists of six cassettes, a user's manual, and a copy of the booklet "Making the Grade," all packaged in a special storage container. The six cassettes are:

- "Taking Aptitude Tests" – This cassette explains the purpose of standardized IQ and aptitude tests. It discusses some of the false beliefs surrounding such tests. It describes ways to improve test-taking skills. And it presents a strategy for answering those questions that are most likely to pay off with correct answers.
- "Vocabulary Builders I and II" – These two cassettes are designed to help develop a student's vocabulary and to build skills needed to answer synonym and antonym questions.

Royal Software

YOUR MARKETPLACE FOR:

* ATARI

WE DEAL EXCLUSIVELY IN PRODUCTS FOR THE ATARI (THE BEST) COMPUTER

CLOWNS AND BALLOONS

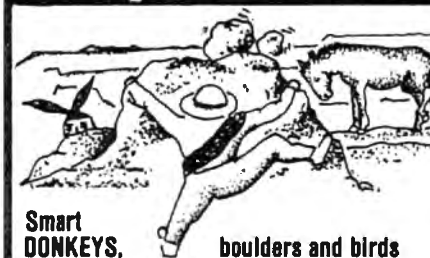
CLOWNS AND BALLOONS . . . an action-packed arcade game that'll make you want to join the circus.

16K DISK or TAPE $26.90

HARDWARE

800 Computer 16K	$648
800 Computer 48K	$757
400 Computer 16K	$288
810 Disk Drive	$448
850 Interface	$178
410 Recorder	$ 78
MX80FT+ Printer	$588
16K Ram module	$ 69
32K Ram module	$109
Graphics Tablet	$269
10 Blank Disks	$ 28
Track Ball Controller	$ 63
Percom Disk Drive	$648

Canyon Climber



Smart DONKEYS, boulders and birds dropping rocks try to stop you from reaching the top. A real challenge. Arcade quality.

16K

DISK $26.90 TAPE $22.40

SOFTWARE

PAC MAN (cart.)	$39.50
Centipede (cart.)	$39.50
Microsoft Basic (D) 32K	$79.10
Ghost Hunter (T) 16K	$25.50
Ali Baba (D) 32K	$28.90
Rear Guard (T) 16K	$17.90
Track Attack (D) 48K	$26.90
Threshold (D) 40K	$33.90
GORF (D) 16K	$35.10
Crypts of Terror (D,T) 16K	$26.30
Choplifter (D)	$31.40
APX-Text Editor (D) 32K	$39.95
Cactus League Baseball (D)40K	$17.90
De Re Atari Book	$17.90
Atari Games Book	$13.90
Megalegs (D, T) 16K	$31.40
Book Keeping Package (D) 40K	$67.50
Eastern Front (T,D)	$25.40
Bug Attack (T,D) 32K	$26.90
Pacific Coast Hwy (T,D) 16K	$26.90
Alien Swarm (T) 16K	$26.90
Midnight Magic (D) 48K	$31.45
Intruder (T) 16K	$26.90
Nautilus (T,D)	$20.20

Descend through mazes into the Dungeons with your squadron of warriors to battle with visible and invisible monsters.

Wizard of Wor

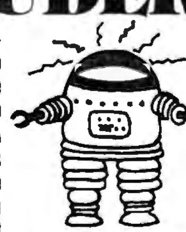
16K DISK ONLY $35.10

CROSSFIRE

The aliens have landed and it's your job to save the city. (If you can!) TOP RATE GAME, ARCADE QUALITY, HI-RES Graphics & Sound.

16K TAPE
32K DISK
$26.90

INTRUDER



A secret research installation is protected by two robo-units whose one purpose in life is to destroy all intruders. A major earthquake has struck this region. The "hot room" is a shambles! A meltdown at the installation is imminent! Your job is to guide a repair robot into this area and avoid being caught by the mad robo-units.

16K
$26.90 TAPE
$31.40 DISK

Call or Write for

FREE

ATARI PRODUCTS CATALOG
with hundreds of items

FROGGER

This is the genuine FROGGER game that you see in the arcades. Made by the same people that made Jawbreaker (One of the top ten sellers.)

16K DISK OR TAPE $31.40

VISA · master charge

TO ORDER CALL TOLL FREE 800-452-8013

FOR INFORMATION CALL 503-683-5361

HOW TO ORDER: Send check or money order or call our toll free number and use your Visa Card. Shipping on software is $2.00 per order anywhere in USA. Hardware shipping call for cost. Add 3% for VISA or MC. Equipment subject to price change and availability without notice.

Store Hours
8 am — 6 pm
Mon. — Sat.

Royal Software
2160 W. 11th Ave.
Eugene, Oregon 97402

- "Analogies" – This course describes the common types of analogies and provides practice in analyzing and solving analogy problems.
- "Number Series" – This course teaches students how to analyze number series patterns and provides practice in number series problems.
- "Quantitative Comparisons" – This course reviews mathematics, from elementary arithmetic through algebra and plane geometry, and provides practice in solving the types of mathematics problems found on standardized tests.

With the exception of "Number Series," which is self-testing, there is a test at the end of each course.

Preparing for the SAT is available for use on Atari 400/800 computers with a memory of at least 16K. It retails for $125.00 and is available from computer stores and by mail from PDI (shipping and handling charge extra). Five of the package's courses – Vocabulary Builders I and II, Analogies, Number Series, and Quantitative Comparisons – are also available individually.

Program Design, Inc.
11 Idar Court
Greenwich, CT 06830
(203)661-8799

Hayes Stack Smartmodem 1200

Hayes Microcomputer Products, Inc. introduces the Hayes Stack Smartmodem 1200, a Bell 212A compatible modem that lets RS-232C compatible computers or terminals communicate over telephone lines at 1200 bps.

The Smartmodem 1200 connects directly to the telephone line and an RS-232C port. It is approved by the FCC for direct connection to any U.S. telephone system for both pulse and Touch-Tone dialing. Both types of dialing may be combined in a single command with pulse used, for example, to access a PBX board, and Touch-Tone used to dial an outside number.

The Smartmodem 1200 is actually two modems in one; it operates at either 0-300 bps or 1200 bps. It is an intelligent system that executes user commands and responds with either decimal digit or English word result codes. The modem can be controlled by any programming language, and it includes all circuitry for auto-dial and auto-answer, thereby eliminating the need for auxiliary equipment and making the Smartmodem 1200 a stand-alone system.

Via an audio monitor, users hear the progress of a call and are alerted to wrong numbers and busy signals. A simple repeat command causes the Smartmodem 1200 to automatically redial a number. Indicator lights on the modem's front panel allow a visual check of its operational status.

Power-on default options are controlled by the position of eight switches, four of which can be overridden by software commands. Options include full or half duplex, enable auto-answer, and result code type. Unique "Set" commands allow selection and change of additional parameters such as dialing speed, escape code character, and number of rings before the modem answers a call.

The Smartmodem 1200 has a two year limited warranty. Included with the Smartmodem 1200 unit are: a power pack, one modular telephone cable to connect the unit to the telephone, and an owner's manual. Suggested retail price is $699.00.

The design permits other Hayes components to be stacked on top of the Smartmodem 1200.

Hayes Microcomputer Products, Inc.
5835 Peachtree Corners East
Norcross, GA 30092
(404)449-8791

The Cosmic Balance For Apple And Atari

Strategic Simulations Inc. has introduced The Cosmic Balance, one of the first four games in their new Rapidfire line.

Each player assumes the dual role of commander and architect of a starship fleet. For a ship of given size, the various parameters and trade-offs include: engine (for power), drives (for speed and maneuverability), weapons (such as phasors, disruptor bolts, plasma torpedoes, seeker missiles, and unmanned fighter swarms), defense belts, shields, transporters, and space marines.

There is a crew efficiency factor which affects the ability of a ship in battle. Ships with greater hull space have more room for recreation, allowing for greater crew morale and efficiency.

There is no best ship in this game – only one designed with optimum compromises to suit a player's style of warfare.

A typical game can be played in as little as ten minutes. With either the two-player or solitaire scenario, the semi-automated graphics and rapid execution of ship-to-ship combat are features you'll use.

The Cosmic Balance sells for $39.95. The game includes a diskette, rule book, and player aid cards. It is designed for the 48K Atari 400/800 computers and for the 48K Apple II Plus, Apple III, or Apple II with

Applesoft ROM card.

Strategic Simulations Inc.
465 Fairchild Drive, Suite 108
Mountain View, CA 94043
(415)964-1353

Commodore Bulletin Board

The Commodore Bulletin Board System has been introduced to turn a user's PET or CBM into a host communications system capable of sending and receiving messages, programs, SEQ and WordPro files.

Developed by Steve Punter, author of WordPro, the program has several other features, including user Log and daily usage Log, formatted messages, optional private messages and protected programs, optional password sign-on, a bulletin section, complete editing functions, and more.

The system may be used in a variety of applications, such as a program distribution service for business or education, a general message service for community computer users, or a private system for clientele or employee communications.

The package comes with sample bulletins and HELP files, remote terminal programs, and a 28-page manual that includes schematics for a simple telephone answering device.

Commodore Business Machines Ltd.
3370 Pharmacy Ave.
Agincourt, Ontario M1W 2K4
(416)499-4292

Two VIC Games From ComputerMat

ComputerMat has announced two new arcade games for the VIC-20 called Alien Invasion and Snakeout. They run on the standard VIC-20 and include color, graphics and sound.

Alien Invasion has 20 different levels of play and lets you move your laser into position to defend against the invader forces descending from space. Each invader has a different point value, and bonus points are scored if you destroy the invading forces in the correct order. An invader game with a new twist.

Snakeout provides a slithering snake that moves around the screen to munch blocks for points that appear at random. The snake's tail keeps getting longer and will get you in the end. Snakeout can be played by one player, or two if you desire. The tape also includes the game Trapper as a bonus.

Each cassette is priced at $9.95; all orders should include

$1.00 per order for shipping.

ComputerMat
Box 1664 Dept. P
Lake Havasu City, AZ 86403

12" Green Screen Monitor From Leading Edge

The new 12" green screen monitor has a suggested retail of $99.00. The 13 pound CRT features a composite video signal and a display format of 1920 characters (80 char. x 24 lines). The dimensions are 40.0 cm (w) x 28.5 cm (h) x 32.0 cm (d).

Leading Edge has a one year, no-questions-asked return/replacement plan should the unit fail.

Leading Edge Products, Inc.
225 Turnpike Street
Canton, MA 02021
(800)343-6833

EPYX Game On ROM Cartridge For The Atari

Alien Garden, a new strategy-action game, has been released by EPYX for the Atari 400/800.

As a hungry alien, the player is challenged to survive in a world thriving with enticing crystal life forms. Some crystals are edible, but others explode, killing the player. To gain the most points, the player must decide whether to sting, avoid, or eat each type of crystal. The faster he does, the more rapidly his score increases.

A wealth of crystals – up to 19 unique species – complicate the player's survival. Nine skill levels add to the challenge.

Alien Garden combines joystick action with arcade-style full color graphics and sound. The player hears the shattering and tinkling of crystals as they grow, shrink, and explode before his eyes.

Alien Garden is available on ROM cartridge for the Atari 400/800 for $39.95 from:

EPYX
P.O. Box 4247
Mountain View, CA 94040

Conferences Scheduled For Educators

EdCOM '82 – the National Computer Conference and Expo for Educators – will be held October 21-24, 1982 at the L.A. Convention Center, Los Angeles, California.

EdCOM '82 will feature over 200 seminars, workshops, demonstrations, exhibits, and hundreds of computers for in-depth tutorials and hands-on sessions.

Presentation topics designed for educators at all levels of expertise will include computer aided instruction, administrative uses, classroom management, programming, research applications, authoring languages, literacy, and more. All of these sessions will be conducted by nationally recognized professionals in the field of computer education.

For more information contact:

Jayne LaFountain
EdCOM '82
2629 N. Scottsdale Road
Scottsdale, AZ 85257

****

The Minnesota Educational Computing Consortium (MECC), involved in the implementation of instructional computing on a local, regional, and statewide basis, is sponsoring a conference (MECC '82) to share a decade of experience serving educators. The conference, scheduled for December 1 and 2, 1982 in Minneapolis, will be dedicated to practical sessions directed at educators involved in promoting the use of computers in schools. Sessions will cover in-service training techniques, software and hardware evaluations, classroom teaching strategies and activities, K-12 curricula planning, and promotional events, including computing contests for teachers and students. Banquet sessions will feature addresses by Steve Jobs, Chairman of the Board of Apple Computer, Inc. and Roger Badeschar, President of Atari, Inc.

In addition to the main conference, one day pre- and post-conference workshops will be offered on Using Computers in the Classroom, Advanced Microcomputer Programming, In-service Training Techniques, and Courseware Development. All conference activities and workshops will provide participants with the ideas, instruction, and materials to enhance instructional computing in their home institutions. For more information and registration forms, write:

MECC '82
2520 Broadway Drive
St. Paul, MN 55113

The International Software Directory Goes On-line

The International Software Database is now available on-line through Lockheed Dialog. As is the hardcopy, it is fully searchable by machine, operating system, subject, vendor and price. And now, with this on-line service through Lockheed, it has full text searching by key words in any or all fields. This feature helps you find the piece of software you need to run on the system you have and within the budget allowed.

Lockheed Dialog is an international information retrieval service making the software database available to its subscribers. This database is updated

COMPUTER PL

NECESSITIES FOR APPLE AND ATARI OWNERS

APPLE DISK		
The ultimate nibble copier		
NIBBLES AWAY II	~~49.95~~	59.95
The ultimate copy utility		
SUPER DISK COPY	~~30.00~~	24.95
The ultimate mailing list		
MR. LISTER	~~139.95~~	69.95

BMC 12" COLOR MONITOR	$289.95
ATARI 400 16K	$289.95

ATARI (c)assette, (d)isk, (b)ook			
programming secrets			
TRICKY TUTORIALS #1-6ea	~~19.95~~	(c)	14.95
invitation to			
PROGRAMMING #1	~~24.95~~	(c)	19.95
invitation to			
PROGRAMMING #2 or #3 ea	~~29.95~~	(c)	20.95
First Book of Atari	~~12.95~~	(b)	10.95
De Re Atari	~~19.95~~	(b)	17.95
Technical Ref Notes	~~29.95~~	(b)	20.95
Inside Atari Dos	~~19.95~~	(b)	16.95
all six tricky tutorials			
In DELUXE BINDER	~~99.95~~	(c)	74.95
incredible			
MICROSOFT BASIC	~~89.95~~	(d)	65.95
incredible			
MACRO ASSEMBLER	~~89.95~~	(d)	65.95

COMPUTER PLACE
P.O.Box 6007-169 1698 Market St
Redding, CA 96001 Ph: 916-221-1312

We are a retail computer store in the downtown shopping mall. We have the largest selection of computer items you are ever likely to see. Stop in and see us. Mail order terms: Cashier's check or money order. Charge cards add 4%. Add 3% shipping and handling (minimum $2.50)-UPS ground. No C.O.D.'s. CA residents add sales tax. Prices subject to change without notice.

MICROMAIL

CRT'S • PRINTERS • TELEPRINTERS

★NEW★ DEC LA100
Dual Mode Matrix Printer
CALL FOR PRICE!
DEC LA12D
"LA 120 Replacement"
$1599.00

★LOW★ DIABLO
630RO – RS232C
$1899.00
630KSR .... CALL
630 API .... NEW
$1649.00
620-RO .... NEW
$1149.00

★NEW★ 3M WHISPER WRITER
• *TWX/TELEX*
• *TIME SHARE*
• *Computer Terminal*

ANADEX
9500/9501
$1125.00

CALL FOR OUR LOW PRICES ON

• TELEVIDEO • DIABLO • DEC
• ANADEX • QUME • T.I.
• TELETYPE • C.ITOH • 3M

TOLL FREE (800) 854-6028

SAVE ON PREPAID ORDERS

MASTERCARD VISA ACCEPTED

C.O.D. REQUIRES 15% DEPOSIT

FOR ORDERING INFORMATION PLEASE CALL

MICROMAIL
P.O. Box 3297
Santa Ana, CA 92703
Phone: 714/731-4338
TWX: 910 595 1146

Software Specials 25% DISCOUNT

ADVENTURE INTERNATIONAL–		Your Cost
Preppie	Cass	18.70
	Disk	22.45
BRODERBUND–		
Choplifter	Disk	26.20
Track Attack	32K Disk	22.45
David's Midnight Magic	Disk	26.20
Stellar Shuttle	Cass or Disk	22.45
Apple Panic	48K Disk	22.45
Star Blazer	Disk	23.95
DATASOFT–		
Text Wizard	32K Disk	74.95
Pacific Coast Hwy.	Cass or Disk	22.45
Shooting Arcade	Cass or Disk	22.45
Clowns & Balloons	16K Cass or Disk	22.45
Canyon Climber	16K Cass	18.70
	16K Disk	22.45
LJK–		
Letter Perfect	24K Disk	112.45
ON-LINE SYSTEMS–		
Frogger	Cass or Disk	26.20
Ultima I	Disk	29.95
Ultima II	Disk	44.95
Crossfire	Cass or Disk	22.45
Jawbreaker	Cass or Disk	22.45

ATARI
A Warner Communications Company

ATARI 800

16K	$639
32K	$718
48K	$797

ATARI 400

16K	$269

410 Program Recorder	$79
810 Disk Drive	$439
830 Modem	$155
850 Interface	$165
16K Ram Module	$79
Intec 32K Ram Board	$95
Intec 48K Ram Board	$195
COMMODORE–	
VIC 20	$235
Datasette	$65
Disk Drive	$489
Printer	$335
Epson MX80/MX100	CALL
Citoh Prowriter	$485

K-BYTE–		Your Cost
K-Razy Shootout	Cart	37.45
K-Razy Kritters	Cart	37.45
K-Razy Patrol	Cart	37.45
K-Razy Antiks	Cart	37.45
ROKLAN–		
Gorf	16K Disk	29.95
Wizard of Wor	16K Disk	29.95
SIRIUS–		
Cyclod	Disk	22.45
Snake Byte	48K Disk	22.45
SYNAPSE–		
File Manager 800	40K Disk	74.95
Disk Manager	32K Disk	22.45
Nautilus	32K Cass or Disk	22.45
Slime	16K Cass or Disk	22.45
Protecter	32K Cass or Disk	22.45
Chicken	16K Cass or Disk	22.45
PRISM–		
Galactic Chase	16K Cass	18.70
	16K Disk	22.45

ORDERING INFORMATION: We accept VISA/MASTERCARD, Cashier Check, Money Order, or Personal Check *(Allow 14 days to clear)*. California residents add 6% Sales Tax.

SHIPPING INFORMATION: We ship All Orders UPS. ADD $2.00 for software orders of any amount. Hardware orders add 2%.

CALL or WRITE for FREE CATALOG. All Atari Software at a 20% Discount.

SPORT 'N' SOUND ELECTRONICS

21999 Van Buren Street
Grand Terrace, CA 92324
(714) 783-0556

continuously, thus keeping subscribers up-to-date on new software packages.

Imprint Editions
1520 South College
Fort Collins, CO 80524
(303)482-5574

Professional Software For Commodore Machines

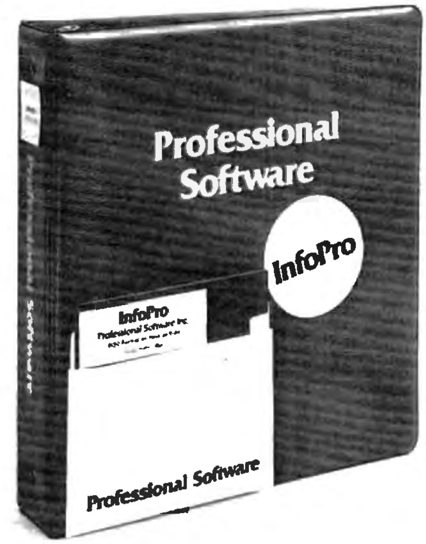
Professional Software has introduced four new software packages:

– *WordPro-ML* (Multi-Lingual) is designed specifically for the Commodore 8000 series computer. It gives the user access to word processing in five languages – English, German, French, Spanish, and Italian. The user can also switch back and forth between languages without losing the text in memory.

– *WordPro-Mail List* is designed for mailing list applications on Commodore computers and for use in conjunction with WordPro software. It includes a mini WordPro text editor and permits automatic entry, without retyping, of mailing lists already stored on disk. Suggested retail price is $195.

– *InfoPro*, for Commodore 8000 and 9000 series computers, has a Super Scan feature for rapid search of stored data for specific information. It prints user-defined reports with specific formats and can create personalized letters or documents. Suggested retail price is $295.

– *The Administrator*, for the Commodore 8032, links "master" records to "transactional" records and tracks transactional history.



It includes a mailing list label print option and a Report Generator. Suggested retail price is $595.

Professional Software, Inc.
51 Fremont Street
Needham, MA 02194
(617)444-5224

Computer Programming Contest

The American Heart Association, Greater Boston Division, is taking a new approach to heart health education – an electronic approach. The American Heart Association, in conjunction with *Classroom Computer News*, has announced a "Heart Health Computer Programming Contest" to solicit the development of programs on health education for eventual national distribution to schools.

Programs should be aimed at an elementary, junior high, or high school audience, and should deal with ways in which heart disease can be prevented. The Surgeon General of the United States has recommended prevention of heart disease as a national priority, and the Heart Association has started its campaign to educate youngsters about the importance of good habits early

in life.

The best program entry has the potential to receive $1500.00; 11 other entries may receive $500.00. Programs are to be written in BASIC language to run on the Apple, Atari, TRS-80, Texas Instruments or the Commodore PET Microcomputers. All entries must be received by January 31, 1983. A panel of computer and education experts will act as judges and will reach a decision by Spring 1983.

For further information, write before December 1, 1982 to:

Heart Health Computer Programming Contest
American Heart Association
Greater Boston Division
33 Fourth Avenue
Needham, MA 02194

Apple Software For Children

Three software programs for use in the home by children three to 13 have been introduced by The Learning Company.

The new, interactive programs – Bumble Games, Bumble Plot, and Juggles' Rainbow – were developed for use on Apple personal computers. The learning programs use color graphics and music to encourage participation by children and parents.

The new programs are additions to The Learning Company's "Building Blocks" software. The building block structure of the programs guides children through progressively more difficult learning games.

Bumble Games is a set of six colorful number games guided by Bumble, a creature from the planet Furrin. These games help children from four to ten learn how to describe locations on a map, to understand charts and graphs, and to do basic computer graphics. Children also learn basic math skills from the program. Its U.S. price is $60.

Bumble Plot also features the Bumble character and is a set of five games using skills learned in Bumble Games. It expands the child's learning experience by using negative and positive numbers. Children from eight to 13 are asked to identify and pair numbers, read a sonar map, and draw on the screen using the game's computer graphics. The U.S. price is $60.

Juggles' Rainbow introduces children from three to six to prereading skills and gives them a basic understanding of computers. Children learn the concepts of above/below and left/right as they generate patterns with playful, colorful graphics and music. They also work with lines and circles that help them identify the letters *P, D, B* and *Q*, the most difficult in the alphabet. The U.S. price is $45.

* * * * * * *

The first two offerings in a family of "Discovery Tool" microcomputer software programs for use by children from six to 13 have also been introduced by The Learning Company. The two programs – Gertrude's Puzzles and Rocky's Boots – were developed for use on Apple personal computers.

"Discovery Tool" software features an animated graphics editor that allows children to create animated objects and move through many connecting rooms via the computer screen. The proprietary software provides continuous, interactive feedback and enables players to create new shapes and colors, draw new characters, and even devise whole new games.

The programs stress logical thinking and help expand a child's potential for creative thinking. The programs can be enjoyed by those as young as four and are challenging even to adults.

Gertrude's Puzzles features Gertrude the Go-Getter Goose, a labyrinth of rooms, and objects of various shapes and colors. By moving Puzzle Pieces players learn to form color and shape patterns. The progressively challenging puzzles involve thinking about the colors and shapes of the pieces on the screen. To begin each puzzle, Gertrude flies off and returns with a new set of shapes. The games help children develop reasoning skills and learn to solve problems with incomplete information by analyzing what they see.

Rocky's Boots challenges players to build point-scoring logic machines in an arcade game setting. Players build machines with conventional logic symbols for *and, or, not* and *flip-flop*. While building, they learn about the basics of computer circuits. The object of the game is to build the most efficient machines.

The U.S. price of each program is $75.

The Learning Company
4370 Alpine Road
Portola Valley, CA 94025
(415)851-3160

Disk Based Data Manager

MicroSpec Ltd. has released its disk Data Manager for the Commodore VIC-20. It is a comprehensive data management system that allows users to define and manage their own data base and record entries on diskette. The system's menu includes:

– *Create a File* builds the data base and creates a description file on the disk.

– *Add a Record* allows entry of records to the disk.

– *Delete a Record* deletes records from the file and makes the record's space available to the user.

– *Change a Record* allows the user to change or edit any previously entered record.

– *Browse Through the File* allows

the user to scan or sequentially review the records in the data base.

– *Search the File* allows the user to search the data base and display specific records on the screen according to user criteria.

– *Print the File* allows users to print a hardcopy report according to their own specifications. The order, limit, number of fields, field position, and optional totals by numeric fields may be specified.

– *Exit* saves the binary record key map and returns to BASIC.

The system will store up to 1200 records on a single disk. Any record can be retrieved and displayed directly without having to read any intervening records. The system requires a minimum of 8K free memory to operate. The Disk Data Manager System sells for $59.95, complete with documentation and a storage binder. MicroSpec Ltd. also offers the system on any Commodore CBM or PET, the new Commodore 64, the Atari 800, the Apple II, and the IBM Personal Computer.

MicroSpec Ltd.
2905 Ports O'Call Court
Plano, TX 75075
(214)867-1333

Mathematics Review For Standardized Tests

Program Design has released *Quantitative Comparisons* for Atari 400 and 800 computers. This course prepares students for the type of math problems commonly found on the Scholastic Aptitude Test (SAT) and on other aptitude and intelligence tests. It reviews the principles that form the basis of mathematics from beginning arithmetic through elementary algebra and plane geometry.

Many of the mathematics questions found on standardized tests ask test-takers to compare two quantities or mathematical expressions, and indicate which of the two is greater. There are four possible answers:

1. Quantity A is larger than Quantity B
2. Quantity B is larger than Quantity A
3. Quantity A equals Quantity B
4. Which quantity is larger cannot be determined from the information given

The questions in *Quantitative Comparisons* are presented in this format. The course includes seven lessons and a final test. Each lesson tests a person's knowledge in one or more mathematical areas:

Lesson 1 – Numbers and Arithmetic
Lesson 2 – Roots and Exponents
Lesson 3 – Fractions
Lesson 4 – Decimals and Percent
Lesson 5 – Angles and Plane Geometry
Lesson 6 – Algebra
Lesson 7 – Graphs and Units of Measurement

The course includes a booklet that contains explanations of the various types of problems. For example, here is a typical problem from *Quantitative Comparisons:*

Which is bigger?

A	*B*
6.5%	*65/100*

1. A is larger than B
1. B is larger than A
3. A equals B
4. Can't determine

The answer is 2: B is bigger than A. If the student doesn't understand the problem, he or she is told to turn to Explanations 46 and 47 in the booklet, which state "A percent is a fraction with a denominator of 100" and "To write a fraction as a percent, change the fraction to a decimal.

Move the decimal point two places to the right. Add the percent symbol." Each statement is followed by mathematical equations illustrating the principle.

For every problem in the course, the number of the relevant explanation will appear on the TV monitor if the person fails in three attempts to correctly answer the problem. In Lessons 2 through 7, he or she can also learn the number of the relevant explanation by requesting a "hint."

Quantitative Comparisons is for use on Atari 400/800 computers with a memory of at least 16K. It retails for $19.95 (cassette) or $26.50 (disk) and is available from computer stores and by mail from PDI (shipping and handling charge extra).

Program Design, Inc.
11 Idar Court
Greenwich, CT 06830
(203)661-8799

IDS Introduces Microprism Printer

The new Microprism dot-matrix printer from Integral Data Systems, Inc. offers print quality approaching that of a daisy-wheel printer at a lower price than has previously been available. It provides higher quality output

without extra printer features.

The high speed and graphics capabilities of a dot-matrix printer have been combined with the print quality approaching that of a daisy-wheel. IDS calls this printing technique Maisey. Maisey describes a dot-matrix printer which can print correspondence-quality text in a single pass.

Standard Microprism printer features include: Dotplot graphics, serial RS232 and Centronics-compatible parallel interface, automatic line buffering with 1400 bytes standard, 75 cps in correspondence-quality and 110 cps in high-speed data mode, and Maisey (dot-matrix printing of correspondence-quality text in a single pass). Microprism is priced at $799.

Integral Data Systems, Inc.
Milford, NH 03055
(603)673-9100 / (800)258-1386

Sterling Swift Introduces DIAL-A-COMMAND

The Apple II DIAL-A-COMMAND, a two-sided com-

ATARI

ATARI

800 16K $639
800 48K $789
400 16K $279

410 Recorder	$ 75
810 Disk Drive	$439
820 Printer	$259
822 Printer	$259
825 Printer	$579
830 Modem	$155
850 Interface	$169
481 Entertainer	$ 79
482 Educator	$125
483 Programmer	$ 55
484 Communicator	$299
853 16K Ram	$ 79

ATARI Software

CS406 Financial Management	$ 55
CX4104 Mailing List	$ 19
CX404 Word Processor	$115
CXL4007 Music Composer	$ 45
Programming 2 & 3	$ 22
Conversational Languages	$ 45
CX4018 Pilot	$ 59
CX405 Pilot	$ 99
CXL4003 Assembler Editor	$ 45
CX8126 Microsoft Basic	$ 67
CX412 Dow Jones Investment	$ 95
CXL4022 Pac-Man (new)	$ 33
CX8130 Caverns of Mars	$ 29
CXL4020 Centipede (new)	$ 33
CXL4006 Super Breakout	$ 28
CXL4008 Space Invaders	$ 28
CXL4009 Computer Chess	$ 28
CXL4010 3-D Tic Tac Toe	$ 26
CXL4011 Star Raiders	$ 35
CXL4012 Missile Command	$ 28
CXL4013 Asteroids	$ 28

New Software for Atari

Frogger	$26
Apple Panic	$23
Raster Blaster	$23
Temple of Apshai	$30
Bishops Square	$23
Graphics Master	$30
Tumble Bugs	$23
Action Quest	$23
Crossfire	$23
Threshold	$30
Computer Stocks & Bonds	$17
Guns of Fort Defiance	$17
Mousekattack	$26
K-Razy Shootout	$36
Jawbreaker	$23
Preppie	$19
Rear Guard	$19
Arcade Pro Football	$26
Bug Attack	$23
Pathfinder	$26
Deadline	$39
Zork I	$29
Zork II	$29
Crypts of Terror	$26
Pool 1.5	$26
Deluxe Invaders	$26
Gorf	$29
Wizard of Wor	$29
Battle Trek	$29
Canyon Climber	$23
Shooting Arcade	$23
Pacific Coast Highway	$23
Clowns and Balloons	$23

For Fast Delivery, send certified or cashier checks, money orders, or direct bank wire transfers. Personal checks allow 2 to 3 weeks to clear. 25% deposit on C.O.D.'s. Prices reflect a cash discount only and are subject to change. Shipping—Software ($2.00 Minimum), Hardware—call. Foreign inquiries invited — add 15% for shipping. Nevada residents add sales tax.

TOP SELLERS
Atari Software

Entertainment

Ghost Hunter	$ 26
Lunar Lander	$ 12
Reversi	$ 23
Gomoku	$ 23
Star Warrior	$ 29
Crush, Crumble, & Chomp	$ 23
Ricochet	$ 15
Empire of the Overmind	$ 26
Tanktics	$ 23
B-1 Nuclear bomber	$ 13
Kayos	$ 26
Match Racers	$ 23
Wiz & Princess	$ 24
Mission: Asteroid	$ 19
Softporn Adventure	$ 23
Ali Baba & the Forth Thieves	$ 24
The Shattered Alliance	$ 29
Galactic Chase	$ 23
Atari World	$ 45

Business & Utilities

Visicalc	$179
Mail Merge	$ 23
Data Perfect	$ 79
Letter Perfect	$105
Text Wizard	$ 69
Datasm 65 2.0	$105
Micropainter	$ 26
The Basic Compiler	$ 69
Color Print	$ 29

Educational

Compu-Read	$ 23
Compu-Math/Fractions	$ 29
Compu-Math/Decimals	$ 29
Vocabulary I	$ 16
Vocabulary II	$ 16
Number Series	$ 16
Analogies	$ 16
Story Builder/Word Master	$ 16
Let's Spell	$ 16
Astro Quotes	$ 16
All APX Software	15% off list

*** SPECIALS OF THE MONTH ***

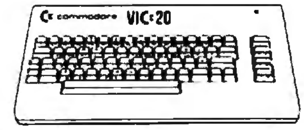
ELEPHANT DISKS (BOX)	**$ 22**
HAYES SMARMODEN	**$229**
AXLON RAMCRAM	**$129**
MOSAIC 32K RAM	**$ 99**
RAMDISK (128K)	**$429**
MICROTEK 32K RAM	**$109**
AMDEK COLOR I MONITOR	**$329**
PERCOM DOUBLE DENSITY DRIVE	**$679**
EPSON MX-100 PRINTER	**$729**
OKIDATA MICROLINE 80	**$329**
K-DOS	**$ 65**
OS/A +	**$ 59**
BASIC A +	**$ 59**
FLIP N' SORT DISKETTE BOX (Holds 50 Diskettes)	**$ 21**
FLIP-SORT CARTRIDGE BOX (Holds 10 Atari Computer Cartridges)	**$ 21**

YOUR ONE STOP MARKETPLACE FOR ALL YOUR COMPUTER NEEDS
If it is not listed, please ask.

Computer Outlet

Park Place — Upper Level
1095 E. Twain — (702) 796-0296
Las Vegas, Nevada 89109

Call Toll Free **800-634-6766**
We accept Major Credit Cards
Mon.-Sat. 8 A.M.-6 P.M.



commodore

VIC 20 . . . . $249

VIC1530 Datasette	$ 67
VIC1540 Disk Drive	479
VIC1515 Graphics Printer	329
VIC1210 3K Memory Expander	$ 30
VIC1110 8K Memory Expander	$ 52
VIC1011 RS 232C Terminal Interface	$ 43
VIC1112 VIC-1EEE-488 Interface	$ 84
VIC1211 VIC 20 Super Expander	$ 52
VIC1212 Programmers Aid Cartridge	$ 45
VIC1213 Machine Language Monitor	$ 45
VM110 Vic Programmers Ref. Guide	$15

VIC Software

Avenger	$23
Superslot	$23
Super Alien	$23
Jupiter Lander	$23
Draw Poker	$23
Midnight Drive	$23
Spiders of Mars	$39
Meteor Run	$39
Amok	$29
Alien Blitz	$29
Renaissance	$39
Outworld	$29
Cloudburst	$39
Satellites & Meteorites	$39
Skymath	$12
Space Division	$12
Long Division	$12
Super Hangman	$15
3-D Maze	$12
Raceway	$13
Kosmic Kamikaze	$21
The Alien	$21
Simon	$10
Kiddie Pak I	$32
Super Four I	$39
Kiddie Checkers	$ 8
Snakman	$18
Astrobase-2001	$13
Space Barrier	$13
Snake	$10

(Programming Techniques)

Print Commands	$14
For-Next Loops	$14
Graphics	$14
Types of Variables	$14
Data Files	$14
Random Numbers	$14

Educational

Money Addition	$10
Math Whiz	$10
State Capitals	$10
World Capitals	$10
Spelling	$ 8
The Verb	$14
The Adverb	$14
The Adjective	$14
Fraction Reduction	$10
Adding Signed Numbers	$10
Plurals	$ 8
Memory	$10

★ ATARI ★
Programming Techniques
(Santa Cruz — Tricky Tutorials)

Display Lists	$ 17
Horiz/Vert Scroll	$ 17
Page Flipping	$ 17
Basics of Animation	$ 17
Player Missile Graphics	$ 24
Sound	$ 17

The Computer Outlet is an associate of The Computer Learning Center For Children. We are experts in educational technology and can customize educational software curriculums for school districts, individual schools, or for the child at home. Please contact us about your software and equipment requirements and feel free to stop by our school in Las Vegas.

We have one of the world's largest educational software inventories featuring our own Computer Learning Center software.

Ten Little Robots (ATARI)	$12.95
Pre-School Math (ATARI)	$19.95

mand reference tool, has been introduced by Sterling Swift Publishing Company.

The DIAL-A-COMMAND weighs less than one ounce, exactly fits the hand, and "speaks" commands an Apple II or II Plus understands. The user can select from 166 functions and thereby automatically dial the proper command to be entered for the following operations: Editing and Format, Input/Output, Flow of Control, Sequential Text File, System and Utility, Graphics and Game Controls, Machine Language Text File, Access, Housekeeping, Math, and Arrays and Strings.

Also listed on the DIAL-A-COMMAND are keystroke entries for Algebraic Operators, Relational and Logical Operators, and Simple Variables.

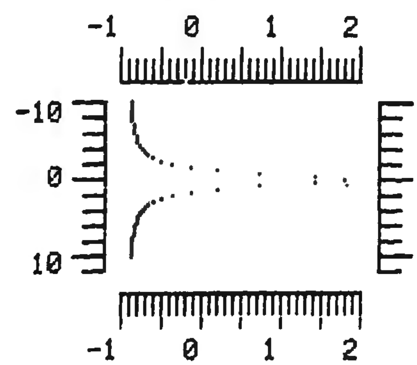
The DIAL-A-COMMAND, soon available from dealers, may now be ordered directly from Sterling Swift. Price is $9.95.

Sterling Swift Publishing Company
1600 Fortview Road
Austin, TX 78704

Plotting Routine For VIC

Improved High Resolution Plotting Routine turns the 1515 Printer with a VIC-20 into a high resolution plotting machine. Resolution is 72 dots/inch horizontally and 63 vertically (the highest possible on the 1515). It plots the results of user supplied programs, functions, or data logging routines (including from the keyboard). Size of the plot and automatic scaling with number labels and tic marks are continuously user adjustable. Scales are provided on all four sides. The length of the plot is unlimited, allowing strip chart recording (the end scale is generated after the plot is finished). It requires only about 2.7K of RAM (the rest of the memory you have, including expanders, is available for your point generating program). Written in BASIC, it is designed to run on any VIC system or the new Commodore-64.

Y=2.7/(1+X↑2)-.9



Includes instructions and demonstration program. The price is $10.00.

Scientivic Software
Dept. N, 525 Lohnes Dr.
Fairborn, OH 45324

Conference Highlights Effective Use Of Small Computers

Effective use of small computers in a variety of functions will be the theme of the National Conference on "Using Small Computers: Implementing Integrated Information Systems," to be held at the Hyatt Regency Crystal City, Arlington, Virginia, on October 18-20, 1982.

Emphasis of the three-day meeting will be on effective use of small computers in areas such as office automation, decision support systems, word processing, training, and graphics. The conference will also focus on strategies that provide reliable, effective, and economical information systems, and increase the quality and effectiveness of software for various corporate functions.

Two full days will be devoted to presentations by experienced users and experts on such topics as application planning and implementation strategies; selecting effective small computer systems; using small computers in networks; and integrating communications, video, and small computers. Three preconference workshops will provide opportunities for in-depth studies on decision support systems; networks and small computers; and selecting software for small computers.

A special feature of the conference will be an exhibit of hardware and software for small computers. In addition, time has been set aside for interaction between attendees and successful users.

Infosystems, the Information Systems Magazine for Management, is sponsoring this conference, to draw together user information on small computers and assist those who will be faced with making MIS decisions in the near future.

For further information, contact the Conference Manager:

U.S. Professional Development Institute
12611 Davan Drive
Silver Spring, MD 20904
(301)622-5696

Letter Quality Printer From Olympia

Olympia USA Inc., a marketer of Olympia office equipment, has introduced a new letter quality printer at COMDEX.

Designated the Olympia ESW 3000, this higher speed,

extremely reliable, cost-effective printer was developed for the midrange letter quality printer market and extends Olympia's current line of lower speed printers. Word processing and office information systems are only two uses for this new printer.

For use with keyboard-oriented displays, the ESW 3000 incorporates an Olympia print wheel with 100 characters. Standard lettering in many popular typestyles, proportional spacing, four pitches, bi-directional printing, and print wheel cassette insertion for operator convenience are all features of the new ESW 3000. Print speed is 35 characters per second. Tractor and sheet feed options are available.

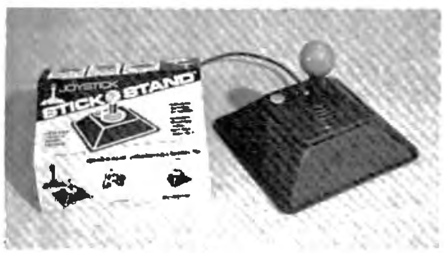
Laser-sharp print quality, a choice of ribbon systems, and economical pricing make the ESW 3000 attractive to both end users and OEMs.

The single unit price of the ESW 3000 is under $2,000.

Olympia USA Inc.
Box 22
Somerville, NJ 08876
(201)722-7000

Joystick Stick Stand

K-Byte's new Stick Stand with the Fastball Easy-Grip Control Knob was designed to reduce the player's hand and wrist fatigue. It also frees one hand for a more skillful operation of the fire button.



This stand allows players to increase their physical dexterity and achieve higher scores. By just snapping the Fastball onto the Joystick, and then snapping the Joystick into the Stick Stand, the player is all set.

For additional information on Stick Stand (Model No. ATH1000), contact:

K-Byte
P.O. Box 456
1705 Austin
Troy, MI 48099
(313)524-9878

New Product releases are selected from submissions for reasons of timeliness, available space, and general interest to our readers. We regret that we are unable to select all new product submissions for publication. Readers should be aware that we present here some edited version of material submitted by vendors and are unable to vouch for its accuracy at time of publication.

Advertisers Index

COMPUTE! Publications

COMPUTE!

For Fastest Service,
Call Our **Toll-Free**
US Order Line
800-334-0868
In NC call 919-275-9809

My Computer Is:

☐ PET ☐ Apple ☐ Atari ☐ OSI ☐ VIC-20 ☐ TI 99/4A ☐ Sinclair ZX-81
☐ Radio Shack Color Computer ☐ Other ________ ☐ Don't yet have one...

☐ $20.00 One Year US Subscription
☐ $36.00 Two Year US Subscription
☐ $54.00 Three Year US Subscription

(Readers outside of the US, please see our foreign readers subscription card or inquire for rates)

Name ______________________
Address ______________________
City ________ State ________ Zip ________

☐ Payment Enclosed ☐ VISA
☐ MasterCard ☐ American Express
Account No. ______________ Expires ____/____

: 10 11 12

COMPUTE!

Foreign Readers

Subscription rates outside the US:

☐ $25.00 Canada FI=2
☐ $38.00 Europe/Air Delivery FI=3
☐ $48.00 Middle East, North Africa, Central America/Air Mail FI=5
☐ $88.00 South America, South Africa, Australasia/Air Mail FI=7
☐ $25.00 International Surface Mail (lengthy, unreliable delivery) FI=4,6,8

Name ______________________
Address ______________________
City ________ Postal Code ________
Country ______________________

Payment must accompany this card.
Payment in US Funds drawn on a US Bank; International Money Order; or charge card: ☐ VISA ☐ MasterCard ☐ American Express
Account No. ______________ Expires ____/____

9 10 11 12

COMPUTE! Books

Quan.	Title	Price		S/H	Total
____	**The Beginner's Guide to Buying A Personal Computer**	$ 3.95	+	$1.00*	____
____	**COMPUTE!'s First Book of Atari**	12.95	+	2.00*	____
____	**Inside Atari DOS**	19.95	+	2.00*	____
____	**COMPUTE!'s First Book of PET/CBM**	12.95	+	2.00*	____
____	**Programming the PET/CBM**	24.95	+	3.00**	____
____	**Every Kid's First Book of Robots and Computers**	4.95	+	1.00*	____
____	**COMPUTE!'s First Book of VIC**	12.95	+	2.00*	____

*$4.00 / **$9.00 For air mail outside US

For Fastest Service
Call Our **TOLL FREE**
US Order Line
800-334-0868
In NC call 919-275-9809

All orders must be prepaid (money order, check, or charge). All payments must be in US funds. NC residents add 4% sales tax.

☐ Payment enclosed

Please charge my:
☐ VISA ☐ MC ☐ Am. Express
Acc't No. ______________
Expires ____/____

Name ______________________
Address ______________________
City ________ State ________ Zip ________
Country ______________________
Allow 4-5 weeks for delivery.

9 10 11 12

The Editor's Feedback:

Computer: ☐ Pet ☐ Apple ☐ Atari ☐ OSI ☐ VIC-20 ☐ TI 99/4A ☐ Sinclair ZX-81
☐ Radio Shack Color Computer ☐ Other ________ ☐ Don't yet have one...

Are you a **COMPUTE!** Subscriber? ☐ Yes ☐ No I would like to see:

☐ More ☐ Fewer Specific applications programs
☐ More ☐ Fewer BASIC programs.
☐ More ☐ Fewer Machine language programs.
☐ More ☐ Fewer Tutorials
☐ More ☐ Fewer Educational articles
☐ More ☐ Fewer Detailed explanations of programs.

☐ More ☐ Fewer Games.
☐ More ☐ Fewer Reviews of game software.
☐ More ☐ Fewer Reviews of business software.
☐ More ☐ Fewer Reviews of educational software.
☐ More ☐ Fewer Reviews of hardware.

What do you like best about **COMPUTE!**?

What do you like least?

9 10 11 12

Place
Stamp
Here

COMPUTE! Magazine
Post Office Box 5406
Greensboro, NC 27403

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 2312 GREENSBORO, NC

POSTAGE WILL BE PAID BY ADDRESSEE

COMPUTE! Magazine
P.O. Box 5406
Greensboro, NC 27403

Place
Stamp
Here

COMPUTE! Magazine
Post Office Box 5406
Greensboro, NC 27403

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

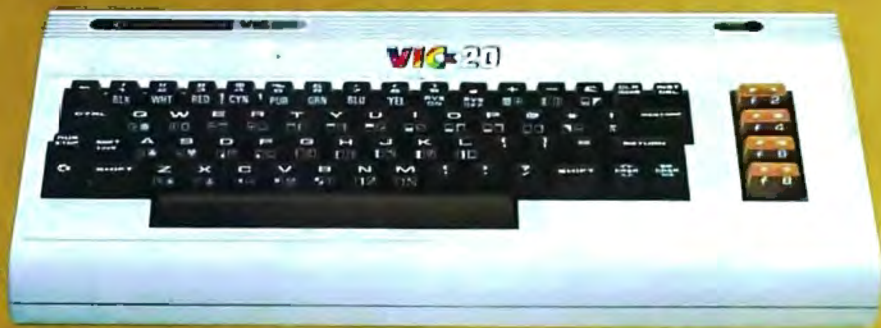
BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 2312 GREENSBORO, NC

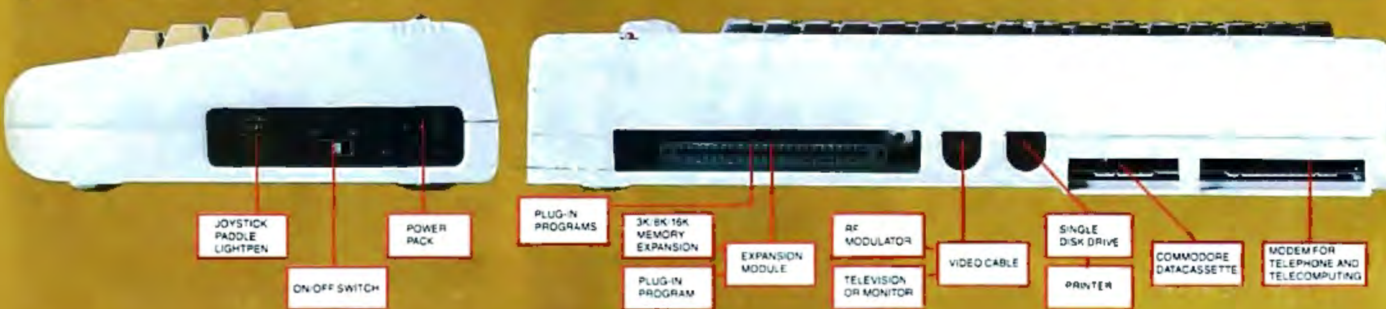
POSTAGE WILL BE PAID BY ADDRESSEE

COMPUTE! Books
Post Office Box 5406
Greensboro, NC 27403

COMMODORE VIC-20®
"THE WONDER COMPUTER OF THE 1980s. UNDER $300."
—WILLIAM SHATNER

"The best computer value in the world today. The only computer you'll need for years to come."





VIC-20® VS. OTHER HOME COMPUTERS

Product Features	Commodore VIC-20	Atari® 400™	TI® 99/4A	TRS-80® Color Computer
Price*	$299.95	$399.00	$454.00	$399.00
Maximum RAM Memory	32K	16K	48K	32K
Keyboard Style	Full-Size Typewriter Style	Flat Plastic Membrane	Full-Size Typewriter Style	Calculator Style
Number of Keys	66	57	48	53
Programmable Function Keys	4	0	0	0
Graphic Symbols On Keyboard	62	0	0	0
Displayable Characters	512	256	192	256
Microprocessor	6502	6502	TMS9900	6809
Accessible Machine Language	YES	YES	YES	YES
Upper/Lower Case Characters	YES	YES	YES	NO
Operates with all Peripherals (Disk, Printer and Modem)	YES	NO	YES	YES
Full Screen Editor	YES	YES	YES	NO
Microsoft Basic	Standard	N/A	N/A	$ 99.00
Telephone Modem	$110.00	$399.95	$450.00	$154.95

*Manufacturer's suggested retail price [illegible]

Read the chart and see why COMPUTE! Magazine[1] calls the VIC-20 computer "an astounding machine for the price." Why BYTE[2] raves: "...the VIC-20 computer unit is unexcelled as a low-cost consumer computer." Why Popular Mechanics[3] says "...for the price of around $300, it's the only game in town that is more than just a game." And why ON COMPUTING INC.[4] exclaims: "What is inside is an electronic marvel... if it sounds as if I'm in love with my new possession, I am."

The wonder computer of the 1980s. The VIC-20 from Commodore, world's leading manufacturer of a full range of desktop computers. See the VIC-20 at your local Commodore dealer and selected department stores.

1 April '81 issue   2 May '81 issue   3 November '81 issue   4 Fall '81 issue

VIC-20
commodore COMPUTER

Commodore Consumer Products Division
487 Devon Park Drive, Wayne, PA 19087
Canadian Residents: Commodore Computer Systems
3370 Pharmacy Ave., Agincourt, Ont., Canada, M1W 2K4

Please send me more information on the VIC-20.

Name ______________________
Address ______________________
City ____________ State ________ Zip ______
Phone ______________________