IRA Planner: A Financial Planning Tool For Radio Shack Color Computer, Atari, Apple, PET/CBM, VIC, And OSI

COMPUTE!

$2.50
July
1982
Issue 26
Vol. 4, No. 7
63379 £1.85 in UK

The Journal For Progressive Computing™

Gold Rush
Action Game For Atari And VIC

Atari Video Graphics: An In-depth Tutorial

An Apple DOS Changer: Protect Your Disks From Accidental Erasure

Super QuadraPET
Independently Partitioning PET/CBM Memory

Whither VIC: An Overview Of The VIC-20 Computer

Reviews: Caverns of Mars for Atari, Two VIC-20 Games, Softbox-CP/M for PET/CBM, and more

07
0 74470 63379

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.

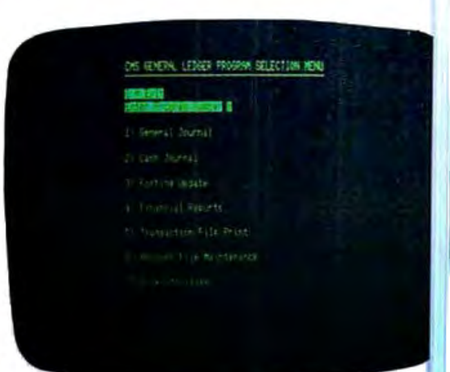
Professional Software Inc.
51 Fremont Street
Needham, MA 02194
(617) 444-5224
TELEX: 95 1579

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

CMS Softwa
General Accou

For the Commodore 8000

Now there is an interactive General Accounting System designed especially for the first time user. The user is led through each function by highlighted prompts explaining the required input at each point. And all input requests are prompted with complete verification. Plus, a step-by-step instruction manual guides the user through the program. The user prompts and the detailed manual make it virtually impossible for the first time user to get lost or to accidentally crash.



General Ledger

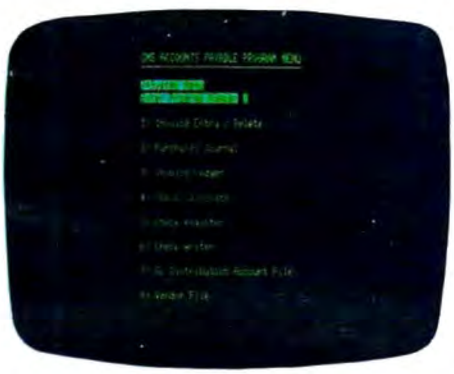
Cash Disbursements and Cash Receipts Journal as well as a General Journal. Maintains account balances for Present Month, Quarter to Date and Year to Date. Accepts postings from Accounts Receivable, Accounts Payable, Payroll and other programs. Also provides user customized financial statements. Up to 1000 accounts and up to 99 departments.

re Systems
nting System II

Series Computer System

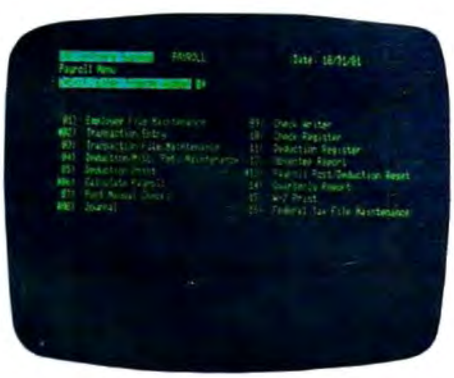
Accounts Receivable

Invoices and Monthly Statements as well as Credit and Debit Memos. Accomodates full or partial payments. User sets finance charge rate and period. User also sets invoice aging reports and aging breaks. Invoices may be distributed among nine different General Ledger accounts with automatic updating to the General Ledger. During invoice data entry the invoice is displayed on the screen and typed exactly as if in a typewriter. File capacity allows 1800 customers and invoices. Five hundred customers allows room for 1300 invoices.



Accounts Payable

Accounts Payable checks with full voucher detail. Detailed check register. Automatic application of Credit Memos. User sets invoice aging reports and aging breaks. Invoices may be distributed among nine different General Ledger accounts with automatic updating to the General Ledger. File capacity allows 2500 vendors and invoices.



Payroll

Payroll checks with full deductions and pay detail. Pays regular, overtime, holiday and piece work hours. Accommodates Weekly, Bi-Weekly, Semi-Monthly and Monthly employees. Up to eight miscellaneous deductions or payments per employee. Prints Payroll Journal, Payroll Check Register and an Absentee Report as well as 941 information and W2 forms. Maintains Monthly, Quarterly and Yearly totals for up to 350 employees. Automatic General Ledger updating.

See your nearest Commodore dealer for a demonstration.

CMS Software Systems

2204 Camp David   Mesquite, TX. 75149   214 285-3581

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.

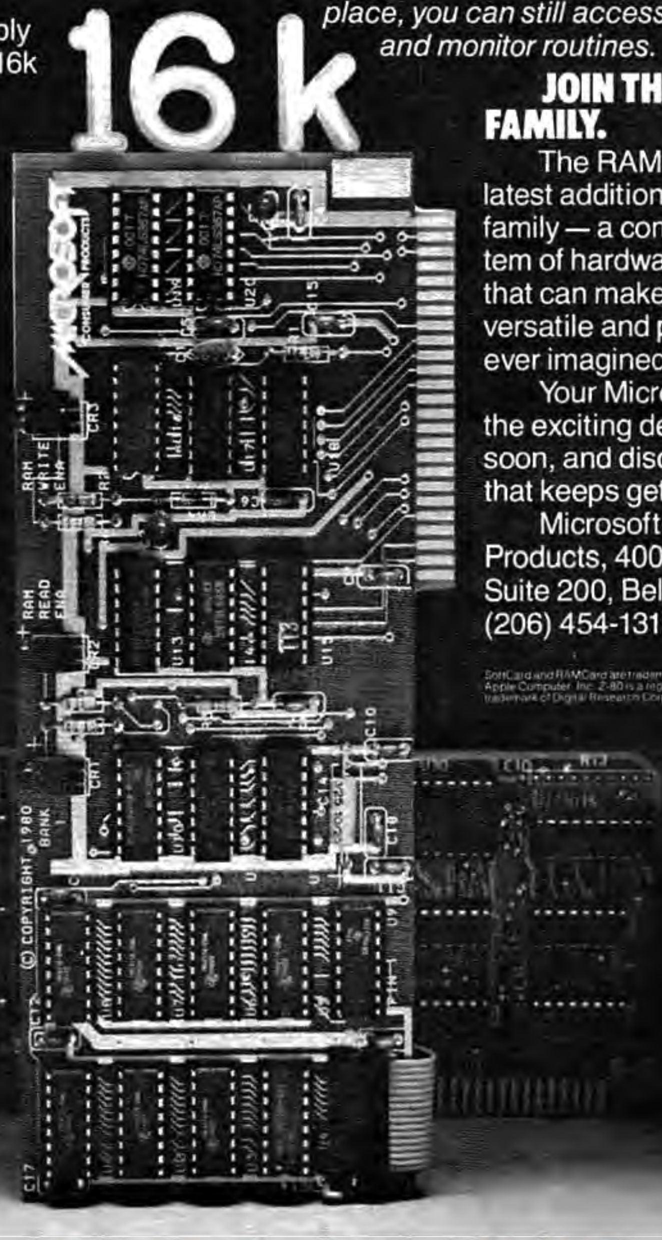
Microsoft's new RAMCard simply plugs into your Apple II,® and adds 16k bytes of dependable, buffered read/write storage.

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

July, 1982 Vol. 4, No. 7

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, 515 Abbott Drive, Broomall, PA 19008. Second class postage paid at Greensboro, NC 27403 and additional mailing offices.

ISSN 0194-357X.

A Small System Services, Inc. Publication

COMPUTE!

The Editor's notes...

A Brief Overview Of The Chicago Consumer Electronics Show And National Computer Conference In Houston

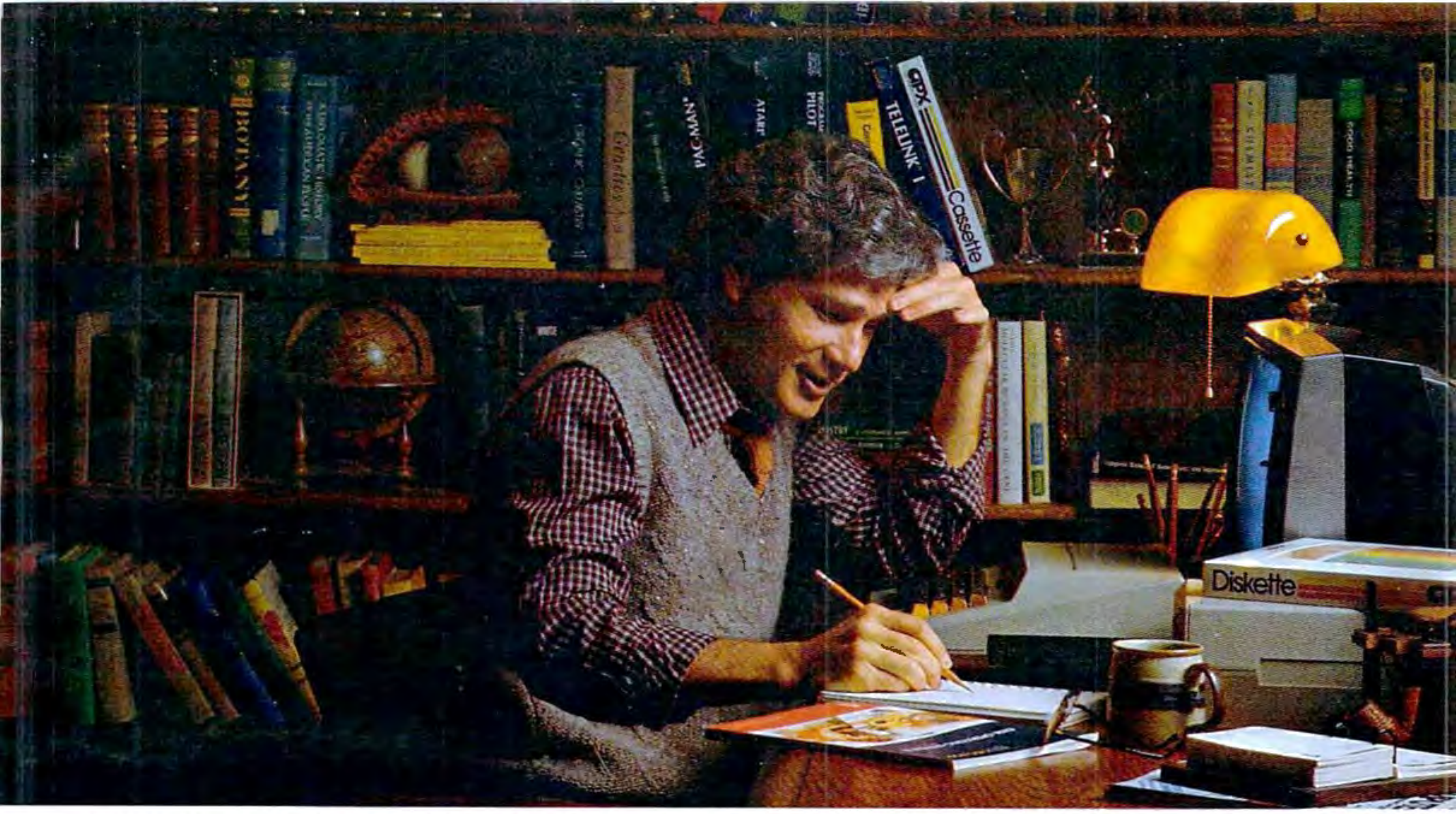
Robert Lock, Publisher/Editor and Tom Halfhill, Features Editor

I don't know why they ended up holding these two significant shows on overlapping days, but it made for some extremely hectic traveling. Tom Halfhill, our new Features Editor, will describe the shows and new products in much more detail next issue (both shows are still in progress), but we thought you'd enjoy a tantalizing preview.

Commodore's Third Generation

In this partial list of features you'll be able to discern the impact of the Commodore introductions. Tom will fill in the blanks next month:

	BX256	B128	"P" Series	64	MAX
Integral Monitor	80X25 Green Phosphor	80X25	40X25 External color (not included)	40X25	40X25
Dual Disks optional cassette	yes	yes	optional	optional	
Standard RAM	256K	128K	128K	64K	unknown
Colors	N/A	N/A	16	16	unknown
Graphics Resolution	N/A	N/A	320X200	320X200	unknown
CP/M Compatible	yes	optional	optional	optional	N/A
Interfaces: IEEE-488	yes	yes	yes	accepts VIC peripherals CBM peripherals optional	N/A
RS-232	Yes	Yes	Yes	unknown	N/A
Standard Language	BASIC 4.0	BASIC 4.0	BASIC 4.0	A BASIC Version Described As Compatible In Most Ways With 4.0	BASIC Cartridge Available
Planned Retail:	$2995	$1695	$995	$595	$179.95
Scheduled Release:	fall	fall	fall	July/August	



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 program 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, 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 uses 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—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 can allow you to create a dialogue with the computer. And a single "match" command can perform complex text evaluation and pattern-matching instantly.

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 program languages from ATARI in the future.

In addition to bringing the computer age home, ATARI is committed to answering the needs of computer professionals.

ATARI®

*Available from the ATARI Program Exchange

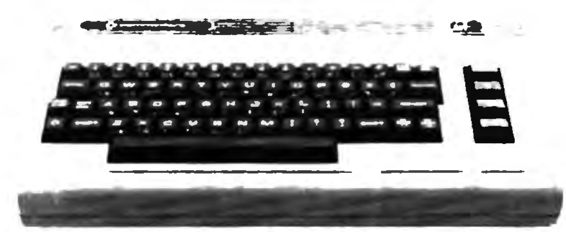
ATARI HOME COMPUTERS

We've Brought The Computer Age Home.™

Commodore "MAX" Machine

Third Generation
Commodore P128 Series Microcomputer

Third Generation
Commodore "B" Series
Microcomputer



Commodore 64

Other New Computers: Sinclair Introduces A Color Computer

The Sinclair Spectrum was introduced in the US at the CES. It's roughly the same size as the ZX-81, but includes a partial stroke keyboard rather than the familiar membrane keyboard. The unit features extended Sinclair BASIC, 16 colors, 16K RAM standard, and a suggested retail in the $200 price range. More next issue.

Epson, the well-known printer manufacturer, introduced a 16K portable computer with a suggested retail of $795. The computer is truly portable, deriving power from four NiCad batteries. Built-in (standard) features include a LCD (Liquid Crystal Display) screen with 20 character by four line display, and a 24 column dot matrix printer. The entire unit measures roughly 11 inches by 8.5 inches by 2 inches, and weighs 3 pounds, 13 ounces. The unit also features a built-in RS-232C communications port with full/half duplex selection and 110 to 4800 baud rate. A standard microcassette unit is optional.

Epson HX-20

Other Random Bits In The "More Later" Category...

Atari, Apple and Commodore were showing off new software. Atari announced a price reduction on the 400's suggested list to $349.95. Microsoft reports increased and substantial support for MS-DOS. TIMEX was actively promoting the $99.00 TIMEX/Sinclair computer.

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

Subscription Information

COMPUTE! Circulation Dept.
515 Abbott Drive
Broomall, PA 19008 USA

TOLL FREE
Subscription Order Line
800-345-8112
In PA 800-662-2444

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

Robert C. Lock, President
W. Jerry Day, Vice-President, Comptroller
Kathleen Martinek, Assistant To The President
Ellen Day, Accounting Manager
Carol Holmquist Lock, Research Assistant

Advertising Sales

In British Columbia, Canada call 408-354-5553

Elsewhere in Canada or outide 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
P.O. Box 7000-462
Redondo Beach, CA 90277
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 August Issue Of COMPUTE!

Keyprint Compendium: PET/CBM

A Color Computer Energy Monitor

Electric Eraser For The VIC

Video Graphics And The GTIA: Atari, Part II

An Apple Data Manager

A Color Computer Energy Monitor


JINSAM™ 8.2
Officially approved by Commodore
For Their 8000-9000 Series Desk-Top Computers
USA
space
age
micro
software
Jinsam 8.2
OFFICIALLY APPROVED BY COMMODORE
has broken the sort limit. Sorts are now unlimited – you may sort to any desired depth: 3,4,5,6,7 ....
We now have:
Unlimited record length, unlimited fields, and unlimited sorts, searching by key quick-scan and pattern matching, allowing limitless applications.
Used at NASA,
Kennedy Space Center
With Multiple Applications Related
to the Columbia Space Shuttle Project
including cue operations
statistical ports, inventory and
vehicle tr cking.
JINI MICRO-SYSTEMS, Inc.
DATABASE MANAGEMENT SYSTEM DESIGN
Box 274 KINGSBRIDGE STN.
RIVERDALE, N.Y. 10463
(212)796-6200

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.*

A Bigger Set Of Commands

I keep reading about this new POKE, and that new POKE, to do all sorts of wonderful things. But it seems cumbersome to me. As micros evolve, might not some scheme evolve so these memory addresses do not need to be memorized?

In other words, a much larger command set. The new commands are directly translated internally to the proper POKE, so it wouldn't require that much more memory. And it would be both easier to remember *and* more transportable.

What do your readers think? Perhaps some of them already have done such a program!

Kevin Sinclair

Language designers face a tough decision: they must balance convenience (many commands) versus the amount of ROM or RAM granted to them (such as 8K BASIC).

Fortunately, many languages can be extended, permitting you to add extra, custom commands. If your language is "soft-loaded," coming off disk or tape into RAM, you can usually "patch into" (modify) the language to force it to recognize your new commands. However, the new commands will generally have to be written in machine language and reside in RAM somewhere.

ROM-based languages are more difficult to alter. Most Microsoft BASICs (Apple, OSI, Commodore) put a fragment of the language into RAM during their start-up sequences when you turn power on. You can put a machine language "JMP" command (a "wedge") in the midst of this fragment which causes the computer to regularly "jump" to a machine language routine of your own which would check to see if you had typed "SPECIAL" or whatever instead of a normal BASIC word.

The "Universal Wedge" for disk commands, Basic Aid, and other BASIC enhancement programs use this method. Non-Microsoft BASICs, such as Atari 8K BASIC, can also be made to recognize new commands, but it is a tricky business. Watch upcoming issues of **COMPUTE!** *for an Atari Wedge routine.*

*For further information about adding "Wedges," see Bill Wilkinson's "Insight: Atari" column (***COMPUTE!***, May, 1982 #24), "Modifying Apple's Floating Point BASIC" (***COMPUTE!***, May, 1982, #24) and, for Microsoft BASIC in general, "The Wonderful Wedge" (***COMPUTE!***, April, 1981, #11).*

Atari Lockup Revisited

Several readers have responded to the issue raised by Greg Kopp in Ask the Readers, May, 1982, about the occasional "lockup" where the Atari will, mysteriously, "go away" and no longer respond to the keyboard.

Bill Wilkinson, **COMPUTE!** *columnist and one of the authors of Atari BASIC, responded that "all substantial software has bugs. If it's in ROM, the bug can't be repaired unless a new set of ROM chips are brought out. For whatever reason, Atari has never brought out a new ROM set. To be fair about it, there are bugs in the original TRS-80 which have never been fixed either."*

We might add that the new IBM computer has a bug in the division routine. Also, the several fixes which Commodore has made to its BASIC in ROM have cured the original bugs, but brought about new problems of software compatibility between versions.

BASIC To BASIC

I'm having trouble with the PET program on page 14 in Issue 23, April, 1982. I couldn't get it to work. Now that I have looked at lines 140 and 150, I'm confused. Isn't [the variable] "TI" always zero? If so...doesn't this cause line 150 just to loop? At least that's where my program gets stuck.

Guy Lillis

The program you refer to is a "fast-find" for cassettes, but is designed to work only on Commodore cassette drives (PET/CBM/VIC). Programs which use only the keyboard or screen can usually work (with some changes) on most computers using BASIC. However, techniques involving peripherals *like tape drives tend not to be very "portable" between computer brands.*

For example, the Commodore machines have a special variable called TI (for TIme*) which is an internal clock. This variable is always being updated by the computer and can, therefore, be used to time events. In your computer, TI is an ordinary variable, always zero unless you assign a value to it.*

The "fast-find" technique is so specific to the machinery involved that it even needs adaptations to work on the different models of the Commodore line. (See the following letter.)

VIC Fast-Find

I need your help with a small problem. I entered the BASIC code for the cassette tape to locate programs on fast-forward as shown on Page 14 of the April, 1982, **COMPUTE!**, and, although you claim that it "can be adapted to any Commodore

A dynamic duo from SSI for the ATARI®!

THE BATTLE OF SHILOH: A brigade-level simulation of the first grand battle of the Civil War, pitting the Confederate Army against Grant's troops and Union gunboats.

TIGERS IN THE SNOW: Ghostlike Nazi Tiger tanks and infantry sweep across the dark, frozen forests of the Ardennes against a surprised U.S. force in this division/regiment-level simulation of Hitler's last desperate attack.

Atari owners, rejoice!

First, SSI brought you THE SHATTERED ALLIANCE™ and you made it the popular fantasy wargame it is today.

Now, we are proud to offer two more games for your pleasure: THE BATTLE OF SHILOH™ and TIGERS IN THE SNOW™.

In producing these games, we've taken full advantage of the Atari 400/800's brilliant graphics capability and combined it with our extensive wargame-design experience.

For example, both games have colorful map displays that can be scrolled effortlessly so you can keep track of all the action on the vast battlefields. We've also employed special rules and innovative gaming systems to enhance the sophistication, realism and playability of these simulations.

And the result of all this hard work? Two strategy games that will truly challenge and excite your game-loving mind!

At $39.95 each (available in either cassette or disc version*), they are waiting for you at your local computer/game store today!

VISA and M/C holders can order by calling **800-227-1617, ext. 335 (toll free)**. In California, call 800-772-3545, ext. 335.

To order by mail, send your check to: Strategic Simulations Inc, 465 Fairchild Drive, Suite 108, Mountain View, California 94043.

All our games carry a 14-day money-back guarantee.

*TIGERS: 40K mini floppy disc or 40K cassette for the Atari 400/800 with Atari BASIC cartridge. SHILOH: 40K mini floppy disc or 32K cassette for same. When ordering, please specify disc or cassette version.

SSI

STRATEGIC SIMULATIONS INC

As part of our demanding standards of excellence, we use maxell floppy discs.

machine," I can't get it to run on the VIC-20.

I suspect that the memory addresses in the PEEK and POKE statements for the Commodore PET are different than the VIC-20, but since I don't own the PET, I don't know how to convert these few memory addresses to the VIC-20.

Since there are only three BASIC statements that refer to memory addresses, could you inform me what the VIC-20 equivalent addresses would be.

I believe that **COMPUTE!** has a wide following among VIC-20 owners and that this data would be of interest to many readers.

William E. Bender

You're right, POKEs, PEEKs, and SYSs are the least "portable" aspect of BASIC. When you try to translate a program written for one computer model or brand, these words are sure to give you problems. The following version of tape Fast Find for VIC, by Associate Editor Harvey Herman, was originally published last fall. It makes for an efficient tape management system, but must be the first program on each tape.

```
100 REM VIC FAST FIND
110 REM
120 REM HARVEY B. HERMAN
130 REM
140 N=5:DIM A$(N):REM N IS # OF PROGRAMS ~
    ON TAPE
150 FOR I=1 TO N:READ A$(I):NEXT I
160 PRINT CHR$(147);CHR$(144);"  PROGRAM"
    ::PRINT:PRINT "NUMBER NAME":PRIN
    T
170 FOR I=1 TO N:PRINT I;"     ";A$(I):NEX
    T I:PRINT
180 INPUT "FIND NUMBER";J:PRINT
190 IF J<1 OR J>N THEN 160
200 IF J=1 THEN 330
210 REM START OF ROUTINE TO FAST FORWARD
220 REM WAIT FOR RELEASE IF NECESSARY
230 IF(PEEK(37151) AND 64)=0 THEN PRINT "
    PRESS STOP ON CASSETTE"
240 IF(PEEK(37151) AND 64)=0 THEN 240
250 PRINT "PRESS FAST FORWARD":PRINT
260 IF(PEEK(37151) AND 64)=64 THEN 260:RE
    M CHECK FOR PRESS
270 PRINT "OK":PRINT:A=TI
280 IF ABS(TI-A)<(J-1)*360 THEN 280:REM F
    AST FORWARD 6 SEC PER PROGRAM
290 POKE 37148,PEEK(37148) AND 247:REM ST
    OP MOTOR
300 PRINT "RELEASE FAST FORWARD"
310 IF(PEEK(37151) AND 64)=0 THEN 310:REM
     WAIT FOR RELEASE
320 REM DYNAMIC KEYBOARD LOAD
330 PRINT CHR$(147);CHR$(17);CHR$(17);CHR
    $(17);"LOAD ";CHR$(34);A$(J);CHR
    $(34);CHR$(19)
340 POKE 198,1:POKE 631,13:END
350 DATA PROGRAM 1,PROGRAM 2,PROGRAM 3,PR
    OGRAM 4,PROGRAM5
```

Hidden Adventure

Regarding Rudolph F. Lauer's query in the May issue: I am co-author of Fantasy Games Software's PET programs Swordquest and Escape From the Death Planet. We have both Original-ROM and Upgrade-ROM versions of these games.

Except for a few early orders, our cassettes include both versions. Users who have upgraded their PETs should try loading from the (unlabeled) flip-side of the cassette. If that side proves to be blank, they may send their cassettes to me with a stamped reply envelope, and I will record the Upgrade version for them at no charge.

J. L. Pietenpol

J. L. Pietenpol
5711 Trinity Rd.
Raleigh, NC 27607

GETting On Atari

I'm a new Atari owner and one thing bothers me much about Atari BASIC. In Apple BASIC and in PET/CBM BASIC, one keyword is GET, in terms of strings. Atari's keyword GET has a different meaning. How can I input characters without pressing return? It would be much easier to make advanced games without just using the joysticks.

Howie Fishman

To use Atari's GET command, you must first open a file to the keyboard, e.g. OPEN#1,4,0,"K:". You can then GET a keystroke with GET#1,A. A (or whatever variable you use) will contain the ASCII value of the key typed (A=69 for "E",97 FOR"a",etc.). GET will always wait for a keystroke.

You can use location 764 to see if a key has been pressed. It normally holds 255, but when a key is pressed, it will contain a "keyboard code" (not ASCII). POKEing 764 with a 255 will reset the register.

```
 100 IF PEEK(764)=255 THEN 100 }
 110 POKE 764,255              } Waits for
  10 OPEN #1,4,0,"K"           } a key
   .
   .
   .
100 GET#1,A            Gets a key
```

VIC Expansion Software

Why don't the manufacturers of software for the VIC-20 offer more sophisticated software requiring memory expansion? They do for all the other computers.

Scott Barber

The VIC is a new machine and memory expansion is even newer. Its library of software is building quickly, however, as the many ads in **COMPUTE!** *illustrate. For predictions and information about VIC software, see Jim Butterfield's "Whither VIC?" in this issue.*

An Attack Of Hearts

When I do a POKE 756,226, enabling lowercase and control characters to be printed on the screen in graphics mode 1 or 2, the computer fills the screen with a bunch of hearts. How do I get rid of them?

Larry L.

The alternate character set for Graphics modes one and two is the bottom half of the full character set, i.e., lowercase, graphics, and control symbols. Unfortunately, the heart (CHR$(0)), maps into the place where SPACE is encoded, resulting in a screen full of hearts. The usual cure is to perform a SETCOLOR 0,0,0, since the SPACE (or heart) defaults to color register zero. This will make the hearts black, the default background color. Naturally, you'll have to change this if the background is not black.

Unfortunately, this prevents you from printing any text in the blacked-out color, reducing your available colors from five to four. A superior, if more complicated, procedure would be to modify the character set by zeroing out the heart character. This requires additional RAM to hold the new character set and POKEing 756 with the pointer to the custom character set. Here is an example program:

```
100 CHBAS=756:CHORG=57344
110 CHSET=(PEEK(106)-8)*256
120 GRAPHICS 1
130 POKE 756,226
140 ? #6:? #6;"  now |charsinsl":? #6;"
 |CHARACTER| SET"
150 FOR I=0 TO 511:REM ONLY HALF NEEDS T
O BE TRANSFERRED
160 POKE CHSET+I,PEEK(CHORG+512+I)
170 NEXT I
180 POKE 756,CHSET/256
190 REM 'FOR W=1 TO 50:NEXT W' IS A PAUS
E USED ONLY FOR EMPHASIS
195 REM NOT NEEDED FOR WORKING PROGRAM
200 FOR I=0 TO 7:POKE CHSET+I,0:FOR W=1
TO 50:NEXT W:NEXT I:REM ERASE HEART
210 ? #6:? #6;"IdOlnE"
220 END
```

©

A Monthly Column

Computers And Society

David D. Thornburg
Associate Editor

Odds And Ends...

You folks sure love to write letters! My mail load from this column and Friends of the Turtle is large enough that I don't always get answers out right away. If you have written and haven't heard from me, don't despair – I will write as soon as I can.

I received several letters in response to the "computers and the arts" columns. Reader Ted Nemeth of New York wrote in part:

Dear Dr. Thornburg:

In "Computers and Society" in the March issue of **COMPUTE!** *you wrote "... Whitney saw a new medium."*

Mary Ellen Bute (my grandmother) was the first (absolutely) to use a light pen on an oscilloscope to create animated patterns with music. She worked with Professor Theraman creating this technique. Her work received major acclaim and showed at Radio City Music Hall in New York for years.

I have shown her Mr. Whitney's book and while she enjoyed many of the graphics, she felt his technique of visual-sound interpretation is not all that well developed. She worked with many composers from George Gershwin to Edgard Varese in developing her theories. Since her methods did not include the use of the digital computer, her work (about 25 short films) has been all but ignored. ...

As for myself, I am a student at NYIT. I own an Atari 800 and am working on programs similar to my Grandmother's, except with Rock and Roll music. ...

Sincerely,

Ted Nemeth

I found Ted's letter quite interesting. All too often we forget to look at the progress made with technologies that preceded the ones which currently hold our interest.

Several other readers shared their favorite graphics programs with me. Among the finest graphics packages I have seen for the Atari computers is the program Graphics Master written by Courtney Goodin (distributed by Datasoft, $49.95). This program allows the user to create pictures in either of two high resolution (320 x 192 pixel) screens. I use one screen for small graphic elements that I can then "pick up" and stand anywhere I wish on the second screen. The user can intermix text and graphics, and has several choices of text size. Symmetry operations, such as rotations and mirror reflections, can be performed as well. Since text is treated as part of the graphics screen, you can split a word in the middle of a letter if you wish. This fine program provides many features for its modest price. It is a tribute to its author that one can create superb drawings using only the Atari joystick as the input device.

I have just begun to explore a professional graphics package for the Apple – more on that in a later column.

The columns on computer games in the classroom also generated a lot of mail. Most of you who wrote agree with the idea that there is something to be learned from the arcade games. In the area of educational software, I have several new programs I am reviewing, including a beautiful set of programs from The Learning Company (formerly ALT) in Portola Valley, California. As with the graphics program for the Apple, these software reviews will have to wait for another column.

Computer Literacy – Who Needs It?

Those of you who read this column regularly know that I am a proponent of the use of computers in the classroom. There seem to be two extremes to the use of the classroom computer – as a tool for programming the child through rote drill and practice, or as an object of study and exploration. My own bias lies toward the use of intrinsically motivating learning games to reinforce basic skills that were taught using conventional (human teacher) methods. In addition, I am a strong proponent of the idea that children should gain mastery over the computer through the use of user friendly languages such as PILOT and LOGO.

Historically, my argument has been with those who see the computer used only as a replacement for or simple adjunct to a teacher – who see the computer as a data processing tool to probe the student's skills and report all failings and progress to a human for analysis and further action. I guess that the thought of row upon row of school children

THE MOSAIC RAM SYSTEMS FOR ATARI*

CLEARLY THE BEST

THE SCREEN CLARITY TEST

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.

*Trademark of Atari, Inc.

MOSAIC ELECTRONICS

P.O. Box 708 Oregon City, Oregon 97045
503/655-9574

sitting in front of sterile terminals performing the same tasks that could be performed better under human guidance and supervision is just not comfortable to me.

But now, it seems, there is even another issue to deal with. There appears to be growing concern that computer literacy is not an appropriate topic for our schools to teach. According to Eric Burtis, a member of the Palo Alto, California-based Ad Hoc Committee on Basic Skills Education, the education of children in basic skills is suffering from the use of computers in the classroom. The committee states that "The suggestion that every American citizen must become computer literate is fallacious and untrue, just as it would be untrue to state that every citizen must become an airline pilot if mass air transportation is to remain available to all mankind."

If the committee is of the opinion that our children are being taught computer skills at a level commensurate with an airline pilot's training, they certainly have not been paying attention to what is happening in the classroom. It is erroneous in the extreme to suggest that those of us who are computer literate are an isolated breed that provide services to the masses (as do airline pilots). The reality is that we can either control technology ourselves, or let others control it for us.

The digital computer promises to be the most significant technological development of the twentieth century (Ada, forgive me for leaving Babbage's machine out of the discussion). Today – just a few short years into the era of the personal computer, there is almost no one who fails to come in contact with computers, if only indirectly.

If computer literacy is not appropriate subject matter for our children, then what is? Is our educational system so sound and complete that it is impervious to further improvement? The function of schools is partly to prepare our children for survival in the real world. This requires the acquisition of skills in math, reading and writing, and a sense of historical perspective, etc. It includes the development of the whole child – the creative as well as the analytical mind – in order that each child is prepared to take an active participatory role in society.

Increasingly, ours is a society governed by the flow of information. And, increasingly, this information flow is mediated by computers. By not having "computer literacy" courses in the classroom, groups such as the Ad Hoc Commitee on Basic Skills Education are insuring that only children from certain economic and cultural backgrounds will have a chance to learn about this technology. To suggest that computer literacy be denied to all our children is elitism in the extreme. In the information age, those of us who are computer literate will be at a tremendous advantage over those of us who are not. If we believe the principle of equality of opportunity so central to our national identity, then we must have the foresight to realize that this equality can only occur when we provide equivalent educational opportunities to all of our children – regardless of their economic background. The schools are the most appropriate place to do this.

A Few Words About Innovision...

I own a company called Innovision. In the past, Innovision has made forays into the commercial marketplace with products such as the Presto-Digitizer tablet. It has also served as the umbrella company for my consulting practice, and as the haven from which I write books and articles.

The fact is that I enjoy writing a great deal, and, as an Associate Editor of **COMPUTE!** I felt a bit awkward also appearing as an occasional advertiser. While I have never done or said anything that, in my opinion, constituted a conflict of interest, it is essential that my readers know that no conflict could ever arise.

Accordingly, I have disbanded the commercial product lines of Innovision. I write, I occasionally consult, and I am involved with some new ventures run by other people. Innovision no longer has any products of its own.

While this was not an easy decision to make, it was aided by the increased time needed to answer your letters to Friends of the Turtle (I received 15 today alone). I have never failed to speak my mind from these pages, and I'm not planning on stopping now! Thank you all for your tremendous support.

Announcing the Printing Breakthrough of the Century: Smith-Corona® TP-1™ Text Printer

- **Low Cost Daisy Wheel Printer**

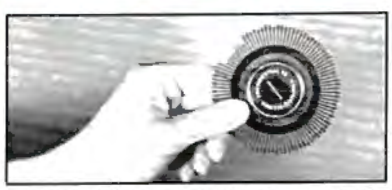
$799.00

- **Microprocessor Electronics**
- **Serial or Parallel Interface**
- **Simple, Reliable Mechanism**

ACT NOW: Limited Supply, Low, Low Cost

Smith Corona, one of the largest manufacturers of small printers in the world, gives a whole new perspective to printing with their electronic text printer—**TP-1.** The **TP-1** is a microprocessor controlled, high quality **daisy wheel printer.** It produces perfectly formed, executive quality printouts at the speed of 120 words per minute. Typewriter quality printing at dot matrix prices.

Simple, durable and dependable, **TP-1** may be used with word processing systems, microcomputers and most small business systems. Compact and attractively styled, the **TP-1** blends well with any setting.



Additional daisy print wheels $4.95

Additional ribbons $2.95

Now, all your letters, documents forms and reports can have the crisp, professional look you demand—for business or personal use—at an affordable price. **TP-1,** the electronic text printer.

Don't delay. Order your **TP-1** TODAY at the low price of **$799.**

Micro Printer Marketing offers same day shipping, nationwide service and invites dealer inquiries. Catalogues available. No shipping charges on pre-paid orders.

Micro™ Printer Marketing

Call Micro-Printer Marketing

CALL TOLL FREE **1-800-523-9859**

In PA Call Collect **215 / 433-3366**

MasterCard and Visa Accepted

If there's a topic that you would like to see covered in this column, send your suggestion to: The Beginner's Page, **COMPUTE!** *Magazine, P.O. Box 5406, Greensboro, NC, 27403.*

The Beginner's Page

Making Files Work

Richard Mansfield
Assistant Editor

Last month we examined *files*, often the most difficult aspect of programming for the new computerist. Here we'll conclude this overview by looking at some specific details about file handling.

In general, a file is a list or a collection of information which has been saved on a tape or a disk. A list of all the people you send Christmas cards to could be "memorized" on tape/disk by the computer and would then be a file.

How is this list of names and addresses "memorized" and how, at Christmas, is it used to print the addresses on envelopes? A file by itself is incapable of *doing* anything: it is not a program, it's just a list. A separate program is needed to create, update, and make use of files.

"Mailing Address," "File Manager," and others of this type are called data base management programs. Their primary function is to build, add to, modify, or print something out from files. They manage a collection (a *base*) of data.

You can, of course, write your own custom data base manager. This would amount to writing a program (or a set of programs) which would let you manipulate the list on tape or disk.

Writing a large data base management program is not an easy task – it can involve sorting, searching, and other complex programming techniques. Nonetheless, handling Christmas card lists is something the novice can accomplish and it's well worth learning. Files do, though, represent something of a challenge. Your computer's manual will contain information on the necessary punctuation and syntax for BASIC's commands which manipulate files. However, the following brief overview might be of help.

OPEN, PRINT#, INPUT#, And CLOSE

Where a program would be stored by the simple SAVE instruction, a file is stored by a combination of OPEN, PRINT#, and CLOSE. (On the Apple, PRINT is used in a special way instead of PRINT#. We'll get to that in a minute.) Likewise, a program is just LOADed, but a file is "loaded" into the computer with OPEN, INPUT#, and CLOSE.

It is a bit more complicated with files, but the bonus is that you can do more manipulating with files, easier *appending* (adding to them), easier *merging* (making two files into one), and so on.

The command OPEN is generally used to communicate with a disk or tape drive. It's like pulling open a file cabinet drawer – once a file is OPENed, you can then get at the records inside. Here's what you would do to OPEN a file named "inventory" on a disk drive attached to a Commodore, Atari, or Apple:

Commodore

```
10 OPEN 1,8,8,"0:INVENTORY,S,R"
```

The first number (1) means that this file will hereafter be called #1. When you pull something out of it, you would use INPUT#1 (you can have up to ten files OPEN at one time). The second number (8) means "disk drive" (a 1 in this position would mean: open a file on the cassette drive). The second eight is a "secondary address" which allows you to give additional instructions. With disk drives, just use eight.

The "0:" specifies drive zero and the "S" means *sequential* file. The Commodore disks can create other kinds of files: random, relative, and program files, but sequential is the simplest. Finally, the "R" means *read*. You will be using INPUT# to get things out of this file. (A "W" here would mean *write* and you would PRINT# to the file.) To make this "reading or writing" distinction for tape files, the secondary address is used: a one means write and a zero means read. (10 OPEN 1,1,0,"INVENTORY" would be the same as the example above, except it makes a cassette file.) No drive number is specified and the "S" is not necessary since cassette files can only be sequential files.)

Atari

The equivalent of the disk example above is similar in Atari BASIC:

```
10 OPEN#1,4,0,"D1:INVENTORY"
```

Here, the second number (4) stands for *read* (use 8 for write), the next number (0) is not used by the disk drive (but is necessary to satisfy the syntax of OPEN), and the "D1" specifies the first drive (there can be up to four drives attached, D1 through D4; D: means D1:).

Apple

To OPEN an Apple disk file, you first define a special character, D$, which holds the "control-D" character (you cannot see it). You would type:

```
10 D$="(hold down both CTRL and D keys)"
   (or you could use 10 D$=CHR$(4) instead)
15 PRINT D$;"OPEN INVENTORY"
```

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

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)

In our simple example, it isn't necessary to include drive and slot numbers with the OPEN command. Line 15, however, could indicate drive one, slot six if we add "OPEN INVENTORY, S6,D1" to the quoted information. The Apple uses the format in line 15 for its disk commands; first control-D is printed and then the desired action is described within the quotes.

Putting Something In And Taking Something Out

On the Commodore and Atari, you can put information into an OPENed file by using PRINT#. For Commodore files you could put the word "PENCILS" into the file by:

```
20 PRINT#1,"PENCILS"
```

and Atari would be:

```
20 PRINT#1;"PENCILS"
```

The Apple uses two lines:

```
20 PRINT D$;"WRITE INVENTORY"
21 PRINT "PENCILS"
```

(After line 20, the following PRINT command will be considered a print to the file, not to the screen).

Going the other way, you get something out of an OPENed file by using INPUT# in combination with a string variable to "hold" whatever comes from the file (they come back to the computer in the order they were PRINT#ed). To get the word "PENCILS" back from a Commodore file:

```
20 INPUT#1,A$
```

(Then later you could print A$ to see "PENCILS")

and Atari is:

```
20 INPUT#1,A$
```

(You must have previously DIMed A$).

Again, the Apple uses two lines:

```
20 PRINT D$;"READ INVENTORY"
21 INPUT A$
```

After you are finished INPUT#ing or PRINT#ing from a file which had been OPENed as file #1, you would CLOSE1 (PET/CBM) or CLOSE#1 (Atari) or PRINT D$;"CLOSE INVENTORY" (Apple). When you've finally CLOSEd, you are free to use that file number (#1 in these examples) for some other file, with a different name. CLOSE is essential, however. Without it you could permanently lose part or all of a file, or even damage other files.

INPUT# And PRINT# Hints

INPUT# or PRINT# work very similarly to the way INPUT and PRINT work from the keyboard and to the screen. The only catch is that PRINT# needs some special handling. It's best to give it a line all to itself:

```
20 PRINT#1,A$ (Commodore)
30 PRINT#1,B$
20 PRINT#1;A$ (Atari)
20 PRINT D$; "WRITE NAMEOFFILE" (Apple)
21 PRINT A$
```

The reason for putting PRINT# on its own line is that this is an easy way to separate items in a file: with "carriage returns." Just as 10 PRINT A$ / 20 PRINT B$ will cause B$ to be on the line below A$ on the screen (since using a new line "forces" a carriage return to take place) – a separate program line puts a carriage return symbol onto the tape or disk.

Manipulating Files

Files are usually created within loops. Here's a simple program to "write" a file to tape:

```
10 DATA BILL, SANDY, KATIE, LARRY
20 OPEN 1,1,1,"NAMES": REM (A PET/CBM TAPE
   FILE)
30 FOR I=1 TO 4
40 READ A$
50 PRINT#1,A$
60 NEXT I
70 CLOSE1
```

Since there are four names in this file, the loop counts up to four, READing a new A$ from the DATA line each time through. Since PRINT#1 is by itself on line 50, it will send carriage returns to the tape each time it PRINT#s, separating the names on tape. This way, there will be no question that a name should be BILLSANDY.

When this file is later read into the computer, it would be very useful to know when it ends, how big it is. There are two easy ways to do this. You could add the word "END" to the DATA line and then change line 30 to read: FOR I = 1 TO 5. Or, you could put the "count" (the number of records for this file) on the tape or disk itself, as part of the file. To do this, you would add a line: 25 PRINT#1,4.

Here's a "reader" program which brings the records from this second type of file back into the computer, finding out first what the count is:

```
10 OPEN1,1,0,"NAMES": REM (A PET/CBM TAPE
   FILE)
20 INPUT#1,COUNT:REM THIS WAS THE FIRST
   ITEM ON THE FILE
30 FOR I=1 TO COUNT
40 INPUT#1,A$
50 PRINT A$ (TO THE SCREEN)
60 NEXT I
70 CLOSE1
```

If you use the "END" technique, the reader program would not use line 20 and would add line: 45 IF A$ = "END" THEN GOTO 70. Notice also that you can PRINT# and INPUT# both numeric and string (alphabetical) variables.

One final note about something which might not be immediately obvious: if you update a file on

a PET/CBM computer, you cannot put it back on a disk with the same name. You might have read it off the disk and into memory to make some changes. But before you OPEN-PRINT#-CLOSE it back onto the disk, you must first "scratch" (remove) the *original* version in order to replace it with the updated one. Again, each computer has different formats for this and your manual will describe them. Some systems, the Atari for example, automatically replace files when one "comes into" the disk with the same name as one already on the disk. This *scratching* of unwanted files is unnecessary for tape files, they will write over the old file (if you rewind the tape).

There are numerous ways to manipulate files – this is precisely why it is a challenging programming task. The programmer has more control over what is happening, but more responsibility, too. As always, start out small by perhaps just creating a file with your name on it, reading it back into memory, and printing it on the screen. Then try ten names, a mailing list, updating records, and so on. Eventually you'll become adept at file manipulation and will find many uses for these valuable programming techniques. ©

This exciting action game is for the Atari and the VIC. Blast out new caverns, dig the gold, and return to the surface for your reward. If you run out of dynamite, get to the surface fast or you'll wander the mine forever. And watch out for cave-ins – they can block your exit, or worse!

Gold Rush!

Joseph Weber
Rapid City, SD

After reading the fascinating article "Using The VIC Joystick," by David Malmberg (*Home and Educational Computing!*, Fall, 1981), I couldn't wait to try out what I had learned. It wasn't long after I got to the VIC keyboard that the Gold Rush game started to emerge.

After a few false starts, here it is.

The game gives you one more chance to use your joystick. In this game you have five miners to dig as much gold as you can from the Lost Goldmine. Each miner has ten sticks of dynamite (charges) to blast his way into the mine. The miner gathers them by moving to the same space as the dot (poke X,81).

The number of pieces of gold is counted and displayed at the bottom of the screen. The number of charges left is also displayed at the bottom of the screen. After gathering all the gold you can, you must move out of the mine to the Assay Office (the heart) to exchange the gold for money. When you touch the heart, the gold is exchanged for money at the rate of Gold times Remaining Charges.

Cave-ins Can ... Put You Out Of The Game

That all sounds easy enough, but there are several things that can happen to slow up your progress. First, there are the cave-ins. Since the Lost Goldmine is very old, cave-ins occur every time you blast. These cave-ins can block your way out or (how can I put this nicely) ah... put you out of the game. When you are caved-in upon, your gold is lost and the miner is replaced with an asterisk. If you can reach the asterisk, you regain all of the gold the "dead" miner had.

Another danger can beset you if you should use up all of your charges. When this happens, you have only a short time to get back to the surface. If you cannot exit the mine, either due to a cave-in or a slow miner, then all is lost and you end up wandering the mine forever.

Let's plug in our joysticks and strike it rich. There's gold in them there hills!

Program 1. Atari Version

Atari Version Notes

This game simulates the appearance of the VIC version by using a custom character set in Graphics mode 1. Notice that only half of the character set in ROM needs to be transferred to RAM, since Graphics modes 1 and 2 can only access 64 characters (to allow multicolor text), so we loop from 0 to 511.

This game makes extensive use of the LOCATE command to "look at" the computer's screen, such as checking for various objects the miner finds. Also, unlike the VIC version, if you clear the mine of all gold, you can start over with a new mine when you "cash in" your gold. You get three "lives."

```
100 REM ATARI GOLDRUSH
110 REM
120 REM Custom characters:
130 DATA 56,56,16,56,84,16,40,68
140 DATA 170,85,170,85,170,85,170,85
150 DATA 0,28,60,110,126,62,28,0
160 DATA 129,66,36,24,24,36,66,129
170 DATA 128,64,32,16,8,4,2,1
180 DATA 1,2,4,8,16,32,64,128
190 DATA 16,16,124,16,16,16,56,124
200 DIM CHAR$(8),WHICH(3,2)
210 CHAR$="%+-=<>%":MINER=3
220 GRAPHICS 1+16:SETCOLOR 4,6,4:SETCOLO
R 0,1,10:SETCOLOR 3,4,10
230 ? #6;"GOLDRUSH!":SETCOLOR 2,3,0
240 POSITION 9,0:? #6;"please wait"
250 CHSET=(PEEK(106)-8)*256:CHORG=57344
260 IF PEEK(CHSET+9)<>0 THEN 340
270 FOR I=0 TO 511:POKE CHSET+I,PEEK(CHO
RG+I):NEXT I
280 FOR I=1 TO 7
290 CHPOS=CHSET+(ASC(CHAR$(I))-32)*8
300 FOR J=0 TO 7
310 READ A:POKE CHPOS+J,A
320 NEXT J:NEXT I
330 FOR I=32 TO 39:POKE CHSET+I,255-PEEK
(CHORG+I):NEXT I
340 POKE 756,CHSET/256
350 POSITION 9,0:? #6;"    {D}        "
360 NUGGETS=0
370 FOR I=1 TO 22:FOR J=0 TO 19
380 IF RND(0)>0.4 THEN COLOR ASC(CHAR$(2
))+128:PLOT J,I:GOTO 400
390 IF I>3 THEN COLOR ASC(CHAR$(3)):PLOT
 J,I:NUGGETS=NUGGETS+1
400 NEXT J:NEXT I
410 CHARGES=10:POSITION 9,23:? #6;"|CHAR
```

```
GES! ";CHARGES
420 XPOS=11:YPOS=0:EMF=0:GOTO 590
430 REM MAIN LOOP
440 ST=STICK(0):TR=STRIG(0)
450 IF PEEK(20)>15 THEN POKE 709,(INT(16
*RND(0))*16+10):POKE 20,0
460 IF EMF THEN SOUND 1,T,10,8:T=T*(T<8)
+2:TI=TI+1:IF TI>200 THEN SOUND 1,0,0,0:
GOTO 1120
470 IF 1-TR THEN IF EMF=0 THEN 820
480 IF ST=15 THEN 440
490 V=-(ST=14)*(YPOS>0)+(ST=13)*(YPOS<22
)
500 H=-(ST=11)*(XPOS>0)+(ST=7)*(XPOS<19)
510 COLOR 32:PLOT XPOS,YPOS
520 XPOS=XPOS+H:YPOS=YPOS+V
530 LOCATE XPOS,YPOS,WHAT
540 IF WHAT=32 THEN 590
550 IF WHAT=ASC(CHR$(3)) THEN GOSUB 640
:GOTO 590
560 IF WHAT=ASC(CHR$(7)) THEN GOSUB 760
:GOTO 590
570 IF WHAT=4 THEN GOSUB 670:REM CASH IN
!
580 SOUND 0,100,12,8:FOR W=1 TO 20:NEXT
W:SOUND 0,0,0,0:XPOS=XPOS-H:YPOS=YPOS-V:
COLOR 138:PLOT XPOS,YPOS:GOTO 440
590 COLOR 138:PLOT XPOS,YPOS
600 IF EMF AND YPOS=0 THEN EMF=0:COLOR 3
2:PLOT XPOS,YPOS:SOUND 1,0,0,0:GOTO 410
610 FOR W=8 TO 0 STEP -1:SOUND 0,W*5,12,
W:NEXT W
620 GOTO 440
630 GOTO 630
640 FOR W=15 TO 0 STEP -1:SOUND 0,20,10,
W:NEXT W:GOLD=GOLD+1:NUGGETS=NUGGETS-1:I
F NUGGETS<0 THEN NUGGETS=0
650 POSITION 0,23:? #6;"gold ";GOLD;"  "
;
660 RETURN
670 REM CASH IN!
680 SOUND 2,4,10,4
690 FOR W=10 TO 5 STEP -1:FOR I=15 TO 0
STEP -1:SOUND 0,W,10,I:NEXT I:NEXT W
700 SOUND 2,0,0,0
710 CASH=CASH+GOLD*CHARGES:GOLD=0
720 GOSUB 650
730 POSITION 14,0:? #6;CASH
740 IF NUGGETS=0 THEN POP :GOTO 360
750 RETURN
760 REM GET THE GOLD FROM DEAD MINER
770 FOR I=3 TO 1 STEP -1
780 IF WHICH(I,0)=XPOS AND WHICH(I,1)=YP
OS THEN 800
790 NEXT I:RETURN
800 GOLD=GOLD+WHICH(I,2):GOSUB 650
810 RETURN
820 REM EXPLOSION
830 XP=XPOS+H:YP=YPOS+V:IF YP=0 THEN 440
840 RESTORE 3007
850 DATA 0,0,-1,-1,1,1,-1,1,1,-1
860 FOR I=1 TO 5:READ A,B
870 IF XP+A>=0 AND XP+A<=19 AND YP+B>=1
AND YP+B<=22 THEN LOCATE XP+A,YP+B,ZZ:IF
 ZZ=45 THEN NUGGETS=NUGGETS-1
880 NEXT I: COLOR ASC(CHR$(4))
890 IF XP>0 AND XP<20 THEN PLOT XP,YP
900 COLOR ASC(CHR$(5)):IF YP>2 AND XP>0
 THEN PLOT XP-1,YP-1
910 IF YP<22 AND XP<19 THEN PLOT XP+1,YP
+1
920 COLOR ASC(CHR$(6)):IF YP>2 AND XP<1
9 THEN PLOT XP+1,YP-1
930 IF YP<22 AND XP>0 THEN PLOT XP-1,YP+
1
940 DL=PEEK(560)+256*PEEK(561):SV=PEEK(7
12)
950 FOR W=15 TO 0 STEP -0.5:SOUND 0,50,0
,W:SW=1-SW:POKE 712,SW*(4*16+6):POKE DL,
112*SW:NEXT W
960 POKE DL,112:POKE 712,SV
970 REM
980 COLOR 32:PLOT XP,YP:IF YP>2 AND XP>0
 THEN PLOT XP-1,YP-1
990 IF YP<22 AND XP<19 THEN PLOT XP+1,YP
+1
1000 IF YP>2 AND XP<19 THEN PLOT XP+1,YP
-1
1010 IF YP<22 AND XP>0 THEN PLOT XP-1,YP
+1
1020 COLOR 138:PLOT XPOS,YPOS
1030 FOR I=1 TO 20
1040 RX=INT(20*RND(0)):RY=INT(22*RND(0)+
1)
1050 LOCATE RX,RY-1,Z2
1060 LOCATE RX,RY,Z:IF Z=32 AND Z2=171 T
HEN COLOR 171:PLOT RX,RY
1070 IF Z=138 THEN 1130
1080 NEXT I
1090 CHARGES=CHARGES-1:POSITION 17,23:?
#6;CHARGES;" ";
1100 IF CHARGES>0 THEN 440
1110 TI=0:EMF=1:POSITION 9,23:? #6;"set
out(A)    ";:GOTO 440
1120 REM DEAD MINER
1130 FOR I=14 TO 0 STEP -0.5:SETCOLOR 3,
4,I:SOUND 0,I,10,I:NEXT I:RESTORE 3400
1140 READ A,W:IF A>0 THEN SOUND 0,A,10,8
:FOR I=1 TO W*2:NEXT I
1150 IF A>0 THEN SOUND 0,0,0,0:FOR W=1 T
O 5:NEXT W:GOTO 1140
1160 COLOR ASC(CHR$(7)):PLOT XPOS,YPOS:
WHICH(MINER,2)=GOLD:GOLD=0:GOSUB 650
1170 WHICH(MINER,0)=XPOS:WHICH(MINER,1)=
YPOS:MINER=MINER-1:IF MINER=0 THEN 1210:
REM GAME OVER
1180 SETCOLOR 3,4,10:GOTO 410
1190 DATA 100,30,100,20,100,5,100,30,85,
```

```
40,90,30,100,20,105,10,100,30
1200 DATA -1,0
1210 POSITION 0,0:? #6;"game over"
1220 POKE 709,PEEK(53770)
1230 IF PEEK(53279)<>6 THEN 1220
1240 RUN
```

Program 2. VIC Version

```
100 CLR:S=0:C=10:M=4:L=0:W=0
110 PRINT"{CLEAR}"
112 GOSUB300
120 POKE36879,9
130 GOSUB800
132 Z=7712:FB=0:Y=250
140 POKEZ,32
150 IFJ1THENZ=Z+22:IFPEEK(Z)=102THENZ=Z-22
160 IFPEEK(Z)=35THENZ=Z-22
170 IFJ2THENZ=Z-1:IFPEEK(Z)=102THENZ=Z+1
180 IFPEEK(Z)=35THENZ=Z+1
190 IFJ3THENZ=Z-22:IFPEEK(Z)=102ORZ<7680THE
    NZ=Z+22
200 IFPEEK(Z)=35THENZ=Z+22
210 IFJ0THENZ=Z+1:IFPEEK(Z)=102THENZ=Z-1
212 IFPEEK(Z)=35THENZ=Z-1
220 IFPEEK(Z)=81THENS=S+1:GOSUB6000
222 IFPEEK(Z)=83THENGOSUB4000
224 IFPEEK(Z)=42THENGOSUB3000
230 IFFBTHENGOSUB5000:GOSUB8000
240 PRINT"{HOME}{20 DOWN}{REV}GOLD{OFF}";S;
    TAB(10)"{REV}CHARGES{OFF}";C
250 POKEZ,90
260 IFC=<0THENGOSUB9000
270 DD=37154:P1=37151:P2=37152
280 GOSUB500
290 GOTO140
300 REM INSTRUCTIONS
310 PRINT"{DOWN}{04 RIGHT}{REV}VIC GOLDRUSH
    {OFF}"
320 PRINT:PRINT:PRINT"{04 RIGHT}{REV}Z{OFF}
     = MINER"
330 PRINT:PRINT"{04 RIGHT}{REV}Q{OFF} = GOL
    D"
340 PRINT:PRINT"{04 RIGHT}{REV}*{OFF} = DEA
    D MINER"
350 PRINT:PRINT"{04 RIGHT}{REV}&{OFF} = DIR
    T"
360 PRINT:PRINT"{04 RIGHT}{REV}S{OFF} = ASS
    AY OFFICE"
362 PRINT:PRINT:PRINT"{02 RIGHT}USE {REV}FI
    RE BUTTON{OFF} TO"
364 PRINT"{02 RIGHT}BLAST"
370 PRINT"{HOME}{20 DOWN}{REV}PRESS ANY KEY
     TO PLAY"
380 A$="":GETA$:IFA$=""THEN380
390 PRINT"{CLEAR}"
400 RETURN
500 POKEDD,127:P=PEEK(P2)AND128
510 J0=-(P=0)
520 POKEDD,255:P=PEEK(P1)
530 J1=-((PAND8)=0)
540 J2=-((PAND16)=0)
550 J3=-((PAND4)=0)
560 FB=-((PAND32)=0)
570 RETURN
800 REM DRAW BOARD
810 X=7702
820 FORI=1TO22
830 POKEX,35
840 X=X+1
850 NEXTI
860 POKE7712,32
870 X=7724
880 FORI=1TO18
890 FORJ=1TO2
900 POKEX,35
910 X=X-1
920 NEXTJ
930 X=X+24
940 NEXTI
950 X=8098
960 FORI=1TO22
970 POKEX,35
980 X=X+1
990 NEXTI
1000 FORI=1TO180
1010 X=INT(RND(1)*374)+7724
1012 IFPEEK(X)=35THEN1010
1014 IFPEEK(X)=102THEN1010
1020 POKEX,102
1030 NEXTI
1040 FORI=7724TO8097
1050 IFPEEK(I)=102THEN1070
1052 IFPEEK(I)=35THEN1070
1060 POKEI,81
1070 NEXTI
1080 POKE7689,35:POKE7692,35:POKE7691,83
1090 GOSUB2000
1200 RETURN
2000 REM PLACE MINERS
2002 IFM<0THEN9500
2010 X=7680
2020 FORI=0TO4
2030 POKEX,32
2040 X=X+1
2050 NEXTI
2060 X=7680
2070 FORI=1TOM
2080 POKEX,90
2090 X=X+1
2100 NEXTI
2110 C=10:S=0
2120 Z=7712
2130 PRINT"{HOME}{20 DOWN}                 "
2140 FORI=8142TO8164:POKEI,32:NEXT
2150 IF M=0THENPOKE7680,32
2200 RETURN
3000 REM DIG UP MINER
3010 S=S+S1
3200 RETURN
4000 REM TALLY GOLD

4002 FORI=7694TO7701:POKEI,32:NEXT
4010 FORJ=1TO5
4020 FORI=15TO0STEP-1
4030 POKE36878,I
4040 POKE36876,230
4050 FORT=1TO10:NEXT
4060 NEXTI
4070 POKE36876,0
4080 NEXTJ
4090 C1=C:IFC1=0THENC1=1
4100 W=S*C1+W
4110 PRINT"{HOME}{REV}{WHT}{13 RIGHT}$";STR$
     (W);".00{RED}"
4120 POKE7691,83
4130 M=M-1
4140 GOSUB2000
4150 RETURN
```

All the advantages of Mupet, PLUS...

MUPET II

saves money and time too!

Mupet II offers all the advantages of Mupet, plus:

- Can protect against incorrect use of harmful disk commands.
- No more printer interface - background printing supported on IEEE, RS 232C and parellel ports included in controller.
- Programmable 32K RAM controller provides user defineable operating systems.
- Up to 16 computers may be connected to central printer and disk drives.

Integrate intra-office terminals for common data access allowing independent analysis for such functions as:

- Accounting
- Sales forecasts, budgets and orders
- Production scheduling
- Word processing

The only product available capable of serving Daisy Chained Commodore Computers up to 100 feet.

CMD

CANADIAN MICRO DISTRIBUTORS LTD.

500 Steele Ave.,
Milton, Ontario L9T 3P7
416-878-7277

MUPET II
a small price for
huge capabilities, from:

$995.00

```
5000 REM EXPLOSION
5002 IFC<=0THEN9000
5004 IFPEEK(Z-1)=35THEN5012
5010 POKEZ-1,86
5012 IFPEEK(Z+1)=35THEN5022
5020 POKEZ+1,86
5022 IFPEEK(Z+22)=35THEN5032
5030 POKEZ+22,86
5032 IFPEEK(Z-22)=35THEN5100
5040 POKEZ-22,86
5100 POKE36877,220
5110 FORL=15TO0STEP-1
5120 POKE36878,L
5130 FORT=1TO75:NEXT
5140 NEXTL
5150 POKE36877,0:POKE36878,0
5152 IFPEEK(Z-1)=35THEN5156
5154 POKEZ-1,32
5156 IFPEEK(Z+1)=35THEN5160
5158 POKEZ+1,32
5160 IFPEEK(Z+22)=35THEN5164
5162 POKEZ+22,32
5164 IFPEEK(Z-22)=35THEN5170
5166 POKEZ-22,32
5170 C=C-1:POKE8138,32:POKE8139,32:POKEZ,90
5300 RETURN
6000 REM DOT PICKUP
6010 POKE36878,15
6020 FORL=10TO1STEP-1
6030 POKE36876,L*25
6040 NEXTL
6050 FORL=1TO10
6060 POKE36876,L*25
6070 NEXTL
6080 POKE36876,0:POKE36878,0
6100 RETURN
8000 REM CAVE IN
8010 X=INT(RND(1)*1)+10
8020 FORI=1TOX
8030 V=INT(RND(1)*418)+7702
8032 IFPEEK(V)=35THEN8030
8034 IFPEEK(V)=90THENGOSUB8600:GOTO132
8036 IFPEEK(V)=81THEN8030
8040 POKEV,102
8050 NEXTI
8100 RETURN
8600 REM SQUASH MINER
8610 POKE36878,15
8620 FORI=255TO128STEP-4
8630 POKE36874,I
8640 FORT=1TO30:NEXT
8650 NEXTI
8660 POKE36878,0:POKE36874,0
8670 M=M-1
8680 POKEZ,42
8682 S1=S
8690 GOSUB2000
8800 RETURN
9000 REM GET OUT COUNTER
9004 IFY<128THENPOKEZ,32:GOSUB2000:GOTO132
9010 PRINT"{HOME}{21 DOWN}{02 RIGHT}{REV}TIM
    E TO GET OUT{OFF}"
9020 POKE36878,15:POKE36876,Y
9030 FORT=1TO30:NEXT
9040 Y=Y-4
9050 POKE 36878,0:POKE36876,0
9100 RETURN
9500 REM PLAY AGAIN
9520 PRINT"{HOME}{21 DOWN}{WHT}{REV}PLAY AGA
    IN?Y=YES N=NO{OFF}{RED}"
9530 IFA$="Y"THEN100
9540 A$="":GETA$:IFA$=""THEN9540
9560 IFA$="Y"THEN100
9999 END
```

©

You've seen the bank ads: "Retire a Millionaire." Type in this short program (it will run on Atari, Radio Shack Color Computer, PET/CBM, VIC, and Apple) and see for yourself how IRA accounts compute. The program uses very little memory. The "^" symbol means "to the power of."

IRA Planner

Richard and Betty Givan
Richmond, KY

Most get-rich schemes of the past have proven to be of questionable legality and dubious worth. The latest promotion, however, is endorsed by the US Government and seems foolproof. This device is the Individual Retirement Account (IRA), expanded in 1982 to allow up to a $2000 ($2250 in a joint plan with a nonworking spouse) yearly deposit to be put into a private retirement account.

This amount is deductible from the person's gross income during the year deposited, decreasing the income tax accordingly. The retirement fund is then free to grow at the prevailing competitive interest rate – compounded daily and tax free – until it is withdrawn during retirement. Although taxes are then due, presumably the taxpayer will be in a lower tax bracket and so pay a lesser tax.

The Relationship Between Inflation And Interest

The allure of the plan is in the rapid growth of the principal through compound interest at the currently high rates. Hence the ads in which banks all but guarantee that you can be a millionaire upon retirement via a $2000 yearly deposit for 35 years at a 12% return. Actually, your account *would* be worth an astounding $1,161,059. Such a modest sacrifice in order to retire a millionaire!

As with all get-rich plans, however, there is a catch – not a legal or even an ethical one, but a matter of economics. The IRA promotion campaigns conveniently overlook the devastating effects of inflation on your million. At the same time that compound interest is building your fortune, inflation is eroding it. Historically, the interest rate is fairly well dictated by the rate of inflation. Although temporary imbalances occur, economists generally agree that, in the long run, the interest rate will seek out a level approximately three to four percent higher than the inflation rate.

Assuming that the prevailing inflation rate (say nine percent) holds steady, your retirement fortune of $1,161,059 from the above example will really be worth only $56,875 in the purchasing power of 1982 dollars. You may have a carload of dollars in the year 2017, but the Cadillac you buy to haul it home would cost $306,000 and the gasoline to power it would be $25 a gallon!

This is not to say that an IRA is a bad way to save. It does offer immediate tax relief, and that in itself might provide you with the incentive to put aside some funds for your golden years. But it would be well to put the numbers in perspective when planning for your future.

The program asks you several questions: the amount of money you wish to set aside each year; the tax bracket you are currently in (which can be found by reference to the IRS booklet accompanying your tax forms, but is not really essential to the rest of the program); your age when you begin and end the plan; and the average interest and inflation rates you expect to experience (here's where a crystal ball program would be nice).

The program then displays the tax savings you would receive the first year in the plan. (Income and tax rate would probably fluctuate too much to benefit from attempting to compute these over the life of your IRA.) The sum of your deposits is displayed, followed by the principal of the account increased by accumulated interest. Then the *real spending power* of your final nest egg is shown by reducing the principal to reflect the inflation rate. You can see its worth in terms of the 1982 dollar. Bear in mind that this money is taxable when withdrawn, also. You can then easily experiment with different interest and inflation rates at this point without having to answer the other questions again.

One note: the two questions about inflation and interest ask for the figures *expressed as decimals*. In other words, if you want to answer 12% inflation, you should type .12 and 6% interest would be entered as .06.

```
10 REM IRA PLANNER
30 DIM A$(1):REM THIS LINE ONLY NECESSARY 
   FOR ATARI
40 PRINT"AT WHAT AGE DO YOU PLAN"
50 PRINT"TO OPEN AN IRA ACCOUNT";:INPUT A
60 PRINT:PRINT"AT WHAT AGE DO YOU PLAN TO 
   RETIRE";:INPUT A1
70 Y=A1-A
80 PRINT:PRINT "HOW MUCH DO YOU PLAN TO"
90 PRINT"DEPOSIT PER YEAR";:INPUT D:C=D
100 PRINT:PRINT"WHAT IS YOUR TAX BRACKET?"
110 PRINT"(ENTER PERCENTAGE EXPRESSED":PRIN
    T"AS A DECIMAL)":INPUT P
120 PRINT:PRINT"WHAT IS THE AVERAGE INTERES
    T RATE YOU
130 PRINT"EXPECT FOR THE ACCOUNT OVER THE
140 PRINT"YEARS IT EXISTS? (ENTER PERCENTAG
    E
150 PRINT"EXPRESSED AS A DECIMAL)":INPUT R
160 PRINT:PRINT"WHAT IS THE AVERAGE INFLATI
```

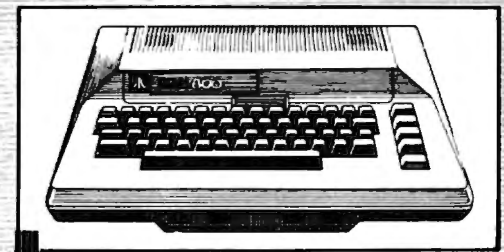
NEECO

JUST A SAMPLE OF THE MANY PRODUCTS WE CARRY, CALL US FOR OUR NEW 80-PAGE CATALOG.
WE WILL MATCH SOME ADVERTISED PRICES ON CERTAIN PRODUCTS LISTED UNDER SIMILAR "IN STOCK" CONDITIONS.

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



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

NEC Spinwriter Printer

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

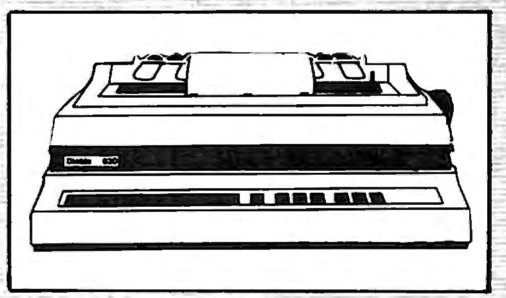
Tractor Feed Options are available

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



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

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

```
    ON RATE"
170 PRINT"YOU EXPECT DURING THE YEARS"
180 PRINT"BETWEEN OPENING THE ACCOUNT AND
190 PRINT"RETIREMENT? (ENTER PERCENTAGE
195 PRINT"EXPRESSED AS A DECIMAL)":INPUT I
200 S=D*P
210 PRINT:PRINT"YOU WILL SAVE $";INT(S);" O
    N TAXES THIS YEAR."
220 T=D*Y
230 PRINT:PRINT"THE TOTAL AMOUNT DEPOSITED ~
    INTO YOUR
240 PRINT"IRA ACCOUNT OVER THE ";Y;" YEARS ~
    IS":PRINT"$";INT(T);"."
250 FOR J=1 TO Y
260 X=D*(1+R/365)^365
270 D=X+C
280 NEXT J
290 PRINT:PRINT"WHEN YOU RETIRE, THE AMOUNT
     IN YOUR"
300 PRINT"ACCOUNT WILL BE $";INT(X);"."
310 Z=(1+I)^Y
320 W=X/Z
330 PRINT:PRINT"...WHICH IS WORTH $";INT(W)
    ;" IN 1982"
340 PRINT"DOLLARS."
350 PRINT:PRINT"WOULD YOU LIKE TO TRY THIS ~
    AGAIN WITH
360 PRINT"DIFFERENT INTEREST AND INFLATION ~
    RATES"
370 PRINT"(Y OR N)";:INPUT A$
380 IF A$="N" THEN END
390 D=C:GOTO 120
```

With versions of this excellent game for PET/CBM Microsoft and Atari BASICs, this should provide hours of amusement. It will run on computers with 8K or more of RAM memory.

Maze Race

Mike Peterson
Cheyenne, WY

I was impressed by the Maze algorithm that appeared in **COMPUTE!**, December, 1981, #19. After watching the maze form several times, I had an idea for a program that incorporated the maze algorithm.

First, I modified the routine so that two symmetrical mazes would be formed. One starts in the upper left corner of the screen while the other starts in the lower right. By POKEing a 32 ($20) in the center of the screen, identical (but opposite) mazes are formed. As one goes left, the other goes right. And as one goes down, the other goes up. This continues until the screen is full and the algorithm completed. The mazes are then linked together in the screen's middle and a "target" character is POKEd in the center.

Now, depending upon your choice of options during initialization, you race against a human opponent (or the computer) to the center of the maze. If you select the two-player option, I strongly recommend the use of joysticks. The program will process both player's joystick input during each cycle. On the keyboard, one player can prevent the other's input by holding a key pressed.

With the one-player version, you will always be on the right and use the right joystick or the number pad to control your movement. The computer will not begin until after your first input. Since the computer doesn't know the best path to follow, and strays down dead-end paths, it is easy to beat. My daughter loses just often enough to keep her interested and make each game a new challenge.

Through experimentation with the joysticks, I found that the left joystick sets the high four bits and the right joystick sets the low four bits of the byte at address 59471 ($E84F). The byte, A, can be separated into a left value and a right value with : A= PEEK(59471) : L= (A/16) AND 15 : R= A AND 15. If both joysticks are in the same position, L=R. An array with 15 elements can be used to determine the new position of each player, move there if not off the screen, and then erase the previous position. The following table shows all possible joystick values, positions, and screen movement offsets:

L or R	POSITION	OFFSET
0	not used	0
1	left/button	–1
2	right/button	+1
3	button	0
4	not used	0
5	left/up	–41
6	right/up	–39
7	up	–40
8	not used	0
9	left/down	+39
10	right/down	+41
11	down	+40
12	not used	0
13	left	–1
14	right	+1
15	center	0

Note: with the button pressed only the row can be determined.

Another feature of this program is the "flipping" of the marker that arrives in the center of the screen's maze first. This is performed by XORing the byte in the center screen address with 128 ($80). The result of an XOR with 128 is that RVS characters become normal characters and vice versa.

Since BASIC doesn't provide us with an XOR function, I consulted a Boolean Algebra text and found the AND, OR, and NOT gates used to produce the XOR truth table. In BASIC, XOR would be this : Y= ((X AND (NOT 128)) OR ((NOT X) AND 128)).

Program 1. Atari Version

Atari Notes

After the dual maze has been generated, select the number of players (player vs. computer or player vs. player) by pressing SELECT. When ready to play, press START. To re-start the game, press START. Use joysticks plugged into jacks one and two to play.

```
100 REM ***   MAZE RACE   ***
110 REM *** ATARI VERSION ***
120 DIM A(3):REM SET UP DIRECTION TABLE
130 A(0)=2:A(1)=-80:A(2)=-2:A(3)=80
140 WL=128:HL=0:SC=PEEK(88)+256*PEEK(89)
:A=SC+43:C=SC+877
150 GRAPHICS 0:POKE 752,1:SETCOLOR 2,INT
(16*RND(0)),4:POKE 712,PEEK(710)
160 FOR I=1 TO 23
```

```
170 PRINT "I
         I"
180 NEXT I
190 REM GENERATE THE MAZE!
200 POKE A,5:POKE C,2:POKE SC+499,HL
210 J=INT(RND(0)*4):X=J
220 B=A+A(J):D=C-A(J):IF PEEK(B)<>WL THE
N 240
230 POKE B,J+1:POKE D,J+1:POKE A+INT(A(J
)/2),HL:POKE C-INT(A(J)/2),HL:A=B:C=D:GO
TO 210
240 J=(J+1)*(J<3):IF J<>X THEN 220
250 J=PEEK(A):POKE A,HL:POKE C,HL:IF J<5
 THEN A=A-A(J-1):C=C+A(J-1):GOTO 210
260 A=SC+43:C=SC+877:J=2
270 POKE A,84:POKE C,111:POKE SC+459,10:
POKE SC+460,HL:POKE SC+458,HL
280 DIM M(15):FOR I=0 TO 15:M(I)=0:NEXT
I:NP=0
290 M(14)=-40:M(13)=40:M(11)=-1:M(7)=1:M
(10)=-41:M(6)=-39
300 M(9)=39:M(5)=41:PLR=1
310 DIM MS$(20):MS$="Maze Race":FOR I=1
TO LEN(MS$):POSITION 0,I+5:? MS$(I,I):NE
XT I
320 POSITION 9,23:? "One Player Vs. Comp
uter";
330 K=PEEK(53279):IF K=7 THEN 330
340 IF K=PEEK(53279) THEN 340
350 IF K=6 THEN 400
360 IF K=3 THEN 330
370 NP=1-NP:IF NP=0 THEN 320
380 POSITION 9,23:? "  Player Vs. Player
   ";
390 GOTO 330
400 PLR=1-PLR
410 IF PLR=0 OR NP THEN 440
420 B=A+INT(A(J)/2):Y=PEEK(B):IF Y=HL OR
 Y=10 THEN POKE B,84:POKE A,HL:A=B:J=(J+
2)-4*(J>1)
430 J=(J-1)+4*(J=0):GOTO 480
440 IF PLR THEN 470
450 D=C+M(STICK(PLR)):IF PEEK(D)=HL OR P
EEK(D)=10 THEN POKE D,111:POKE C,HL:C=D
460 GOTO 480
470 B=A+M(STICK(PLR)):IF PEEK(B)=HL OR P
EEK(B)=10 THEN POKE B,84:POKE A,HL:A=B
480 REM SOMEONE WON
490 IF A<>SC+459 AND C<>SC+459 THEN 400
500 P=PEEK(SC+459):PF=0
510 FOR I=1 TO 50:POKE SC+459,P+128*PF:P
F=1-PF:NEXT I
520 REM
530 IF C=SC+459 THEN 560
540 IF NP THEN POSITION 9,23:? "    Play
er 2 Won!    ";:GOTO 580
550 POSITION 9,23:? "         I WIN!
     ";:GOTO 580
560 IF NP THEN POSITION 9,23:? "   Playe
r 1 Won!    ";:GOTO 580
570 POSITION 9,23:? "       You win!
   ";
580 IF PEEK(53279)<>6 THEN 580
590 RUN
```

If you want to make Maze Race (for Atari) more challenging, add the following lines. They cause the screen to blank out during the maze generation, preventing a "sneak preview." A side effect is that this decreases the time necessary to generate the maze:

```
155 POKE 559,0
325 POKE 559,34
```

Program 2. Microsoft Version

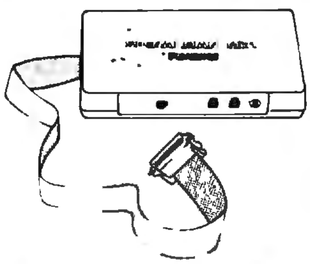
```
10 REM *** MAZE RACE ***
20 REM * MIKE PETERSON *
30 REM ** CHEYENNE,WY **
40 REM
50 REM MAZE GENERATOR BY CHARLES BOND
60 REM COMPUTE! : DECEMBER, 1981
70 REM
80 GOTO120
90 POKE167,0
100 GETZ$:IFZ$=""THEN100
110 PRINTZ$:Z=VAL(Z$):POKE167,1:RETURN
120 PRINT"{CLEAR}{05 DOWN}":PRINTTAB(15)"MA
    ZE RACE{02 DOWN}"
130 PRINTTAB(5)"HOW MANY PEOPLE ARE PLAYING
     ?{LEFT}";:GOSUB90:NP=Z-1:PRINT
140 IFZ>2THENPRINTTAB(5)"SORRY, ONLY 2 CAN
    PLAY.{DOWN}":GOTO130
150 IFZ=0THENPRINTTAB(5)"PLEASE ENTER 1 OR
    2.{DOWN}":GOTO130
160 PRINTTAB(7)"ARE YOU USING JOYSTICKS ?{L
    LEFT}";:GOSUB90:PRINT
170 IFZ$="Y"THENJJ=1:GOTO190
180 IFZ$<>"N"THENPRINTTAB(7)"PLEASE ENTER Y
     OR N.{DOWN}":GOTO160
190 IFNPTHENPRINT"{DOWN}LEFT PLAYER = Q :RI
    GHT PLAYER = W":GOTO210
200 PRINT"{DOWN}I AM THE Q :YOU ARE THE W"
210 IFJJTHENPRINT"{DOWN}USE THE JOYSTICK TO
     MOVE AROUND":GOTO250
220 IFNPTHENPRINT"{DOWN}Q USES W FOR UP,A F
    OR LEFT,D FOR RIGHT,"
230 IFNPTHENPRINT"AND X FOR DOWN."
240 PRINT"{DOWN}W USES 8 FOR UP,4 FOR LEFT,
    6 FOR RIGHT, AND 2 FOR DOWN."
250 PRINT"{DOWN}PRESS ANY KEY TO BEGIN *{LE
    LEFT}";:GOSUB90
260 DIMM(255):JS=151:IFJJTHENJS=59471:GOTO2
    90
270 M(18)=40:M(42)=-1:M(41)=1:M(50)=-40
280 M(24)=40:M(48)=-1:M(47)=1:M(56)=-40:GOT
    O300
290 M(7)=-40:M(11)=40:M(13)=-1:M(14)=1
```

```
300 DIMA(3):A(0)=2:A(1)=-80:A(2)=-2:A(3)=80
    :WL=160:HL=32:SC=32768
310 BL$="{REV}
                        {OFF}"
320 A=SC+81:C=SC+917
330 PRINT"{CLEAR}":FORI=1TO23:PRINTBL$:NEXT

340 POKEA,4:POKEC,1:POKESC+499,HL
350 J=INT(RND(TI)*4):X=J
360 B=A+A(J):D=C-A(J):IFPEEK(B)<>WLTHEN380
370 POKEB,J:POKED,J:POKEA+A(J)/2,HL:POKEC-A
    (J)/2,HL:A=B:C=D:GOTO350
380 J=(J+1)*-(J<3):IFJ<>XTHEN360
390 J=PEEK(A):POKEA,HL:POKEC,HL:IFJ<4THENA=
    A-A(J):C=C+A(J):GOTO350
400 POKESC+498,HL:POKESC+499,42:POKESC+500,
    HL:A=SC+81:C=SC+917
410 POKEA,81:POKEC,87:J=2:K=2
420 PP=PEEK(JS):IFPP=255THEN420
430 PP=PEEK(JS)
440 IFNPTHEN470
450 B=A+A(J)/2:Y=PEEK(B):IFY=HLORY=42THENPO
    KEB,81:POKEA,HL:A=B:J=(J+2)+4*(J>1
    )
460 J=(J-1)-4*(J=0):GOTO510
470 IFJJTHEN500
480 IFPP<>24ANDPP<>48ANDPP<>47ANDPP<>56THEN
    520
490 B=A+M(PP):IFPEEK(B)=HLORPEEK(B)=42THENP
    OKEB,81:POKEA,HL:A=B:GOTO510
500 Q=(PP/16)AND15:B=A+M(Q):IFPEEK(B)=HLORP
    EEK(B)=42THENPOKEB,81:POKEA,HL:A=B

510 IFJJTHEN540
520 IFPP<>18ANDPP<>42ANDPP<>41ANDPP<>50THEN
    550
530 D=C+M(PP):IFPEEK(D)=HLORPEEK(D)=42THENP
    OKED,87:POKEC,HL:C=D:GOTO550
540 Q=(PPAND15):D=C+M(Q):IFPEEK(D)=HLORPEEK
    (D)=42THENPOKED,87:POKEC,HL:C=D
550 IFA<>SC+499ANDC<>SC+499THEN430
560 FORI=0TO99:W=PEEK(SC+499):X=(WAND(NOT12
    8))OR((NOTW)AND128):POKESC+499,X
570 NEXT
580 PRINT"ANOTHER GAME ?{LEFT}";:GOSUB90
590 IFZ$="Y"THEN320
600 IFZ$<>"N"THENPRINT"{02 UP}":GOTO580
```

SPECIALS on INTEGRATED CIRCUITS

6502	7.45	10/6.95	50/6.55	100/6.15
6502A/6512A	8.40	10/7.95	50/7.35	100/6.90
6520 PIA	5.15	10/4.90	50/4.45	100/4.15
6522 VIA	6.45	10/6.10	50/5.75	100/5.45
6532	7.90	10/7.40	50/7.00	100/6.60
2114-L200		2.45	25/2.30	100/2.15
2114-L300		2.25	25/2.10	100/2.00
2716 EPROM		5.50	5/5.15	10/4.80
2532 EPROM		9.90	5/9.45	10/8.90
6116 Hitachi 2Kx8 CMOS RAM		9.90	5/9.45	10/8.90
4116-200 ns RAM				8 for 15
Zero Insertion Force 24 pin Socket				2.00
S-100 Wire Wrap Socket				2.40

A P Products 15% OFF
A P Hobby-Blox 15% OFF



MODEM SPECIAL $99

SIGNALMAN Mk1 from Anchor Automation

DIRECT CONNECT Modem with RS232 Cable and Connector included. Fully compatible with all Bell 103 modems. 0 to 300 bps, full duplex, frequency shift keyed modulation, automatic ANSW/ORIG selection, direct connect, audible tone carrier detect indicator, self-contained battery powered

PET/CBM Version (Mk1P) $169
ATARI Version $99

COMPACK INTELLIGENT TERMINAL PACKAGE $115
Combines intelligent RS232 port hardware from cgrs Microtech with EHS Intelligent Terminal software to allow you to connect any RS232 Modem to PET/CBM

RS232 MODEM – STAR SALE $128
IEEE 488 MODEM – STAR SALE $199

We carry Apple II+ from Bell & Howell

16K RAM Card for Apple 95

fullFORTH+ for Apple II $75
A full-featured FORTH with enhancements. Conforms to FORTH Interest Group standards.

REVERSAL (Spracklen) 32K Apple	28.00
Data Manager (Lutus) 24K Apple	40.00
Data-Graph (Boyd) 48K Apple	40.00
Apple II User's Guide (Osborne)	12.00
Introduction to Pascal (Sybex)	10.30
Pascal Handbook (Sybex)	12.00
Musical Applications of Micros (Chamberlin)	20.00
User Guide to the Unix System	13
6502 Assembly Language Subroutines	11
PET Fun and Games	8

commodore

CBM-PET SPECIALS

	list	SALE
8023 Printer – 136 col, 150cps bi-directional	(995)	775
8300 Daisy Wheel – 40 cps bi-directional	(2250)	1750
8032 80 x 25 CRT, business keyboard	(1495)	1100
Super Pet	(1995)	1600
8096 Board (extra 64K RAM for 8032)	(500)	400
8050 Dual Disk Drive - 1 megabyte	(1795)	1345
8250 Dual Disk Drive - 2 megabyte	(2195)	1760
CBM IEEE Modem	(395)	199
4016 full size graphics keyboard	(995)	795
4032 full size graphics keyboard	(1295)	999
4040 Dual Disk Drive - 330,000 bytes	(1295)	999
2031 Single Disk Drive - 165,000 bytes	(695)	550
4022 Tractor Feed Printer	(795)	630
C2N External Cassette Deck	(75)	65
Used CBM/PET Computers		CALL

WE WILL MATCH ANY ADVERTISED PRICE

***** EDUCATIONAL DISCOUNTS *****
Buy 2 PET/CBM Computers, receive 1 FREE

PETSCAN I $345 base price.
Allows you to connect up to 20 CBM/PET Computers to shared disk drives and printers. Completely transparent to the user. Perfect for schools or multiple word processing configurations. Base configuration supports 3 computers.

VIC 20	259	VIC Superslot	24
VIC 1515 Printer	335	VIC Super Alien	24
VIC 3K RAM	36	VIC Maze in 3-D	12
VIC 8K RAM	54	VIC Cosmic Debris	9
VIC 16K RAM	112	VIC Amok (UMI)	17
VIC 1540 Disk Drive	500	VIC Snakman	13
VIC Invaders	24	VIC Rubik's Cube	13
VIC Jupiter Lander	24	VIC Night Rider	11

TNW 1000 Serial Interface	110
TNW 488/103 with DAA	450
Compute!'s First Book of PET /CBM	11
WordPro 3 Plus - 32K CBM, disk, printer	195
WordPro 4 Plus - 8032, disk, printer	300
VISICALC for PET, ATARI, or APPLE	190
SM-KIT - PET ROM Utilities	40
Programmers Toolkit - PET ROM Utilities	35
PET Spacemaker II ROM Switch	36
2 Meter PET to IEEE or IEEE to IEEE Cable	40
Dust Cover for PET	8
IEEE-Parallel Printer Interface for PET	110
IEEE-RS232 Printer Interface for PET	120
The PET Revealed	12
Library of PET Subroutines	12

4 PART HARMONY MUSIC SYSTEM for PET
The Visible Music Monitor, by Frank Levinson, allows you to easily enter, display, edit, and play 4 part harmony music. Includes whole notes thru 64ths (with dotted and triplets), tempo change, key signature, transpose, etc. The KL-4M unit includes D to A converter and amplifier ready to hook to your speaker.
KL-4M Music Board with VMM Program $60

DISK SPECIALS

SCOTCH (3M) 5"	10/2.45	50/2.35	100/2.30
SCOTCH (3M) 8"	10/2.60	50/2.45	100/2.40

WE STOCK VERBATIM DISKS
Try the new Verbatim Head Cleaner Kits

BASF 5" or 8"	10/2.00	20/1.95	100/1.85
Wabash 5"	10/2.00	20/1.95	100/1.85
Wabash 8"	10/2.00	20/1.95	100/1.85

WE STOCK MAXELL DISKS

Diskette Storage Pages		10 for 3.95
Disk Library Cases	8" - 2.85	5" - 2.15
Disk Hub Rings	8" - 50 @ 7.50	5" - 50 @ 6.00

CASSETTES - AGFA PE-611 PREMIUM
High output, low noise, 5 screw housings.

C-10	10/.56	50/.50	100/.48
C-30	10/.73	50/.68	100/.66

SPECIALS

EPSON MX-80 Printer	440
EPSON MX-80 F/T Printer	540
EPSON MX-70 Printer	300
EPSON MX-100 Printer	695
Centronics 739 Printer with dot graphics	575
STARWRITER Daisy Wheel Printer F10	1445
Zenith ZVM-121 Green Phosphor Monitor	119
DC Hayes Smartmodem	230
Watanabe Intelligent Plotter 1195 6-Pen	1445
BMC BM12A Green Phosphor Monitor	99
BMC BM1400BLUC 13" Color Monitor	285
BMC RGB 13" Color Monitor	329
POWERMATE Multiple Outlet Power Strips	
ISOBAR Isolated and Filtered Outlet Strips	

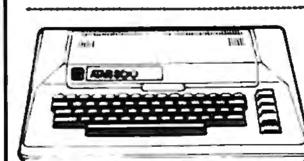
ALL BOOK and SOFTWARE PRICES DISCOUNTED

Synertek Systems

SYM-1 Microcomputer SALE	199
SYM BAS-1 BASIC or RAE 1/2 Assembler	85
KTM-2/80 Synertek Video and Keyboard	349
KTM-3/80 Synertek Tubeless Terminal	385

ZENITH data systems

Z90-80 64 K	2170
Z90-82 64K, 1 double dens. drive	2395
Z89-0 48K	1950
Z89-1 48K, 1 drive	2150
Z67 10 Megabyte + Floppy Drive	4495
Z37 1.3 Megabyte Dual Floppy	1495
Z25 High Speed Printer	1195
Z19 Video Terminal (VT-52 compatible)	695
ZVM-121 Green Phosphor Monitor	119
All Zenith Software discounted	



ATARI SPECIALS

800 Computer	675	Microsoft BASIC	72
400 - 16K	330	**MISSILE COMMAND**	32
810 Disk Drive	440	**ASTEROIDS**	32
825 Printer	629	**STAR RAIDERS**	37
850 Interface	170	Space Invaders	32
Inside Atari DOS	18	Music Composer	45
Paddle Pair	19	**MODEM for Atari**	99
Joystick Pair	19	Super Breakout	30
16K RAM	69	**PAC-MAN**	36
32K RAM	99	**CENTIPEDE**	36
Pilot	68	First Book of Atari	11

Write for prices on other Atari items.

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.

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, Relocater, 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**

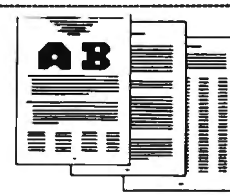
Features include:
- full FIG FORTH model.
- all FORTH 79 STANDARD extensions.
- structured 6502 Assembler with nested decision making macros.
- full screen editing (same as when programming in BASIC).
- auto repeat key
- sample programs
- standard size screens (16 lines by 64 characters).
- 150 screens 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

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.

Hayden Software

Complex Mathematics	12.70
Engineering Mathematics	12.70
General Mathematics	12.70
MCAP:Circuit Analysis Program	21.00
Energy Miser	24.50

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.

Test RAM For Bad Bits, Nondestructively

Leo J. Scanlon
Inverness, FL

In a recent article in this magazine (**COMPUTE!**, April, 1981 #23) I presented a 6502 assembly language program that tests the integrity of a selected portion of RAM. That program was designed to detect "dead" bits or bytes, pattern sensitivity, crosstalk, and a variety of other error conditions. It could also be used to detect soft errors, in which the memory accepts the test data, but reverts back to its previous state after some period of time.

As useful as it is, that program has one possible shortcoming: it clobbers the contents of the portion of memory being tested. Clearly, that doesn't matter if you are just verifying a newly installed memory board, but is unacceptable if a program or some data is sitting within the test area. In this article, I present another kind of program, one that performs a *nondestructive* test on RAM memory. That is, a program that alters memory, but subsequently restores all locations to their previous (pre-test) values.

The Test Algorithm

Essentially, the test program described here validates RAM by comparing the actual contents of memory to the known data that should be contained within it. To make this comparison, the program uses a method that is often employed for testing punched paper tape and read only memories (ROMs) – the *checksum*. A checksum is that value produced by taking the exclusive-OR of all bytes in test memory (see box).

Briefly, here is the sequence of operations for the test program:

1. Calculate a checksum value for the entire range of test memory, by exclusive-ORing all bytes.
2. Invert the state of the first bit in test memory – Bit 7 of the "start" location – but leave all other bits unchanged.
3. Calculate a new checksum value.
4. Invert the state of the altered bit position in the new checksum.
5. Compare the new (altered) checksum with the initial checksum.
6. The result of this comparison can cause either of two things to take place:

    If the checksums are different, the program jumps to an error routine, to print out the bit position and address of the bad bit.

    If the checksums are identical, the program restores the state of the test bit – by reinverting it – then branches back to Step 2, to test the next bit (Bit 6 of the "start" location).

This process continues until all bits have been tested, or until a mismatch is detected.

Will this nondestructive test program catch all of the fault conditions that can be detected by the previously published destructive test program? Probably not all of them. The nondestructive test program will not detect pattern sensitivity or soft errors (unless you modify the program to include a time delay), but it should be able to detect most other types of errors.

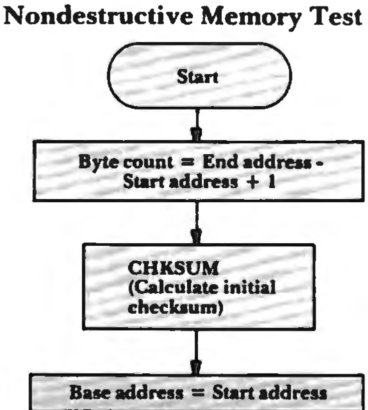
Program Flowchart

Now that you understand *what* the test program must do, and know *how* the program will do it, it's time to look at the structure of the program itself. This program is comprised of three parts: a main program loop, a checksum calculating subroutine and an error printout routine.

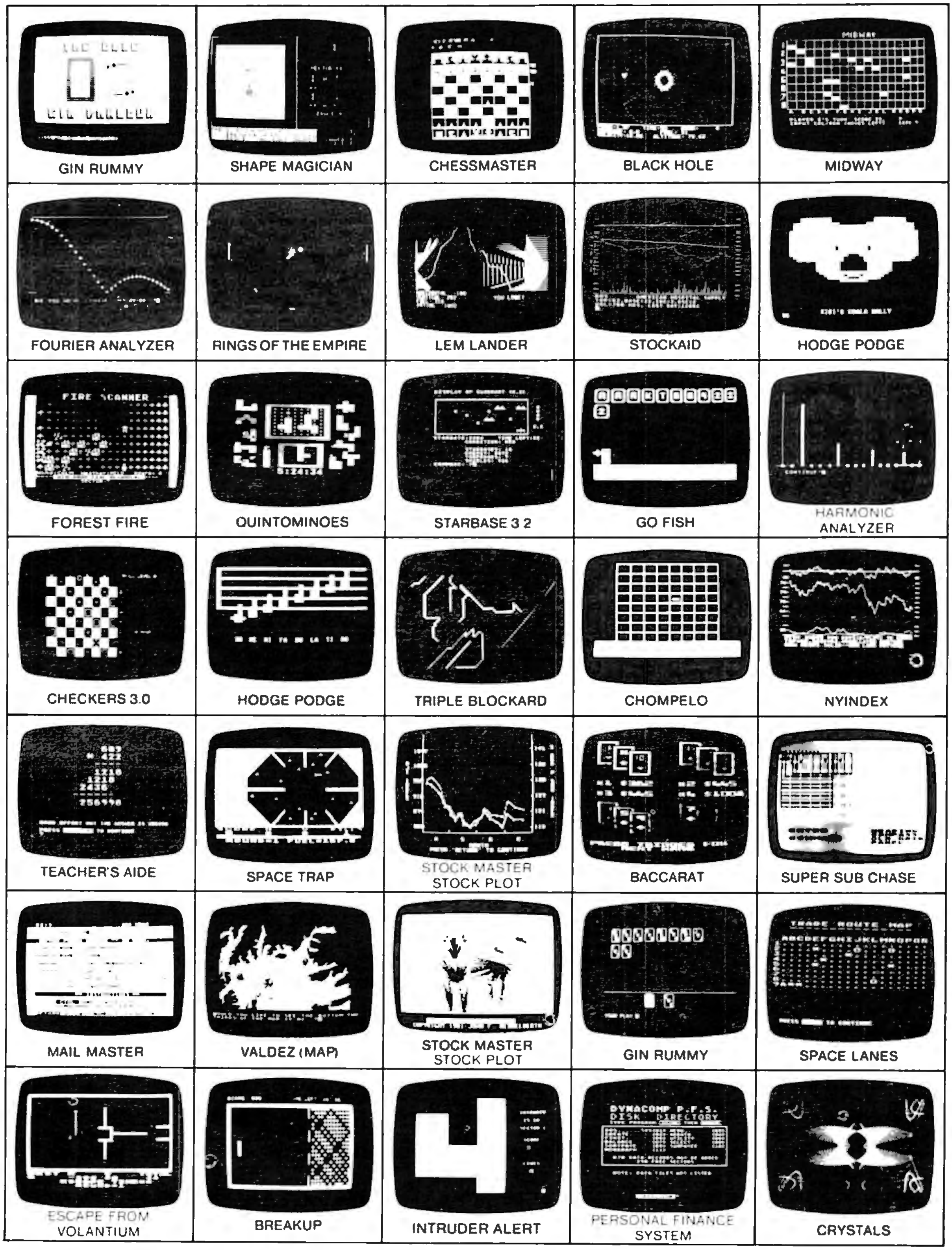
A flowchart for the main program loop is shown in Figure 1. As you can see, this flowchart is nothing more than a detailed version of the algorithm we defined in preceding Steps 1 through 6. The program begins by calculating the byte count, then calls the checksum subroutine (CHKSUM) to generate the initial value of the checksum. This done, the base address and byte index are initialized to reference the first byte in test memory.

Next, the bit mask index is initialized to reference the most significant bit, Bit 7. With this initialization out of the way, the program inverts the current test bit. The first time through the loop, this will be Bit 7 of the Start location. Now the

Figure 1: Nondestructive Memory Test Program



It's hard to picture all of DYNACOMP's software



TURN TO NEXT PAGE....

BUSINESS and UTILITIES

NEW
MAILMASTER (Atari diskette only) — Price: $39.95 Diskette
MAILMASTER is a very versatile software package for managing and manipulating mail lists and mini data bases. Each disk can hold over 600 customer entries containing name, address, 38 letter key words and a phone number. The display is marked so that entries may be made and edited with ease. The status (e.g., disk space left, options, etc.) is shown at all times. Labels may be printed 1,2 or 3 up, and all sorting (zip code and alphabetic) is performed by a fast machine language program.

NEW
PERSONAL FINANCE SYSTEM (Available for all computers) — Price: $39.95 Diskette
PFS is a single diskette, menu-oriented system composed of ten different programs. Besides recording your expenses and tax deductible items, PFS will sort and summarize expenses by payee, and display information on expenditures by any of 26 user defined codes by month or by payee. PFS will even produce monthly bar graphs of your expenses by category! This powerful package requires only one disk drive, minimal memory (24K Atari, 32K North Star) and will store up to 600 records per disk (and over 1000 records per disk by making a few simple changes to the programs). You can record checks plus cash expenses so that you can finally see where your money goes and eliminate guesswork and tedious hand calculations. Contains high speed machine language sort. PFS has been demonstrated on network (CBS) TV!

FAMILY BUDGET (Apple and Atari only) — Price: $34.95 Diskette
FAMILY BUDGET is a very convenient financial record-keeping program. You will be able to keep track of cash and credit expenditures as well as income on a daily basis. You can record tax deductible items and charitable donations. FAMILY BUDGET also provides a continuous record of all credit transactions. You can make daily cash and charge entries to any of 21 different expense accounts as well as to 5 payroll and tax accounts. Data are easily retrieved giving the user complete control over an otherwise complicated (and unorganized!) subject.

INTELINK (Atari only) — Price: $49.95 Diskette
This software package contains a menu-driven collection of programs for facilitating efficient two-way communications through a full duplex modem (required for use). In one mode of operation you may connect to a data service (e.g., The SOURCE or MicroNet) and quickly load data such as stock quotations onto your diskette for later viewing. This greatly reduces "connect time" and thus the service charge. You may also record the complete contents of a communications session. Additionally, programs written in BASIC, FORTRAN, etc. may be built off-line using the support text editor and later "uploaded" to another computer, making the Atari a very smart terminal. Even Atari BASIC programs may be uploaded. Further, a command file may be built off-line and used later as controlling input for a time-share system. That is, you can set up your sequence of time-share commands and programs, and the Atari will transmit them as needed; batch processing. All this adds up to saving both connect time and your time.

NEW
PAYFIVE (Apple II plus diskette, two drives required) — Price: $149.95
This is an enormously flexible employee payroll system with extraordinarily good human engineering features. PAYFIVE prints checks and computes the required federal, state and local forms for up to 148 employees! The pay methods may be hourly, salary, commission or any combination. There are multiple options for pay periods, and they also can be used in any combination. PAYFIVE includes many other features and comes extremely well documented with a 200 page manual. The manual may be purchased separately for $30 and that payment later applied to the software purchase.

SHOPPING LIST (Atari only) — Price: $12.95 Cassette/$16.95 Diskette
SHOPPING LIST stores information on items you purchase at the supermarket. Before going shopping, it will remind you of all the things you might need, and then display (or optionally print) your shopping list and the total cost. Adding, deleting, changing and storing data is very easy. Runs with 16K.

NEW
TAX OPTIMIZER (Available for all computers) — Price: $59.95 Diskette
The TAX OPTIMIZER is an easy-to-use, menu oriented software package which provides a convenient means for analyzing various income tax strategies. The program is designed to provide a quick and easy data entry. Income tax is computed by all tax methods (regular, income averaging, maximum and alternate minimum tax). The user may immediately observe the tax effect of critical financial decisions. TAX OPTIMIZER has been thoroughly field tested in CPA offices and comes complete with the current tax tables in its data files. TAX OPTIMIZER is tax deductible!

NEW
UTIL (Apple only, 48K) — Price: $19.95 Diskette
UTIL is a disk-oriented utility system which permits examining and changing of the contents of DOS 3.2 and 3.3 diskettes at the bit (nibble or byte) level. With UTIL you can easily examine the contents of a diskette sector by sector, restructure the sector pointers, reallocate sectors (e.g. bad sectors may be "hidden"), and perform many other sophisticated operations. For the experienced programmer.

NEW
TURNKEY AND MENU (Atari only) — Price: $17.95 Diskette
TURNKEY is a utility program which allows you to create autoboot/autorun diskettes easily. Simply load and run TURNKEY, load the program diskette to be modified, and answer the questions! The TURNKEY diskette also comes with DOS 2.0 and includes another program, MENU. MENU lists the contents of your diskette alphabetically, and permits the running of any BASIC program on the diskette by typing a single key. TURNKEY and MENU provide you with the ability to run any program on your diskette by simply turning on the computer and pressing a single key.

NEW
STOCKAID (Atari only) — Price: $29.95 Diskette
STOCKAID provides a powerful set of tools for stock market analysis. With STOCKAID you can display point and figure charts, as well as bar charts with oscillators. You can also examine long term moving averages and on-balance volume features. STOCKAID allows you to input daily data with a single diskette storage capability of 239 days × 16 stocks. Included are stock dividend and split adjustment capabilities. A very professional package!

NEW
SHAPE MAGICIAN (Apple II, 48K, diskette only) — Price: $29.95
At last! An utility for painlessly creating graphics shapes for the Apple. Create, edit and save up to 30 shapes which can then be used to develop arcade games or to simply enhance your programs. Add that professional touch!

EDUCATION

HODGE PODGE (Apple 48K and Atari diskette only) — Price: $14.95 Cassette/$18.95 Diskette
Let HODGE PODGE be your child's teacher. Pressing any key on your Apple will result in a different and intriguing "happening" related to the letter or number of the chosen key. The program's graphics, color and sound are a delight for children from ages 1½ to 7. HODGE PODGE is a non-intimidating teaching device which brings a new dimension to the use of computers in education. See the excellent reviews of this very popular program in INFOWORLD and SOFTALK.

NEW
TEACHERS' AIDE (Atari only) — Price: $13.95 Cassette/$17.95 Diskette
TEACHERS' AIDE consists of three basic modules contained in one program. The first module provides addition and subtraction exercises of varying levels of difficulty. The second module consists of multiplication problems in which the student may be tested both on the final answer and/or on the subtotal answers in the long hand procedure. Several levels of complexity are provided here as well. The third module consists of division problems, one particularly nice feature of the division module is that the long hand division steps can be displayed along with the remainder in order to clearly demonstrate the procedure by which the remainder is derived. Using TEACHER'S AIDE is not merely a drill, but rather a learning experience.

NEW
PHARMACOLOGY UPDATE (PET only) — Price: $169.95 Cassette/$149.95 Diskette
This is DYNACOMP's first educational software entry for the medical profession (more are coming!). PHARMACOLOGY UPDATE was written by a R.N. as a masters project, with the aid of a practicing pharmacologist and an electronics instructor. This package comes in two parts. The first part is a 200 page manual which is divided into 11 sections. Each of these sections provides both concise information and probing questions. The second part consists of 11 programs that are keyed to the text and which test the degree of your understanding of the text material. This package has great educational value for the beginning student as well as the professional interested in an efficient way to review and update his or her knowledge. Available on cassettes (11) or diskette.

NEW
TEACHER'S GRADEBOOK (Apple 48K dual/single drive) — Price: $49.95 Diskette
TEACHER'S GRADEBOOK is a complete password protected record-keeping system for the classroom. It supports up to nine users, and each user may have data for up to nine classes on one disk (with up to 90 students per class). Typical information which can be entered, edited and processed includes rosters, absences and grades. Summary reports may be displayed (on the screen) or printed in various ways, with automatic weighted averaging and conversion to letter grades. This system has been tested ("goof-proofed") in the class environment and is both well-written and well-documented.

STATISTICS and ENGINEERING

DIGITAL FILTER (Available for all computers) — Price: $39.95 Cassette/$43.95 Diskette
DIGITAL FILTER is a comprehensive data processing program which permits the user to design his own filter function or choose from a menu of filter forms. In the explicit design mode the shape of the frequency transfer function is specified by directly entering points along the desired filter curve. In the menu mode, ideal low pass, high pass and bandpass filters may be approximated to varying degrees according to the number of points used in the calculation. These filters may optionally also be smoothed with a Hanning function. In addition, multi-stage Butterworth filters may be selected. Features of DIGITAL FILTER include plotting of the data before and after filtering, as well as display of the chosen filter functions. Also included are convenient data storage, retrieval and editing procedures.

DATA SMOOTHER (Not available for Atari) — Price: $19.95 Cassette/$23.95 Diskette
This special data smoothing program may be used to rapidly derive useful information from noisy business and engineering data which are equally spaced. The software features choice in degree and range of fit, as well as smoothed first and second derivative calculation. Also included is automatic plotting of the input data and smoothed results.

FOURIER ANALYZER (Available for all computers) — Price: $19.95 Cassette/$23.95 Diskette
Use this program to examine the frequency spectra of limited duration signals. The program features automatic scaling and plotting of the input data and results. Practical applications include the analysis of complicated patterns in such fields as electronics, communications and business.

TFA (Transfer Function Analyzer) — Price: $19.95 Cassette/$23.95 Diskette
This is a special software package which may be used to evaluate the transfer functions of systems such as hi-fi amplifiers and filters by examining their response to pulsed inputs. TFA is a major modification of FOURIER ANALYZER and contains an engineering-oriented decibel versus log-frequency plot as well as data editing features. Whereas FOURIER ANALYZER is designed for educational and scientific use, TFA is an engineering tool. Available for all computers.

HARMONIC ANALYZER (Available for all computers) — Price: $24.95 Cassette/$28.95 Diskette
HARMONIC ANALYZER was designed for the spectrum analysis of repetitive waveforms. Features include data file generation, editing and storage/retrieval as well as data and spectrum plotting. One particularly unique facility is that the input data need not be equally spaced or in order. The original data is sorted and a cubic spline interpolation is used to create the data file required by the FFT algorithm.

FOURIER ANALYZER, TFA and HARMONIC ANALYZER may be purchased together for a combined price of $51.95 (three cassettes) and $63.95 (three diskettes).

REGRESSION I (Available for all computers) — Price: $19.95 Cassette/$23.95 Diskette
REGRESSION I is a unique and exceptionally versatile one-dimensional least squares "polynomial" curve fitting program. Features include very high accuracy; an automatic degree determination option; an extensive internal library of fitting functions, data editing; automatic data, curve and residual plotting, a statistical analysis (eg: standard deviation, correlation coefficient, etc.) and much more. In addition, new fits may be tried without reentering the data. REGRESSION I is certainly the cornerstone program in any data analysis software library.

REGRESSION II (Available for all computers) — Price: $19.95 Cassette/$23.95 Diskette
PARAFIT is designed to handle those cases in which the parameters are imbedded (possibly nonlinearly) in the fitting function. The user simply inserts the functional form, including the parameters (A(1), A(2), etc.) as one or more BASIC statement lines. Data, results and residuals may be manipulated and plotted as with REGRESSION I. Use REGRESSION I for polynomial fitting, and PARAFIT for those complicated functions.

MULTILINEAR REGRESSION (MLR) (Available for all computers) — Price: $24.95 Cassette/$28.95 Diskette
MLR is a professional software package for analyzing data sets containing two or more linearly independent variables. Besides performing the basic regression calculation, this program also provides easy to use data entry, storage, retrieval and editing functions. In addition, the user may interrogate the solution by supplying values for the independent variables. The number of variables and data size is limited only by the available memory.

REGRESSION I, II and MULTILINEAR REGRESSION may be purchased together for $51.95 (three cassettes) or $63.95 (three diskettes).

ANOVA (Not available on Atari cassette or for PET/CBM) — Price: $39.95 Cassette/$43.95 Diskette
In the past the ANOVA (analysis of variance) procedure has been limited to the large mainframe computers. Now DYNACOMP has brought the power of this method to small systems. For those conversant with ANOVA, the DYNACOMP software package includes the 1-way, 2-way and N-way procedures. Also provided are the Yates 2^{K-P} factorial designs. For those unfamiliar with ANOVA, do not worry. The accompanying documentation was written in a tutorial fashion (by a professor in the subject) and serves as an excellent introduction to the subject. Accompanying ANOVA is a support program for building the data base. Included are several convenient features including data editing, deleting and appending.

BASIC SCIENTIFIC SUBROUTINES, Volumes 1 and 2 (Not available for Atari)
DYNACOMP is the exclusive distributor for the software keyed to the popular texts *BASIC SCIENTIFIC SUBROUTINES, Volumes 1 and 2* by F. Ruckdeschel (see advertisements in BYTE magazine). These subroutines have been assembled according to chapter. Included with each collection is a menu program which selects and demonstrates each subroutine.

Volume 1
- Collection #1. Chapters 2 and 3 - Data and function plotting, complex variables and functions
- Collection #2. Chapter 4 - Extended matrix and vector operations
- Collection #3. Chapters 5 and 6 - Random number generators (Poisson, Gaussian, etc.), series approximations

Price per collection: $16.95 Cassette/$20.95 Diskette
All three collections are available for $44.95 (three cassettes) and $53.95 (three diskettes).

Volume 2
- Collection #1. Chapter 1 - Linear, polynomial, multidimensional, parametric least squares
- Collection #2. Chapter 2 - Series approximation techniques (economization, inversion, reversion, shifting, etc.).
- Collection #3. Chapter 3 - Functional approximations by iteration and recursion
- Collection #4. Chapter 4 - CORDIC approximations to trigonometric, hyperbolic, exponential and logarithmic functions.
- Collection #5. Chapter 5 - Table interpolation, differentiation and integration (Newton, LaGrange, splines)
- Collection #6. Chapter 6 - Methods for finding the real roots of functions
- Collection #7. Chapter 7 - Methods for finding the complex roots of functions
- Collection #8. Chapter 8 - Optimization by steepest descent

Price per collection: $14.95 Cassette/$18.95 Diskette
All eight collections are available for $99.95 (eight cassettes) and $129.95 (eight diskettes).
Because the texts are a vital part of the documentation, *BASIC SCIENTIFIC SUBROUTINES, Volumes 1 and 2* are available from DYNACOMP:

BASIC SCIENTIFIC SUBROUTINES, Vol 1 (319 pages): $19.95 + 75¢ postage
BASIC SCIENTIFIC SUBROUTINES, Vol 2 (790 pages): $23.95 + $1.50 postage

See reviews in KILOBAUD and Dr. Dobbs.

NEW
SOFTNET (Apple II, 48K, diskette only) — Price: $129.95
SOFTNET may be used to create models of liquid pipeline systems to evaluate their flow performance. Up to 150 nodes with up to 150 connecting elements may be simulated, and models may be combined to form yet larger models. If you are involved in water distribution systems, chemical fluid flow problems, building plumbing, or similar situations, this is an ideal analysis tool.

NEW
MATCHNET (TRS-80 only) — Price: $19.95 Cassette/$23.95 Diskette
It often takes days to iteratively optimize an L, Pi or T matching network for a particular application. Take a few minutes with MATCHNET and you will have the Q, frequency response and reflection coefficients for any of twelve matching networks. You input the source and load impedance and MATCHNET calculates the R, C, (and L) values and plots (and/or prints) the frequency response and reflection coefficients for each configuration. The reviewer of this program remembers when he use to do this by hand and loves MATCHNET!

ACTIVE CIRCUIT ANALYSIS (ACAP) (48K Apple only) — Price: $35.95 Cassette/$39.95 Diskette
With ACAP you may analyze the response of an active or passive component circuit. The circuit may be probed at equal steps in frequency, and the resulting complex voltages at each component juncture examined; the frequency response of a filter or amplifier may be completely determined with respect to both amplitude and phase. In addition, ACAP prints a statistical analysis of the range of voltage responses which result from tolerance variations in the components. ACAP is easy to learn and use. Circuit descriptions may be saved onto cassette or diskette to be recalled at a later time for execution or editing. ACAP should be part of every circuit designer's program library.

LOGIC SIMULATOR (Apple only; 48K RAM) — Price: $35.95 Cassette/$39.95 Diskette
Test your complicated digital logic design with respect to given set of inputs to determine how well the circuit will operate. The elements which may be simulated include multiple input AND, OR, NOR, EXOR, EXNOR and NAND gates, as well as inverters, J-K and D flip-flops, and one-shots. Inputs may be clocked in with varying clock cycle lengths/displacements and delays may be introduced to probe for glitches and race conditions. A timing diagram for any given set of nodes may be plotted using HIRES graphics. Save your breadboarding until the circuit is checked by LOGIC SIMULATOR.

NEW
NUMBERKRUNCHER (TRS-80 only) — Price:69.95 Cassette/$73.95 Diskette
This program is the most complete numerical analysis system available for the TRS-80. It can handle up to 255 data sets, each set having a six character name. It includes complete data editing facilities and convenient data input/output capability. The analyses available are multiple linear regression and correlation determination of residuals, data transformations and extensive graphics generation, including axis naming, and more. The supporting documentation is extremely well written and well organized, and includes appendices which describe the numerical procedures used in the program.

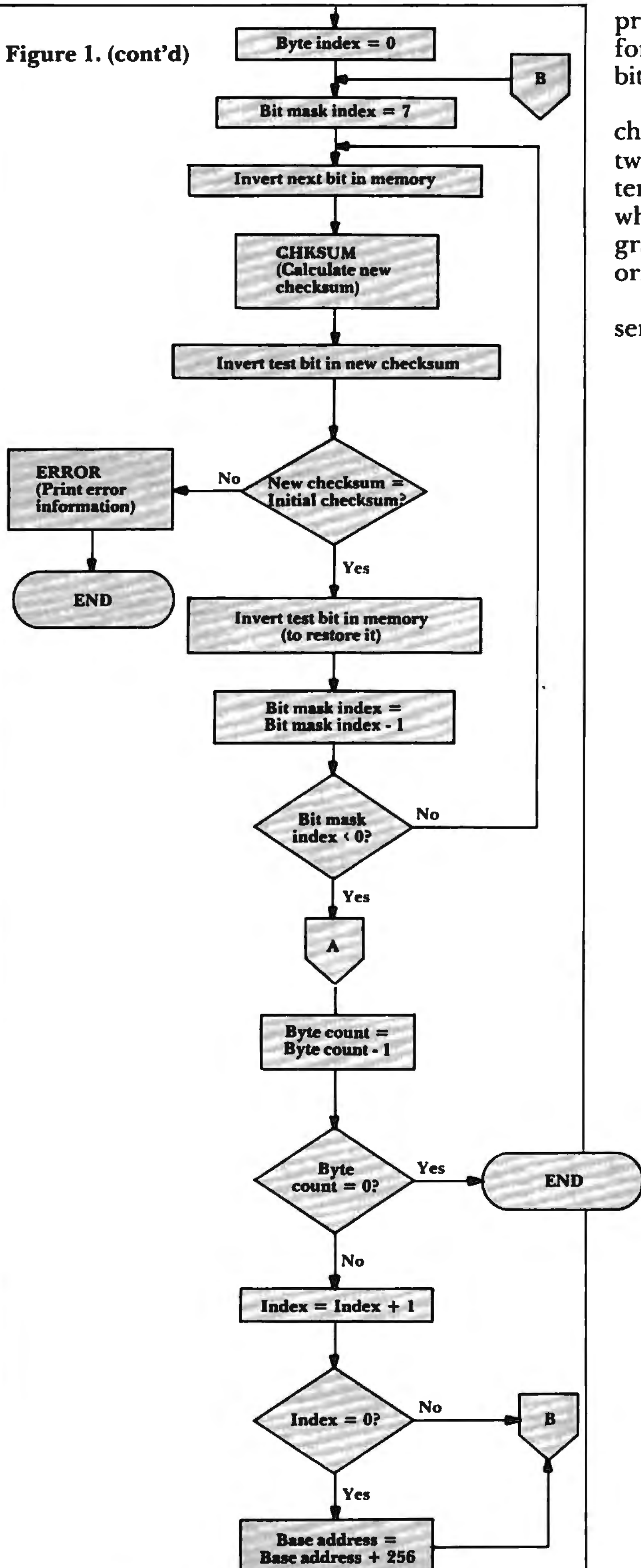
NEW
STATSORT (TRS-80 only) — Price: $39.95 Cassette/$43.95 Diskette
STATSORT consists of several menu selected programs which allow the user to create (build, edit, merge), format and print files, (machine) sort them on any field, and numerically analyze (maximum, minimum, average, variance, standard deviation) tabulated data. STATSORT is well documented and easy to use. The cassette version can also be employed to create a data type which can be read by the Radio Shack Advanced Statistical Package.

NEW
STATTEST (Not available on Atari cassette or for PET/CBM) — Price: $19.95 Cassette/$23.95 Diskette
This is a statistical inference package which helps you make wise decisions in the face of uncertainty. In an interactive fashion you can build and edit data files and test the differences in means, variances and proportions. STATTEST will also perform data analysis as well as do linear correlation and regression. This menu-directed statistical workhorse is rounded out with a chi square contingency test and a (uniform and normal) random sample generator. The documentation is written by a college professor who guides you through the various tests.

ABOUT DYNACOMP

DYNACOMP is a leading distributor of small system software with sales spanning the world (currently in excess of 50 countries). During the past three years we have greatly enlarged the DYNACOMP product line, but have maintained and improved our high level of quality and customer support. The achievement in quality is apparent from our many repeat customers and the software reviews in such publications as COMPUTRONICS, 80 Software Critique, A.N.A.L.O.G., Softalk, Creative Computing and Kilobaud. DYNACOMP software has also been choosen for demonstration on network television. Our customer support is as close as your phone. It is always friendly. The staff is highly trained and always willing to discuss products or give advice.

Figure 1. (cont'd)

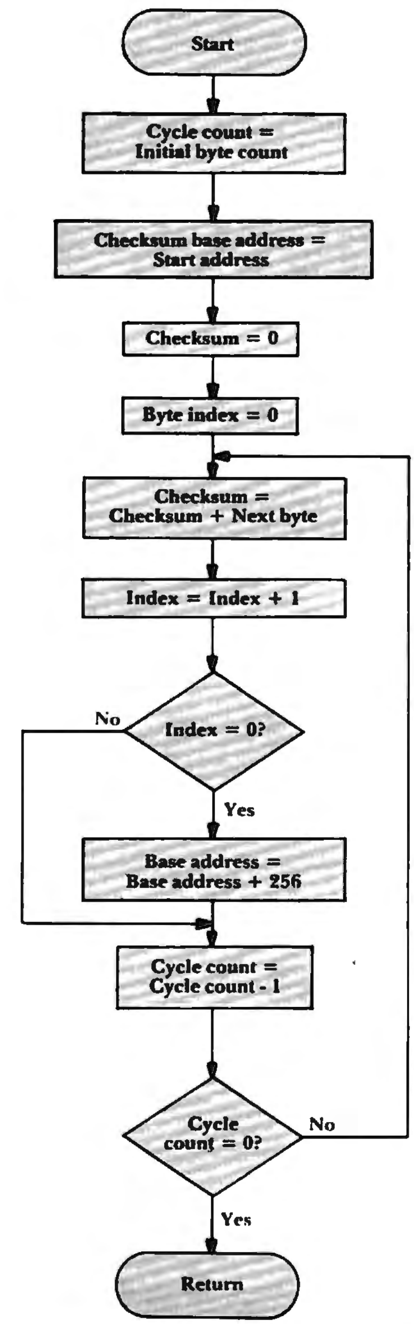


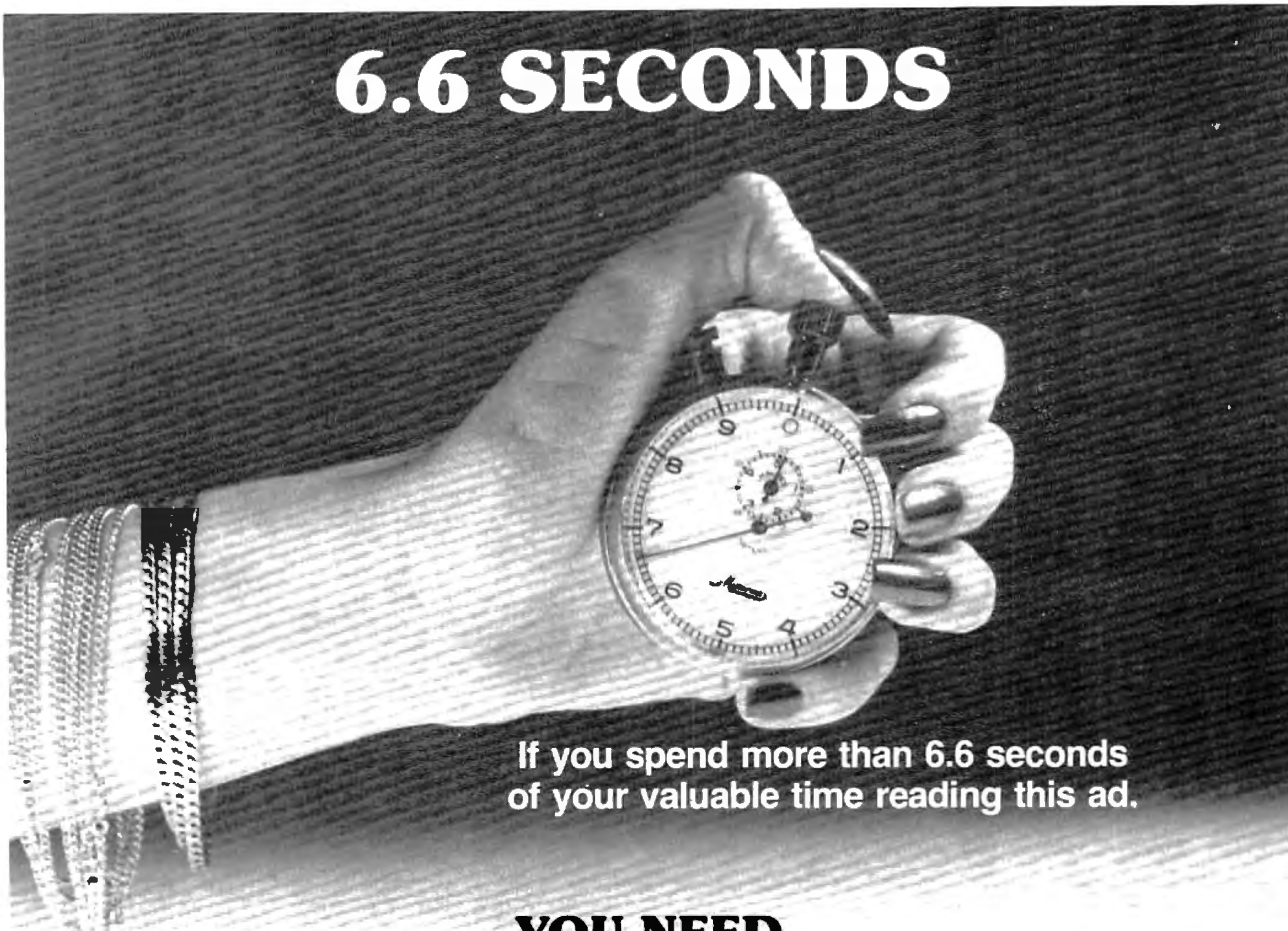
program calls CHKSUM again, to get the checksum for memory with one bit inverted, and inverts that bit position in the checksum.

This invert operation should make the new checksum identical to the initial checksum. If the two checksums are not identical, the program terminates by printing the bit position and address where the error was detected. Otherwise, the program reinverts the current test bit, to restore its original state.

The remainder of the program involves a series of three counter/index adjustment opera-

Figure 2: Checksum-Calculating Subroutine (CHKSUM)





YOU NEED

SpeedRead+

- Optimized Systems Software, the company that brought you BASIC A+ and OS/A+, proudly presents another *"PLUS"*.
- ***SpeedRead+*** is the *world's first* speed reading tutor designed for use on your personal computer.
- ***SpeedRead+*** *begins* with training your eyes and mind to function as the incredible precision machine they were meant to be.
- ***SpeedRead+*** goes beyond mere words — it trains you to recognize phrases and columns instantly — it exercises your peripheral vision — it increases your comprehension.
- ***SpeedRead+*** matches *your* pace — now and in the future.

AVAILABLE NOW!

For 16K ATARI® computers with disk, Introductory Price **$59.95**
Coming soon for APPLE II®. Inquire about availability of TRS-80® and cassette versions.

SEE YOUR DEALER TODAY!

Call or write for a descriptive brochure of ***SpeedRead+***™ and other fine OSS products, such as OS/A+, BASIC A+, and TINY C®.

Optimized Systems Software, Inc., 10379 Lansdale Ave., Cupertino, CA 95014, (408) 446-3099

tions, with each followed by a branch/no-branch decision. In the first of these operations, the bit mask index is decremented; if it is nonnegative, the program branches back to invert the next bit. Otherwise, the byte count is decremented; if all bytes have been tested, the program terminates, error free. Otherwise, the byte index is incremented. The byte index is eight bits long, and can hold values from 0 to 255 (decimal). If the incrementation caused the byte index to overflow to zero, the program increments the high order byte of the base address, then branches back to reinitialize the bit mask index. Otherwise, the branch takes place with no change to the base address.

Figure 2 shows the flowchart for the checksum subroutine, CHKSUM. This subroutine is called from two places in the program: (A) it is called at the beginning of the program, to calculate the initial checksum, and (B) it is called from within the main loop, to calculate a new checksum after a test bit has been inverted. This second source of call requires the subroutine to maintain its own, separate byte count and base address, so as not to disturb the current values of these parameters in the main program. In the flowchart, these "working" parameters are labeled *cycle count* and *checksum base address*, respectively.

To start, cycle count is set equal to initial byte count, checksum base address is set equal to test start address, and the checksum and byte index are initialized to zero. The rest of the subroutine is just one big loop. In this loop, the checksum is accumulated, byte by byte, with intervening index and cycle count adjustments. The loop is terminated when all bytes have been processed; that is, when cycle count has been decremented to zero.

The Test Program

Now that you understand the criteria of the program and its sequences, we can look at the program itself. Program 1 shows the source code for the nondestructive test program, which was flowcharted in Figure 1. Note that before executing the program, the starting address must be stored in locations 00 and 01 (00 holds low byte) and the ending address must be stored in locations 02 and 03 (02 holds low byte).

Besides these four locations, the program uses 13 other zero page locations, as working storage. These include six parameters that are used in the main program – initial byte count (IBYTES), byte count (BYTES), base address (BADDR), initial checksum (CSUM) and temporary storage for the X and Y registers (SAVEX and SAVEY), and two parameters that are used in the checksum subroutine, a working copy of the byte count (CYCLES) and a checksum base address (CBADDR). Of these parameters, only IBYTES and CSUM remain unchanged throughout the program; all six other parameters will change during execution.

Following these reserve equates come three equates that reference subroutines in the AIM 65 monitor: CRLOW initializes the display and printer to their START positions; NUMA prints the contents of the accumulator, as two ASCII digits; OUTPRI sends one character to the print buffer. Other 6502-based computers have equivalent subroutines.

The actual code that follows is straightforward, so you should have no problem following it if you studied the flowchart in Figure 1. Some readers may wonder why I chose to save X and Y in zero page (locations SAVEX and SAVEY), rather than on the stack, during the call to CHKSUM in the main loop. There are two reasons why this was done:

1. The instructions used to save X and Y in zero page execute eight cycles faster than those to save X and Y on the stack (12 cycles versus 20 cycles). If you consider that for each byte tested, CHKSUM is called eight times – once for each bit position – saving X and Y in zero page saves 64N microseconds for an N-byte test run.
2. We need to use the checksum contents of the accumulator upon return from CHKSUM, and a pull from the stack (PLA) always loads the stack information into the accumulator. If the 6502 had the instructions PHX, PHY, PLX and PLY, the stack would have been the likely place to hold X and Y, but unfortunately it has no such instructions.

Programmers may also be interested in the way the bit masks are accessed by the EOR BMASK,X instructions that follow the labels INVERT and NXTBIT. The bit mask table, BMASK (shown at the end of Program 2), is arranged by *ascending* bit position. That is, the mask for Bit 0 comes first, followed by the mask for Bit 1, and so on. However, this table is accessed in *descending* order; Bit 7 is tested first and Bit 0 is tested last. This allows us to initialize the bit mask index to 7 (LDX #7 at label IBMSK), then decrement this index until it goes negative. Otherwise, working with a descending table and an incrementing index, the program would have to include a CPX #8 instruction to make the done/not done branch decision. By using the ascending table and decrementing index approach we've eliminated that compare instruction. Since the CPX #8 instruction executes in just two cycles, the difference in approaches is not significant, but the backwards access is a handy gimmick for your programming bag of tricks.

THE MILLION DOLLAR WINNER WORDCRAFT ULTRA

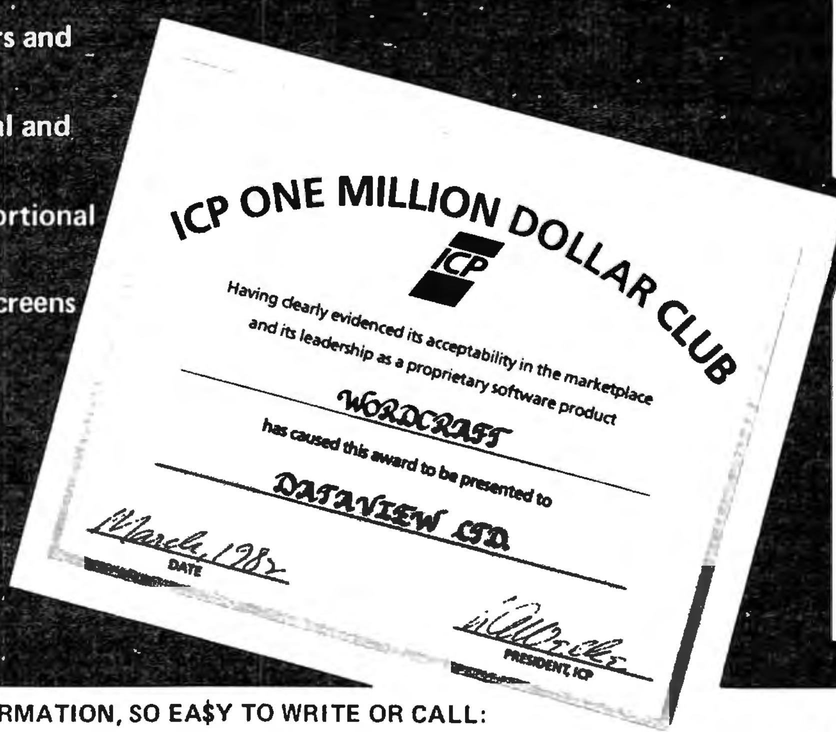
The word processing package that's EA$Y to use

Used on thousands of COMMODORE$, the micro computer that's EA$Y to operate

It's EA$Y to $ell and only sold through Dealers

The reasons it's 7 ways EA$Y

1. Screen layout matches printed document.
2. Designed for stenographers and secretaries to use.
3. Full choice over horizontal and vertical spacing.
4. True inter-character proportional spacing.
5. Multiple-user:— up to 8 screens can be connected.
6. Full text communication between Wordcraft users with automatic error recovery.
7. Comes complete with comprehensive manual & self-pace training guide.



FOR INFORMATION, SO EA$Y TO WRITE OR CALL:

WE$T COA$T: CIMARRON
666, Baker Street,
Suite 319
Costa Mesa, CA. 92626
Telephone: (714) 641 1156

EA$T COA$T: COMPUTER MARKETING $ERVICE$
300, W. Marlton Pike,
Cherry Hill, N.J. 08002
Telephone: (609) 795 9480

Program 2 shows the code for the checksum calculating subroutine, CHKSUM, which was flowcharted in Figure 2. It follows the flowchart closely, and needs no additional explanation. Program 2 also includes the previously mentioned bit mask table, BMASK, and the text for the error message.

This program will produce one of two messages. If the test memory is error free, the message *OKAY!* will be printed, otherwise an error message of the form *BIT n OF LOC. aaaa* will be printed. In the error message, the bit position and address that are printed identify the bit that was being tested when the checksum mismatch occurred. It's possible, of course, that inverting that bit actually caused some other bit in the memory to be inverted, due to crosstalk, so the printout position may not be the actual culprit. One way of finding out is to run a second test, starting at the location following the printout location; that is, rerun the test starting at "aaaa + 1."

Execution Times For The Test Program

As you can see from the listings, the program occupies slightly less than a page of memory; to be exact, it occupies 245 bytes. Of even greater significance, however, is the amount of time it takes to execute. That is, the amount of time it takes to test a selected portion of memory. In a test that I ran, the program took just over four minutes to check out a 1K portion of memory (1024 bytes).

At first I suspected that something was wrong with the program, but after a few calculations I became convinced that this was indeed a respectable time, in light of what the program was doing. First, consider that in a 1K byte test, the CHKSUM subroutine is called 8193 times; once to get the initial checksum, then once more for each of the 8192 bit positions in the 1024 byte test memory. The CHKSUM subroutine takes 28 + (29 x N) cycles to calculate the checksum for an N-byte memory, so it takes 29,724 cycles (microseconds) for a 1024 byte memory. Cranking out the math, we find that with

Exclusive-ORs And Checksums

An exclusive-OR is a logical operation in which two byte operands are combined to produce a result byte with these characteristics:

- For each bit position in which the operands are different (one is logic 0, the other is logic 1), the result will contain a logic 1.
- For each bit position in which the operands are the same (both logic 0 or both logic 1), the result will contain a logic 0.

These rules can be summarized as follows:

Bit Operand #1	Bit Operand #2	Result Bit
0	0	0
0	1	1
1	0	1
1	1	0

All of the popular 8-bit microprocessors have an exclusive-OR instruction. In the 6502, it has the mnemonic *EOR*. The EOR instruction operates on the contents of the accumulator with an immediate value or a value in memory, and leaves the result in the accumulator.

For example, if the accumulator contains the value $AB (where $ denotes hexadecimal) and location $40 contains the value $0F, the instruction *EOR $40* will produce a value of $A4 in the accumulator. The binary arithmetic looks like this:

```
  0000 1111   Contents of location $40=$0F
⊕ 1010 1011   Contents of accumulator =$AB
  1010 0100   Result in accumulator =$A4
```

Note what has happened here. The value $0F in location $40 has caused the four low order bits (0 through 3) to be *inverted,* but has left the four high order bits (4 through 7) intact.

This shows one of the primary uses for the EOR instruction: to invert some selected bits, but leave all other bits unchanged. In fact, the test program in this article uses the EOR instruction to invert a single bit in memory, by reading the appropriate memory byte into the accumulator, then exclusive-ORing it with a "mask" value that has just one bit set to logic 1. To invert Bit 7, the program applies a mask value of 10000000_2 ($80); to invert Bit 6, the program applies a mask value of 01000000_2 ($40); and so on.

The program in this article also uses a series of EOR instructions to calculate a *checksum* value. As mentioned in the article, the checksum is the exclusive-OR of all bytes being tested. For example, if locations $0400, $0401 and $0402 are being tested, the program will perform this type of operation:

```
  0010 1101   ($0400)=$2D
  1010 0011   ($0401)=$A3
⊕ 0001 1000   ($0402)=$18
  1001 0110   Checksum=$96
```

8193 calls, the program spends about 4.06 minutes in the CHKSUM subroutine!

Since the program is spending virtually all of its time in the CHKSUM subroutine, the total execution time of the program is directly dependent on the efficiency of this subroutine. If any readers have suggestions on how to streamline CHKSUM, I'd be happy to hear from them.

Program 1: Source Code for Nondestructive Test Program

```
ADDR   OBJECT     LABEL   SOURCE               PAGE 0001

2000              ; THIS PROGRAM PERFORMS A NONDESTRUCTIVE TEST
2000              ; ON RAM MEMORY, BY CALCULATING A SERIES OF CHECKSUMS.
2000              ; BEFORE EXECUTING, STORE THE STARTING ADDRESS
2000              ;  AT LOCS. 00 AND 01, AND THE ENDING ADDRESS
2000              ;  AT LOCS. 02 AND 03.
2000              ; IF THE TEST IS SUCCESSFUL, AN 'OKAY!' MESSAGE
2000              ; IS PRINTED.  OTHERWISE, THE BAD BIT POSITION
2000              ; AND ADDRESS ARE PRINTED.

2000              ; USER-SUPPLIED PARAMTERS

2000                      *=0
0000              START   *=*+2              ; STARTING ADDRESS
0002              END     *=*+2              ; ENDING ADDRESS

0004              ; EQUATES FOR WORKING STORAGE IN ZERO PAGE

0004              IBYTES  *=*+2              ; INITIAL BYTE COUNT
0006              BYTES   *=*+2              ; BYTE COUNT
0008              CYCLES  *=*+2              ; WORKING COPY OF BYTES
000A              BADDR   *=*+2              ; BASE ADDRESS
000C              CBADDR  *=*+2              ; BASE ADDRESS FOR CHECKSUM SUBR.
000E              CSUM    *=*+1              ; INITIAL CHECKSUM
000F              SAVEX   *=*+1              ; TEMP. STORAGE FOR X REGISTER
0010              SAVEY   *=*+1              ; TEMP. STORAGE FOR Y REGISTER

0011              ; AIM 65 MONITOR SUBROUTINES

0011              CRLOW   =$EA13             ; RESET DISPLAY & PRINTER
0011              NUMA    =$EA46             ; PRINT A, AS TWO ASCII CHARS.
0011              OUTPRI  =$F000             ; OUTPUT A TO PRINT BUFFER

0011                      *=$200
0200   38                 SEC                ; BYTE COUNT = END ADDR. - START
                                               ADDR. + 1
0201   A5 02              LDA END
0203   E5 00              SBC START
0205   85 04              STA IBYTES
0207   85 06              STA BYTES
0209   A5 03              LDA END+1
020B   E5 01              SBC START+1
020D   85 05              STA IBYTES+1
020F   85 07              STA BYTES+1
0211   E6 04              INC IBYTES
0213   E6 06              INC BYTES
0215   D0 04              BNE GETSUM
0217   E6 05              INC IBYTES+1
0219   E6 07              INC BYTES+1
021B   20 B1 02   GETSUM  JSR CHKSUM         ; CALCULATE INITIAL CHECKSUM
021E   85 0E              STA CSUM           ;  AND SAVE IT IN MEMORY
0220   A5 00              LDA START          ; BASE ADDRESS = START ADDRESS
0222   85 0A              STA BADDR
0224   A5 01              LDA START+1
0226   85 0B              STA BADDR+1
0228   A0 00              LDY #0             ; BYTE INDEX = 0
022A   A2 07      IBMSK   LDX #7             ; BIT MASK INDEX = 7
```

```
01-0560  022C  B1 0A     INVERT LDA (BADDR),Y     ; INVERT NEXT BIT IN MEMORY
01-0570  022E  5D DF 02         EOR BMASK,X
01-0580  0231  91 0A            STA (BADDR),Y
01-0590  0233  86 0F            STX SAVEX         ; SAVE X AND Y IN MEMORY
01-0600  0235  84 10            STY SAVEY
01-0610  0237  20 B1 02         JSR CHKSUM        ; CALCULATE NEW CHECKSUM
01-0620  023A  A6 0F            LDX SAVEX         ; RETRIEVE X AND Y
01-0630  023C  A4 10            LDY SAVEY
01-0640  023E  5D DF 02         EOR BMASK,X       ; INVERT TEST BIT IN NEW CHECKSUM
01-0650  0241  C5 0E            CMP CSUM          ; NEW CHECKSUM = INITIAL CHECKSUM?
01-0660  0243  D0 39            BNE ERROR         ;  NO.  PRINT ERROR INFO.
01-0670  0245  B1 0A     NXTBIT LDA (BADDR),Y     ;  YES.  INVERT TEST BIT IN MEMORY
01-0680  0247  5D DF 02         EOR BMASK,X
01-0690  024A  91 0A            STA (BADDR),Y
01-0700  024C  CA               DEX               ;  NO.  DECREMENT BIT MASK INDEX
01-0710  024D  10 DD            BPL INVERT        ; BIT MASK INDEX NEGATIVE?
01-0720  024F  C6 06            DEC BYTES         ;  YES.  DECREMENT BYTE COUNT
01-0730  0251  E4 06            CPX BYTES
01-0740  0253  D0 02            BNE BCNTO
01-0750  0255  C6 07            DEC BYTES+1
01-0760  0257  A6 06     BCNTO  LDX BYTES         ; BYTE COUNT = 0?
01-0770  0259  D0 1B            BNE INCIDX
01-0780  025B  A6 07            LDX BYTES+1
01-0790  025D  D0 17            BNE INCIDX
01-0800  025F  A0 00            LDY #0            ;  YES.  ALL DONE, WITH NO ERRORS
01-0810  0261  B9 70 02  OKLOOP LDA OKMSG,Y
01-0820  0264  20 00 F0         JSR OUTPRI
01-0830  0267  C8               INY
01-0840  0268  C0 06            CPY #6
01-0850  026A  D0 F5            BNE OKLOOP
01-0860  026C  20 13 EA         JSR CRLOW
01-0870  026F  00               BRK

01-0890  0270  20 4F     OKMSG  .BYT ' OKAY!'

01-0910  0276  C8        INCIDX INY               ;  NO.  INCREMENT BYTE INDEX
01-0920  0277  D0 B1            BNE IBMSK         ; BYTE INDEX=0?
01-0930  0279  E6 0B            INC BADDR+1       ;  YES.  ADD 256 TO BASE ADDRESS
01-0940  027B  4C 2A 02         JMP IBMSK
01-0950  027E

01-0970  027E            ; THIS ROUTINE PRINTS OUT THE BIT POSITION
01-0980  027E            ; AND ADDRESS AT WHICH THE MISMATCH OCCURRED

01-1000  027E  20 13 EA  ERROR  JSR CRLOW         ; RESET DISPLAY & PRINTER
01-1010  0281  A0 00            LDY #0            ; PRINT FIRST PART OF TEXT
01-1020  0283  B9 E7 02  LOOP1  LDA EMSG,Y
01-1030  0286  20 00 F0         JSR OUTPRI
01-1040  0289  C8               INY
01-1050  028A  C0 05            CPY #5
01-1060  028C  D0 F5            BNE LOOP1
01-1070  028E  8A               TXA               ; PRINT BIT PATTERN
01-1080  028F  09 30            ORA #$30
01-1090  0291  20 00 F0         JSR OUTPRI
01-1100  0294  B9 E7 02  LOOP2  LDA EMSG,Y        ; PRINT SECOND PART OF TEXT
01-1110  0297  20 00 F0         JSR OUTPRI
01-1120  029A  C8               INY
01-1130  029B  C0 0E            CPY #14
01-1140  029D  D0 F5            BNE LOOP2
01-1150  029F  A5 10            LDA SAVEY         ; ERROR ADDRESS = BASE ADDRESS +
                                                    INDEX
01-1160  02A1  18               CLC
01-1170  02A2  65 0A            ADC BADDR
01-1180  02A4  48               PHA
01-1190  02A5  A9 00            LDA #0
01-1200  02A7  65 0B            ADC BADDR+1
01-1210  02A9  20 46 EA         JSR NUMA          ; PRINT ERROR ADDRESS
```

```
01-1220  02AC  68                 PLA
01-1230  02AD  20 46 EA           JSR NUMA
01-1240  02B0  00                 BRK               ; RETURN TO MONITOR
```

Program 2: Source Code for CHKSUM Subroutine

```
 LINE#    ADDR   OBJECT     LABEL  SOURCE           PAGE 0004

01-1260   02B1              ; THIS SUBROUTINE ACCUMULATES THE CHECKSUM,
01-1270   02B1              ; BY EXCLUSIVE-ORING ALL BYTES

01-1290   02B1   A5 04      CHKSUM LDA IBYTES       ; CYCLE COUNT = BYTE COUNT
01-1300   02B3   85 08             STA CYCLES
01-1310   02B5   A5 05             LDA IBYTES+1
01-1320   02B7   85 09             STA CYCLES+1
01-1330   02B9   A5 00             LDA START        ; BASE ADDRESS = START ADDRESS
01-1340   02BB   85 0C             STA CBADDR
01-1350   02BD   A5 01             LDA START+1
01-1360   02BF   85 0D             STA CBADDR+1
01-1370   02C1   A9 00             LDA #0           ; CHECKSUM = 0
01-1380   02C3   A0 00             LDY #0           ; BYTE INDEX = 0
01-1390   02C5   51 0C      ACCUM  EOR (CBADDR),Y   ; CHECKSUM = CHECKSUM EOR NEXT BYTE
01-1400   02C7   C8                INY              ; INCREMENT INDEX
01-1410   02C8   D0 02             BNE DECCYC       ; INDEX = 0?
01-1420   02CA   E6 0D             INC CBADDR+1     ;  YES.  ADD 256 TO BASE ADDRESS
01-1430   02CC   A2 FF      DECCYC LDX #$FF         ;  NO.  DECREMENT CYCLE COUNT
01-1440   02CE   C6 08             DEC CYCLES
01-1450   02D0   E4 08             CPX CYCLES
01-1460   02D2   D0 02             BNE CYCZ
01-1470   02D4   C6 09             DEC CYCLES+1
01-1480   02D6   A6 08      CYCZ   LDX CYCLES       ; CYCLE COUNT = 0?
01-1490   02D8   D0 EB             BNE ACCUM        ;  NO.  GO PROCESS NEXT BYTE
01-1500   02DA   A6 09             LDX CYCLES+1
01-1510   02DC   D0 E7             BNE ACCUM
01-1520   02DE   60                RTS              ;  YES.  RETURN WITH CHECKSUM IN A


01-1550   02DF              ; MASKS USED TO INVERT BITS IN MEMORY

01-1570   02DF   01         BMASK  .BYT 1,2,4,8,$10,$20,$40,$80
01-1570   02E0   02
01-1570   02E1   04
01-1570   02E2   08
01-1570   02E3   10
01-1570   02E4   20
01-1570   02E5   40
01-1570   02E6   80

01-1590   02E7              ; ERROR MESSAGE TEXT

01-1610   02E7   20 42      EMSG   .BYT ' BIT '
01-1620   02EC   20 4F             .BYT ' OF LOC. '
01-1630   02F5                     .END

ERRORS = 0000

END OF ASSEMBLY = 02F4
:
```

©

A simple game illustrating a most sophisticated topic. The idea of recursive subroutines *is one of the more complex notions in programming – a subroutine which is its own subroutine. "Towers of Brahma," a child's game with pegs and disks, is used to illustrate how recursion works with versions here for PL/I and Microsoft (Commodore, Apple, OSI) and Atari BASICs.*

Recursive BASIC Subroutines

Earl Wuchter
Catasauqua, PA

Recursive *(adj): of, relating to, or constituting a procedure that can repeat itself indefinitely, or until a specified condition has been met.*
Subroutine *(n): a sequence of computer instructions for performing a specified task that can be used repeatedly in a program or in different programs.*
– *Webster's New Collegiate Dictionary*

Used together, these two words describe one of the most powerful, and most fascinating entities in programming, but it is a rare bird.

Recursive subroutines are most useful in compiled languages, but are seldom allowed. On the other hand, an interpreted language like BASIC can't help but allow a subroutine to reenter itself, but some serious problems must be overcome before the technique is of any use.

The problem with BASIC is that all variables are *global*. (When any variable is changed in a subroutine it is changed for the whole program.) To see what is missing in BASIC, look at a *call* statement and a subroutine statement in a typical compiled language. The variables in parentheses are called arguments or parameters:

```
CALL TRY (A,X)
SUBROUTINE TRY (A,Z)
```

When TRY is invoked, A and Z will take on the values of A and X from the calling routine. On return, A and X will receive the values of A and Z. Even though both routines use the variable A, the A in the calling routine will not be changed until a return is executed. The status (global or local) of other variables is normally up to the programmer.

BASIC subroutines do not have arguments. The normal programming practice is to have the calling routine assign values to subroutine variables before the GOSUB is executed. To prevent a subroutine from inadvertently affecting any other routine, we give it a set of local variables by using names that do not appear in any other routine.

A Solution to Passing BASIC Variables

Now you can see the problem with recursive code. How can we give a subroutine a set of variables that are different from those in the calling routine when they are one and the same routine?

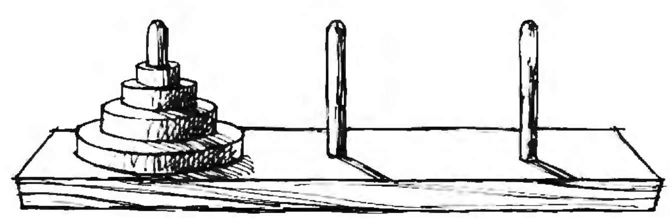
There is a solution. We can create an array of arguments and have each generation of the subroutine use its own "row" in the array.

The Towers Of Brahma

The "Towers of Brahma" is one problem that is best solved with a recursive subroutine. The solution, in turn, best illustrates the capability of recursive code. The two seem to have been made for each other.

This interesting mathematical problem is often found disguised as a child's toy: a board with three pegs and four or five disks of varying diameters that fit on the pegs. The disks are initially arranged on the first peg in order of their diameters, with the smallest one on top. They are to be moved one at a time, always keeping the smaller ones on top, until they are on the third peg.

Towers of Brahma



When the recursive solution to this problem is coded in the proper language, it is a thing of beauty. A PL/I version of the subroutine is shown below. This will be the model for a BASIC version. The main routine is not shown. The main routine simply inputs the number of disks and makes the initial call:

```
CALL MOVE (NDISKS,1,3);
```

Note: In this version of PL/I, arguments passed in as expressions become dummy arguments and do not change anything in the calling routine. Adding zero to a variable makes it an expression without altering its value. This feature provides for extremely compact code.

```
MOVE: PROCEDURE (N,F,T) RECURSIVE;
IF N=0 THEN RETURN;
CALL MOVE (N-1,F+0,6-F-T);
PUT SKIP DATA (N,F,T);
```

Brief Definitions

Some of the terms used in this article might be unfamiliar. Here's a short description of the main ideas:

CODE – Refers to the statements, the lines, or the entirety of a program, as in the phrase "there's an error in your code at line 110." Also used as a verb, as another word for programming: "I'm coding a subroutine."

COMPILED LANGUAGE – Unlike BASIC (an *interpreted language*) there are three steps to getting a program to run when you're working with a compiled language: 1. Write it. 2. Compile it. 3. Run it. After you write the program, a separate program called a *compiler* translates what you've written into a form of *machine language*. This results in a program which can then be executed (run).

Some examples of compiled languages are FORTRAN, COBOL, and PL/I. The language the program is written in (step 1, above) is called the *source language* and when you finish typing it you've got not surprisingly, the *source program*. Step 2 is where you instruct the compiler to "examine" and transform the source program into an *executable* (it can be executed) program called the *object* or *target* program.

You can write, easily read, and easily modify source programs, but you can't run them. Object programs, on the other hand, do run, but they are difficult to read (they're not written in words like PUT IF THEN DECLARE, but rather in machine or assembly language.)

INTERPRETED LANGUAGE – Step 2 (above) does not take place in an interpreted language. There are only two steps to writing an interpreted language program (BASIC is an example): 1. Write it. 2. RUN it. However, while the program is RUNning, a separate program in the computer is "interpreting" the meaning of words like PRINT and INPUT. In other words, the computer slows down somewhat while the interpreter decides what PRINT means and just what kind of PRINT is involved (to the screen, the printer, a disk file?). Then the interpreter sets up pointers and flags and variables and momentarily sends control to the machine language subroutine (in a library of such subroutines) which can do a PRINT.

All of this translating and interpreting is going on *during* the RUN of the program. It's obvious why interpreted programs tend to RUN more slowly than compiled programs. On the other hand, because you can write it/RUN it, it is easier to test parts of a program immediately and make the necessary changes to the program itself. Debugging is easier simply because the program you write is the same program that you will RUN.

PARAMETER PASSING – This generally refers to the ways that variables are made available to several programs at once, (or, if the variables are local, to several parts of the same program at once). For example, if the variable CASH "holds" 15.98, we might want to "pass" that value (15.98, the parameter) to another program. Some computers will cancel all variables when a new program is LOADed into RAM memory. How can the 15.98 be "passed" to the new program?

Likewise, if variables are local, the value of CASH will be different in subroutines from what is in the main body of the program unless 15.98 is "passed" to the subroutines.

Passing parameters is handled in different ways depending on which computer or language is being used. Compiled languages often allow you to put the variables in parentheses on the same line as a CALL to or RETURN from a subroutine. The example CALL SUB3 (CASH, TOTAL) *passes* the values of CASH and TOTAL to the subroutine SUB3. It can later pass variables back to the main routine in the same way, using parameters (also called *arguments*).

PL/I – A "high level" language (as opposed to lower level, closer to machine language) which combines attributes of FORTRAN and COBOL.

DUMMY ARGUMENTS – Parameters which have no effect other than to take up some necessary space. An example is the FRE(1) statement in Microsoft BASIC where the value within the parentheses can be anything. It's just there because the computer will not recognize FRE().

NESTING – When something is contained by something else, FOR I = 1 TO 10: FOR J = 1 TO 2: NEXT J: NEXT I. This J loop is said to be *nested* within the I loop.

STACK – The computer must be able to remember "return addresses." 100 GOSUB 500 will turn over control to whatever is on line 500, but eventually there will be a RETURN. So, before GOSUBbing, the "address" to RETURN to is put on the computer's *stack* and later pulled off the stack by RETURN. There is a limit to how many return addresses can be pushed on the stack. This is normally no problem since most subroutines go right back via RETURN, relieving the stack of the address number. Recursive, self-calling subroutines, however, aren't RETURNing. It's GOSUB-GOSUB-GOSUB, etc. Unless the recursion is carefully managed, the stack could quickly fill with return addresses and is then said to *overflow*.

CALL MOVE (N-1,6-F-T,T+0);
END MOVE;

The BASIC program has arrays for variables N, F, and T. Variable G serves as the index to the arrays. G represents the number of nested GOSUBs. The subroutine increments G on each entry and decrements it on each return, thereby keeping count of its own generation number. Arguments are passed to the next generation by putting values into the arrays at (G+1) before each GOSUB. The variable G is always the current subscript for any generation of the subroutine.

This method of using arrays requires that you know beforehand what the maximum nesting depth will be in order to dimension the arrays. This number is a function of the problem, but may be limited by the BASIC interpreter.

Solving 21 Disks Would Take More Than Eight Days

The program runs in less than 2K. The internal stack size of my PET restricts the number of disks to 21. When I try to run more than 21 I get an "OUT OF MEMORY" message. Your system may allow more or less, but don't expect to run this many disks to completion. The time required to complete 21 disks (at a rate of approximately three moves per second) will be more than eight days, and one more disk would double that.

The BASIC program here shows the moves to be made, along with a plot of the nesting level made down the left side of the screen. Each increase in the length of the plotted line indicates a GOSUB. Each decrease in length indicates a RETURN. You can see a more detailed view of the process by replacing the plot with a printout of the current array variables.

Program 1. Microsoft Version

```
10 REM TOWERS OF BRAHMA (RECURSIVE)
11 DIM N(22),F(22),T(22)
12 P$="----+----+----+----+--"
13 INPUT"N DISKS (1 TO 21) ";N(1)
14 IFN(1)<1 OR N(1)>22 THEN 13
15 F(1)=1
16 T(1)=3
17 GOSUB 31
18 END :::::::::::::::::::::::
31 G=G+1
32 PRINT LEFT$(P$,G)
33 IF N(G)=0 THEN 43
34 N(G+1)=N(G)-1
35 F(F+1)=F(F)
36 T(G+1)=6-F(G)-T(G)
37 GOSUB 31
38 PRINT TAB(19)"DISK#"N(G)"FROM"F(
   G)"TO"T(G)
39 N(G+1)=N(G)-1
40 F(G+1)=6-F(G)-T(G)
41 T(F+1)=T(F)
42 GOSUB 31
43 G=G-1
44 PRINT LEFT$(P$,G)
45 RETURN
```

Program 2. Atari Version

```
11 DIM N(22),F(22),T(22),P$(20)
12 P$="----+----+----+----+---"
13 ? "N DISKS (1 TO 21)";:INPUT T:N(1)=T

14 IF T<1 OR T>21 THEN 13
15 F(1)=1
16 T(1)=3
17 GOSUB 31
19 END
31 G=G+1
32 IF G THEN ? P$(1,G)
33 IF N(G)=0 THEN 43
34 N(G+1)=N(G)-1
35 F(G+1)=F(G)
36 T(G+1)=6-F(G)-T(G)
37 GOSUB 31
38 ? ,"DISK #";N(G);" FROM ";F(G);" TO "
;T(G)
39 N(G+1)=N(G)-1
40 F(G+1)=6-F(G)-T(G)
41 T(G+1)=T(G)
42 GOSUB 31
43 G=G-1
44 IF G THEN ? P$(1,G)
45 RETURN
```

©

Santa Cruz Educational Software * Tricky Tutorials * Santa Cruz Educational Software * Tricky Tutorials * Santa Cruz Educational Software * Tricky

ATARI (tm) GRAPHICS AND SOUND MADE EASY!

SANTA CRUZ EDUCATIONAL SOFTWARE HAS WRITTEN A SERIES OF AFFORDABLE PROGRAMS THAT DEMONSTRATES MANY OF THE SPECIAL "TRICKS" THAT ONLY THE ATARI 400/800 COMPUTER IS CAPABLE OF DOING. WE OFFER EVERYTHING FROM A PROGRAM THAT DOES FANTASTIC HIGH RESOLUTION GRAPHICS TO ONE THAT ACTUALLY DIALS YOUR PHONE.....

****

MASTER MEMORY MAP(tm) — This is really the key to using the ATARI'S capabilities. We start out by explaining how to PEEK and POKE values into memory so that even new programmers can use this. Then: we give you over 15 pages of the memory locations that are the most useful. The information is condensed from both the ATARI'S Operating System Manual and various articles and programs. It is, of course, useful even for experienced programmers as a reference. Also, we highly suggest that dealers offer this Memory Map to customers who request to be told how to use the power of the machine. We guarantee it will answer many of the questions you have about the machine.

$6.95

TRICKY TUTORIALS(tm)

#1: DISPLAY LISTS — This program teaches you how to alter the program in the ATARI that controls the format of the screen. For example: 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 the various modes on the screen at the same time. Just think how nice your programs could look with a mix of large and small text, and both high and low resolution graphics. this program has many examples plus does all of the difficult caluculations! **$19.95**

#2: HORIZONTAL/VERITCAL SCROLLING — The information you put on the screen, either graphics or text, can be moved up, down or sideways. This can make some nice effects. You could move only the text on the bottom half of the screen or perhaps create a map and then move smoothly over it by using the joystick **$19.95**

#3: PAGE FLIPPING — Normally you have to redraw the screen every time you change the picture or text. Now you can learn how to have the computer draw the next page you want to see while you are still looking at the previous page, 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 how complicated the picture, you could flip between many pages, thus allowing animation or other special effects with your text. **$19.95**

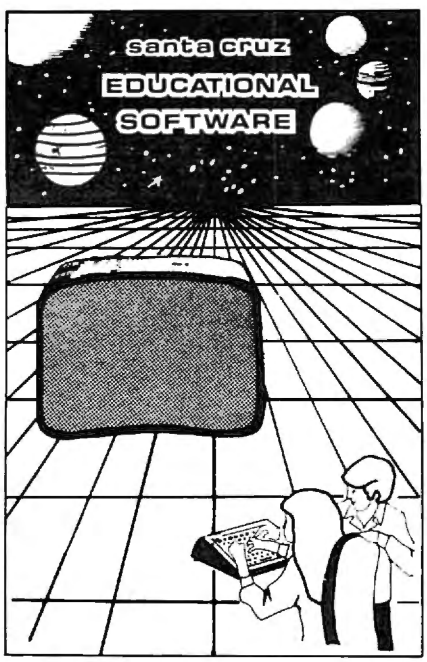
#4: BASICS OF ANIMATION — Shows you how to animate simple shapes using the PRINT and PLOT commands, and also has nice little PLAYER/MISSILE Graphics demo to learn. This would be an excellent way to start making your programs come alive on the screen. Recommended for new owners **$19.95**

#5: PLAYER MISSILE GRAPHICS — Learn to write your own animated games and business applications! The Tutorial begins with many small examples that complement the 50 page manual, then gradually builds up to a complete game as everything you need to know is explained fully. Also included are two machine language routines that you can use to animate Players from Basic. Next we include one of the best editors currently available, one for Playfield shapes, and one for Player shapes. All of the examples and games run with 16K Tape or 32K Disk. The two utilities require 32K to run. **$29.95**

#6: SOUND — From explaining how to create single notes, to demonstrating complex four channel sound effects, this newest tutorial is great. Even those experienced with ATARI's sound capabilities will find the menu of sound effects a needed reference that can be used whenever you are in the need of a special sound for your programs. Everyone will learn something new! Written by Jerry White. **$19.95**

Tricky Tutorials (except #5) require 16K memory for cassette orders and 24K for disk. The price is $19.95 each. Tricky Tutorials #1 to 6 in a notebook cost only $99.95, a $30 savings.

THE GRAPHICS MACHINE!! — Turn your computer into an incredible graphics tool with advanced commands like circle, box, fill, polygon, line, help, etc. 3 colors in graphics 8 with instant text!!! Create colorful business charts or beautiful drawings and then save or retrieve them from disk in 5 SECONDS. YES, it's that fast. Needs all 48K, disk, and costs **$19.95**



MINI-WORD PROCESSOR — This is for those of you who have a printer, but don't want to spend $100 or more for a fancy word processor. It is suitable for simple editing of text, accepts most control characters for your printer, and text is stored on disk for easy retrieval. Holds 2½ typed pages at a time. Requires 32K, disk or tape. **$19.95**

BOB'S BUSINESS — 14 small business type programs for home or office, all chosen from a nice menu. Supports printed output. 169 sectors of output require 16K tape, or 32K disk. **$14.95**

KID'S #1 — Includes the following: 1) TREASURE — search for the lost treasure while trying to keep from falling into the sea. Nice graphics if you find it!; 2)DIALOGUE — talk back to the computer about four subjects; 3) MATH QUIZ — Nice musical and graphical rewards for good scores. Parents input the level of difficulty.

KID'S #2 — A spelling quiz, a "scrabble" type game, and a version of Touch with the computer giving all the directions! Both Kid's programs require 16K tape or 24K disk and cost **$14.95** each.

MINI-DATABASE/DIALER — This unique new program stores and edits up to 8 lines of information such as name, address, and phone numbers, or messages, inventories or anything you want. It has the usual sort, search, and print options, but it also has an unusual feature: If your files include phone numbers and you have a touch-tone phone, the program will DIAL THE PHONE NUMBERS FOR YOU! This is perfect for those who make a lot of calls like salesmen, teens, or those trying to get through to busy numbers (acts as an auto-redialer). It is also a lot of fun to use. Requires 16K cassette or 24K disk and costs **$24.95**

FONETONE — For those who only want to store name and phone numbers and have the dialer feature as above, we offer this reduced version. Same memory requirements, but only costs **$14.95**. Don't forget you must have a touch-tone phone.

PLAYER PIANO — Turns your keyboard into a mini-piano and more. Multiple menu options provide the ability to create your own songs, save or load data files using cassette or diskette, fix or change any of up to 400 notes in memory, and play all or part of a song. The screen displays the keyboard and indicates each key as it is played from a data file or the notes you type. You don't have to be a musician to enjoy this educational and entertaining program. Requires 24K cassette or 32K disk. **$14.95**

BOWLERS DATABASE — Provides the league bowler with the ability to record and retrieve bowling scores providing permanent records. The data may then be analyzed by the program and displayed or printed in summary or detail form. Data may be stored on cassette or diskette and updated quickly and efficiently. The program proivdes such information as highest and lowest scores by individual game, (first, second, and third games throughout the season), high and low series, current average, and more. The program listing and documentation provided are a tutorial on ATARI basic and record keeeping. Requires 16K for cassette or 24K for disk. **$14.95**

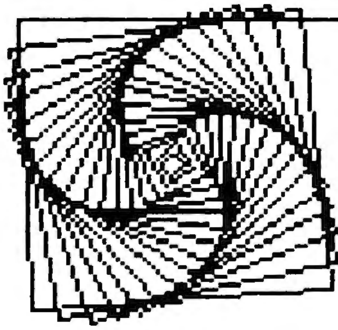
#7: DISK UTILITIES TUTORIAL — This Tutorial both offers you several programs to aid in the use of your 810 Disk drive, and teaches you how these utilities were written at the same time. The utilities included are a Menu program, a Disk inspector (look at sectors on the disk), a Disk Directory printout program, a Format utility to quickly format large numbers of disks, and a utility that allows your programs to start running by just turning on the 810 (Autorun.sys). Requires 32K of memory and costs only **$29.95**

****

SEE YOUR LOCAL DEALER
OR
ORDER FROM:

SANTA CRUZ EDUCATIONAL SOFTWARE
5425 Jigger Dr., Soquel, CA 95073
C.O.D. Orders call (408) 476-4901
***Include $3.00 postage ($1.00 for Memory Map)**
***In California, include 6.5% tax**
Visa and MasterCard Accepted

Santa Cruz Educational Software * Tricky Tutorials * Santa Cruz Educational Software * Tricky Tutorials * Santa Cruz Educational Software * Tricky



A Monthly Column

Friends Of The Turtle

David D. Thornburg
Associate Editor

A Fractal Of My Former Self...

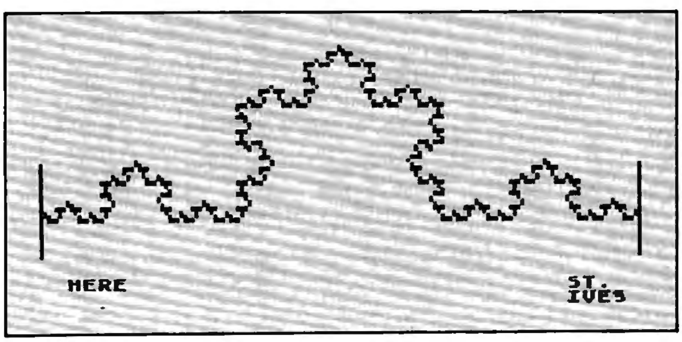
As I was going to St. Ives,
I met a man with seven wives.
Each wife carried seven sacks;
And in each sack was seven cats;
And with each cat was seven kits.
Kits, cats, sacks, and wives,
How many were going to St. Ives?

If you remember this puzzle from your childhood you probably remember that the answer is one.

Well, that may not be very exciting; but suppose you are walking to St. Ives and you want to know how far it is from some other location – Here, for example. The first thing we might do is look at a map.

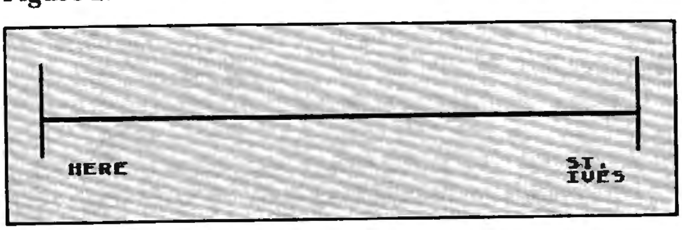
From the map we see that St. Ives and Here are both coast towns located near bays. In between, the coast juts out to sea.

Figure 1.



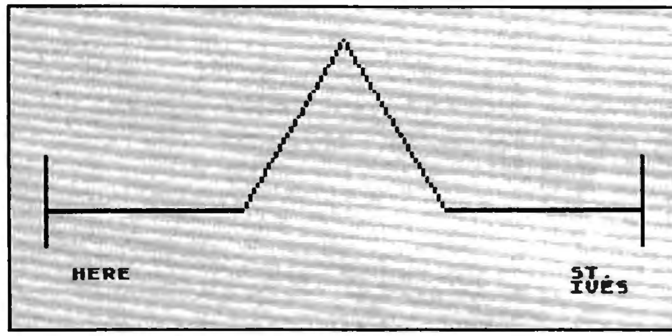
How long is the coastline between the towns of Here and St. Ives? If we set a pair of dividers to span the distance between the two towns we get one measurement. This distance is one kilometer.

Figure 2.



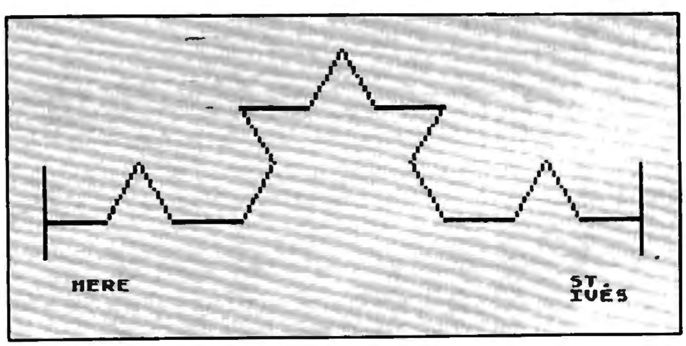
But this path doesn't take us along the coast. If we set our dividers to measure in units of 1/3 of a kilometer, we can trace a path which more closely follows the coastline.

Figure 3.



By counting each 1/3 kilometer increment we can see that this path is 4/3 kilometer in length. Next, let's set the dividers to 1/9 of a kilometer and measure the distance again.

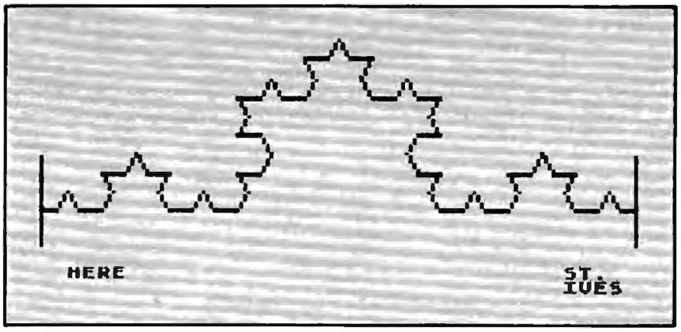
Figure 4.



This looks more like the actual coastline and gives us a distance of 16/9 kilometers. Since we replaced each straight line in Figure 3 with a replica of Figure 3, we can see that the length of the coast in Figure 4 can be written as 4/3 * 4/3.

Now let's set the dividers to 1/27 of a kilometer and measure again.

Figure 5.



This brings us even closer to the coastline and gives us a distance of 4/3 * 4/3 * 4/3 or 64/27 kilometers – more than twice our original estimate.

By now it should be obvious that each time we reduce the setting of our dividers by 1/3, the path length increases by a factor of 4/3. If we were to walk along the coast by following every nook and cranny, the distance would be 4/3 * 4/3 * 4/3 * 4/3 * 4/3,... etc. In other words, the length is infinite.

The reason for this is that each part of the coastline is an infinite set of replicas of the overall shape shown in Figure 3: If you were to look at a high magnification view of the coastline it would be just as bumpy as a lower magnification view.

Real coastlines don't have infinite lengths, of course, but our imaginary coastline does because it is made from a type of self-similar curve called a *fractal.* Mathematical expressions of this type have been known for some time but were not studied much because mathematicians thought their properties were too strange. For an insight into just how strange these functions are, you might want to look at the book *Fractals: Form, Chance, and Dimension* by Benoit Mandelbrot (Freeman 1977). This book discusses the properties of curves such as the one I chose for the imaginary coastline. This function is called a Triadic Koch curve and was discovered by von Koch in the early 1900's.

Mandelbrot's book provides an interesting glimpse of this strange mathematical world, but is quite frustrating in that he doesn't show the reader how to express these curves so that a computer could draw them.

Have you guessed that the turtle might hold the key to drawing fractals?

First, let's look at Figure 3, since this is the basic building block from which all the other curves are made. A procedure for creating this figure is shown below in both Apple LOGO and Atari PILOT. (This also lets you see the similarities and differences between these two languages.)

Apple LOGO	Atari PILOT
TO F0 :SIZE	*F0
FORWARD :SIZE	GR: DRAW #A
LEFT 60	GR: TURN -60
FORWARD :SIZE	GR: DRAW #A
RIGHT 120	GR: TURN 120
FORWARD :SIZE	GR: DRAW #A
LEFT 60	GR: TURN -60
FORWARD :SIZE	GR: DRAW #A
END	E:

If you move the turtle to the left edge of the screen and turn it to face right, then, by using the procedure F0, a replica of the curve in Figure 3 should appear on your display. For a line length of 81 units (a good size for Apple LOGO) you would type F0 81. In Atari PILOT you might want to use a line length of 54 units (because of screen resolution differences between the Atari and Apple graphics). To do this, just type:

```
C: #A=54
U: *F0
```

and you should see a curve similar to that in Figure 3.

Now, what about Figure 4? In our coastline, each straight line section is replaced with a copy of Figure 3 which has been reduced by one-third. Let's call the procedure to do this F1.

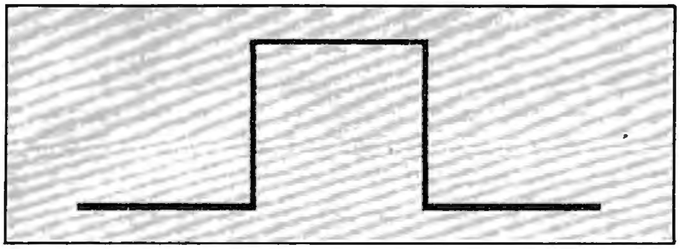
Apple LOGO	Atari PILOT
TO F1 :SIZE	*F1
F0 :SIZE/3	U: *F0
LEFT 60	GR: TURN -60
F0 :SIZE/3	U: *F0
RIGHT 120	GR: TURN 120
F0 :SIZE/3	U: *F0
LEFT 60	GR: TURN -60
F0 :SIZE/3	U: *F0
END	E:

Suppose we next want to draw the next level of this curve. All we need to do is create a copy of F1 (called F2) in which references to F0 are replaced by references to F1. Each time this procedure is used it will use F1 which will use F0. If you follow this process a few more times, you might make a procedure called F20 which uses F19, and so on.

How far do we need to go? If our original curve (Figure 3) used a line length of 27, for example, then each line should be 9 units long for F1,3 units long for F2 and 1 unit long for F3. Since the display can't show lines less than 1 dot long, it hardly makes sense for use to try to make this curve with any finer resolution.

In a future column we will show how LOGO lets you create these curves with just one procedure. Meanwhile, you can test your abilities by using a different figure as a starting pattern:

Figure 6.



Because of the 90 degree turns in this figure, those of you with WSFN will also be able to make this fractal curve.

Here is the first procedure you will need:

Apple LOGO	Atari PILOT
TO F0 :SIZE	*F0
FORWARD :SIZE	GR: DRAW #A
LEFT 90	GR: TURN -90
FORWARD :SIZE	GR: DRAW #A
RIGHT 90	GR: TURN 90
FORWARD :SIZE	GR: DRAW #A
RIGHT 90	GR: TURN 90
FORWARD :SIZE	GR: DRAW #A
LEFT 90	GR: TURN -90
FORWARD :SIZE	GR: DRAW #A
END	E:

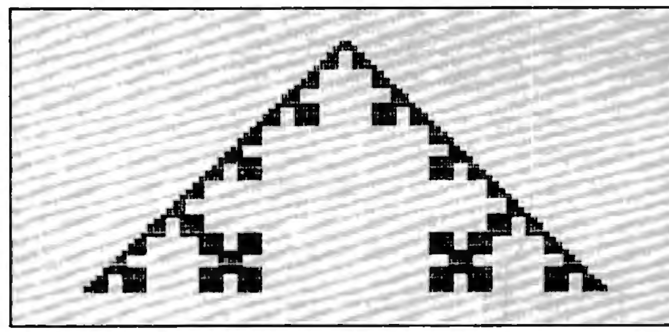
In WSFN you would create the procedure *A this way:

```
=*A(F6RF2RF2RF6RF)
```

(You might want to have the turtle move forward more with each step for the first few levels so you can see what is going on more easily.)

Here is what you should get for your final picture:

Figure 7.



Keep experimenting with other patterns – and let me know how you are doing.

Friends of the Turtle
P.O. Box 1317
Los Altos, CA 94022

©

How to move and rotate objects on screen. These ideas are adaptable to many computer graphics problems and can ease the task of creating games, simulations, etc. An Atari and Commodore version are presented as complete programs and the author describes the modifications necessary for other computers.

Screen Graphics

Ian Paull
Rochester, NY

Many computer simulations and games use graphics to represent a mathematical event or model in the program. In a lunar lander program, for example, the lander position is first calculated by the model (taking operator input into account) and then displayed. Another simulation might calculate how a machine or robot should be responding to computer control, and then graphically display this information.

Programs of these general types can benefit from graphics routines which can illustrate an object at any location on a video screen (translation) and at any angle (rotation). The BASIC program presented here uses a 25 x 40 screen area to display a user-generated figure at any location and angle that the screen can represent.

The screen is treated as an X Y coordinate system, with the X axis running from left to right, the Y axis running from top to bottom, and the origin (0,0) located at the upper left corner. (See Figure 1.) The plotting routine is the main section which can be modified. Briefly, line 1060 clears the screen. Line 1080 computes the memory address to be POKEd, from the X and Y values computed elsewhere. The POKEd address is the screen position which will be turned on. Line 1090 insures that only points on the screen will be POKEd. This line may be deleted for slightly faster running time if you are sure to always remain on screen.

Lines 1070 to 1110 form a loop that plots all points in the figure to be displayed. For computers with continuous memory mapped display, you may need to change the values of SS (screen start address), SE (screen end address), RL (row length), and the character code for a completely lit cell (160 for the PET, line 1100). The PET can support greater resolution (50 x 80) with additional software, replacing the PLOT subroutine. (See references 1 and 2.)

Visible Errors

Note that due to the coarseness of the display, rotated lines will often appear warped. This is because, with only a limited number of points on the screen to choose from, the computer must pick points as close as it can to the spot where the point should go, often resulting in visible error. This program is thoroughly REMarked, so the user should have little difficulty changing or omitting statements to suit his particular needs or computer.

The ten point example does one rotation in little more than one second.

The subroutines actually doing the graphics manipulations begin at line 830 for rotation and 970 for translation. Since translation is simpler, let's look at it first.

In moving a point from one spot on a graph to another, the X and Y processing can be handled separately. (See Figure 2.) In travelling from point X_0, Y_0, to X_1, Y_1, the difference in X (shown as ΔX) is simply added to X_0, and the difference in Y, a negative value in this example (shown as ΔY), is added to Y_0. For this single point translation, then:

$$X_1 = X_0 + \Delta X$$
$$Y_1 = Y_0 + \Delta Y$$

Subroutine 970 performs these additions for each translation of X and Y, for each point in the figure, and replaces the old X and Y values for each point with the new values. It also keeps a running total on the X and Y translation, to be used in the Rotation subroutine. Also, the Plot routine is called here to display the new, translated points.

The Rotation subroutine at line 830 is more complicated. The relationship between an existing point (X_0, Y_0) and the new computed point (X_1, Y_1) can be expressed as:

$$X_1 = X_0 \cos(a) - Y_0 \sin(a)$$
$$Y_1 = X_0 \cos(a) + Y_0 \cos(a)$$

where a is the angle of rotation. For details on the derivation of these relationships, see reference 3.

The Method Of Rotation

These equations describe rotation of a point about the origin. On initial entry of data for the pattern to be displayed, coordinates are chosen so the axis of rotation is at the origin. Then, if we later wish to

rotate about this axis after the figure has been translated elsewhere on the screen, we must first translate the figure back to the origin, do the rotation, and then translate back to where the figure had been. Naturally, these intermediate steps are not displayed on the screen.

The pattern of points which makes up the displayed figure is in the DATA statement of line 230. The first number is an X value, next comes its associated Y value, and so on until all points are described. The sequence in which points are described here is the sequence with which they will be plotted. (See Figure 3.)

Before branching to the ROTATION subroutine, the following variables must be initialized:

NP = Number of points in displayed pattern.

IA = Incremental angle of rotation, in radians.

AX, AY = Absolute X and Y values that the axis of rotation has been translated from the origin.

XP (NP), YP (NP) = Arrays containing X Y coordinates.

Before branching to the Translation subroutine, you must initialize:

XT, YT = Value of X and Y translation.
NP, XP (NP), YP (NP) = As before.

Note that AX and AY are normally updated in the Translation subroutine and should be set to zero at the start of the program.

When entering this program, the user can, if desired, omit all REMarks, since there are no branches to REM statements. When running the program, the figure will first appear in the upper left of the screen, partially hidden by the screen edge and wrapped around the display. Pressing the numeric keys translates the figure, a lop-sided plus sign, according to the standard scheme of Figure 4. In addition, pressing the zero will rotate the pattern 15 degrees counter-clockwise, and pressing the five will rotate 15 degrees clockwise.

As with most BASIC programs which manipulate graphics, these routines work slowly with large numbers of points. The ten point example does one rotation in little more than one second. If the program must rotate quickly, either the incremental angle of rotation can be increased, or sections of this program can be rewritten in machine language.

References

1) *High-Resolution Plotting for the PET*, J. Sherburne, Micro No. 10, March 1979, Machine language routine.

2) *Workbook 3*, Total Information Services, Basic routine for hi-res. plotting, PET.

3) *The Mathematics of Computer Graphics*, J. Posdamer, Byte Vol. 3, No. 9, Sept. 1978.

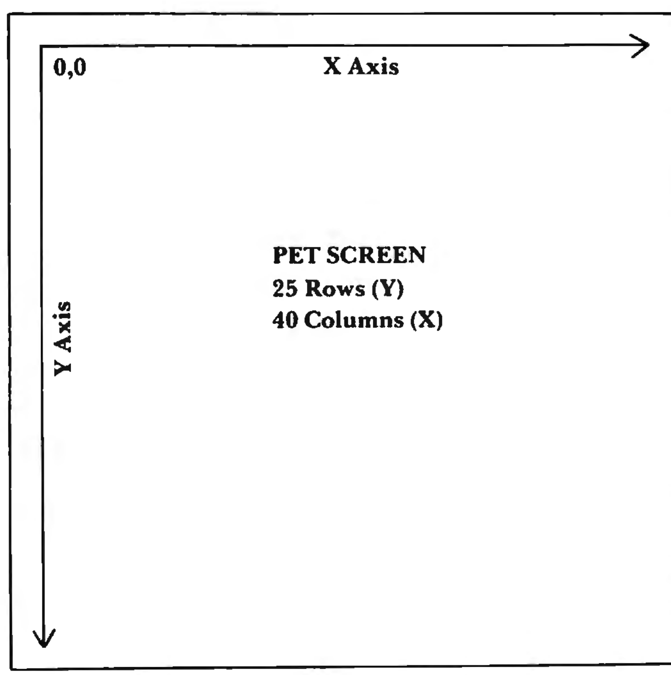
Notes For Atari Users

The Atari version of this routine closely parallels the PET version. Instead of screen POKEs, ordinary X,Y PLOTting is used, and instead of using the numeric keypad to generate the values for the translation routine, this program uses a joystick to move (translate) the figure around the screen. The less-than "<" and greater-than ">" keys are used to rotate the figure. Because of the general nature of this program, it can be run in Applesoft BASIC with only minor changes, such as changing GRAPHICS 3 + 16 to GR ; COLOR 1 to COLOR = 1, TRAP to ONERR GOTO, GET #1,A to GET A$:A = ASC(A$), etc., and using the keyboard, paddles, or a joystick to get the values for XT and YT (1,0, or -1).

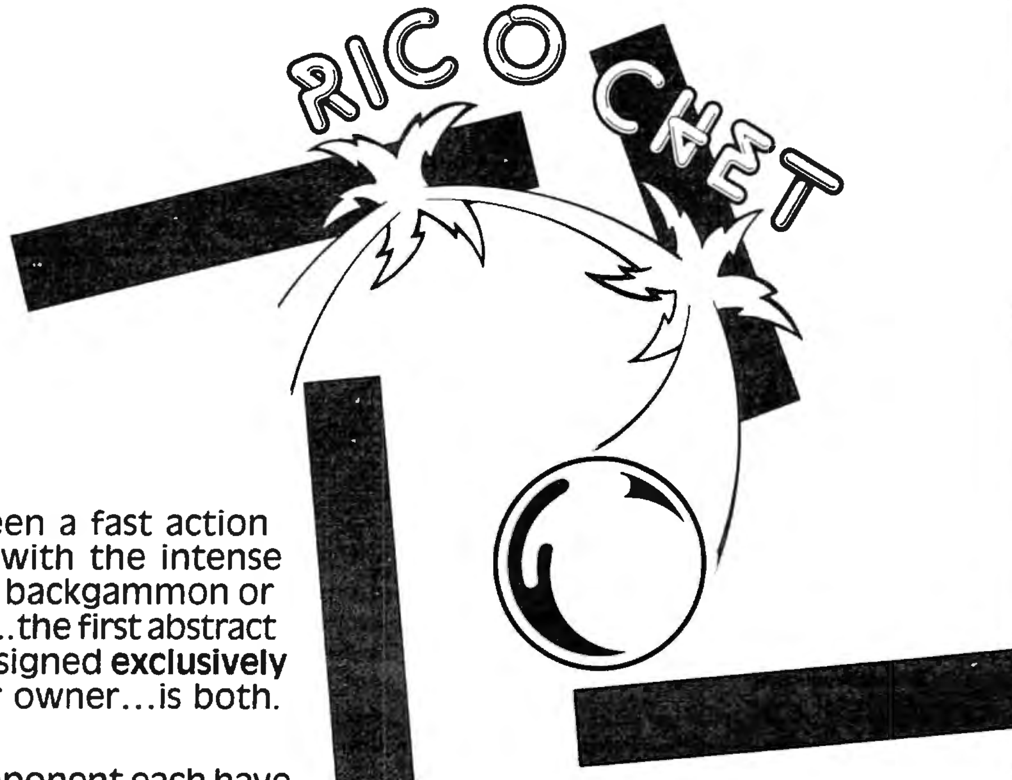
Other Machines:

If your BASIC supports X,Y PLOTting, use the Atari version as a guide, otherwise modify variables SS, SE, and RL in the PET/CBM version if you have a memory-mapped text display. (See the article for details.)

Figure 1.



It's Here! The Computer Strategy Game with Bounce!



For:
Apple
Atari
TRS-80

Have you ever seen a fast action game combined with the intense strategy of chess, backgammon or Othello? Ricochet...the first abstract strategy game designed **exclusively** for the computer owner...is both. And loads of fun!

You and your opponent each have six opposing blocks to maneuver and two launchers to fire. Your shots ricochet off the blocks, changing their position with every hit, while earning you points on the way to their targets. The longer your shot ricochets, the more points you get.

You don't have to play alone, either. Play against any of four different computer opponents, (each a different personality), or another human. Five game variants to choose from—each requires a different strategy.

Ricochet is truly competitive...if you want it to be. A "smart clock" lets you put more pressure on your opponent by forcing him to play faster than you. But you've got to win two out of three (or three out of five) games to claim victory.

Your computer rates you after each match, so you can compare your mastery of the game with that of other players. In the long run, you're trying to rack up points for your personal Ricochet Rating.

Get Ricochet now at your local dealer for your Atari, Apple or TRS-80. Suggested retail price: $19.95.

Another Mind Toy From EPYX

COMPUTER GAMES THINKERS PLAY

P.O. Box 4247, Mountain View, CA 94040.

Figure 2.

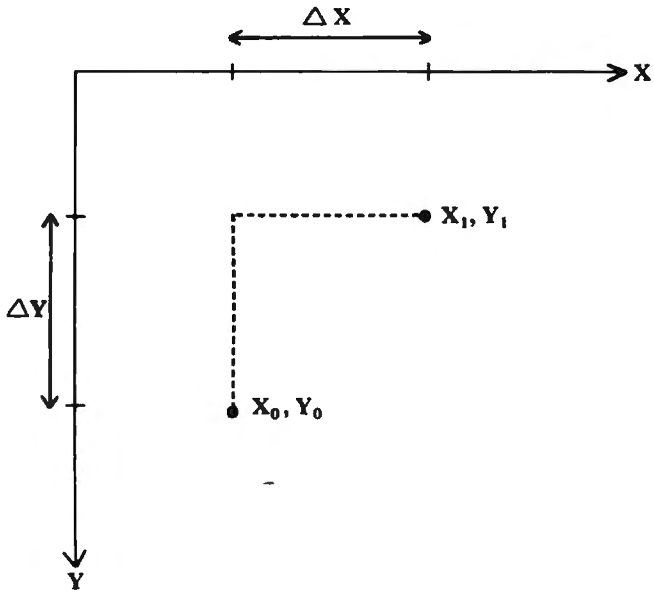


Figure 3.

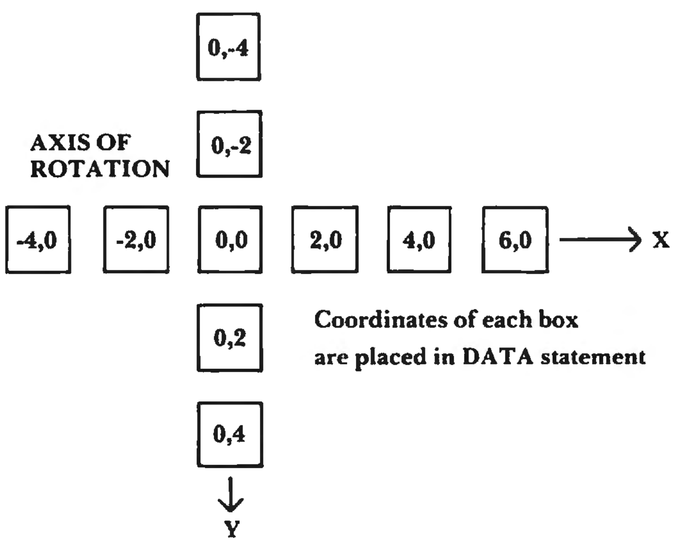
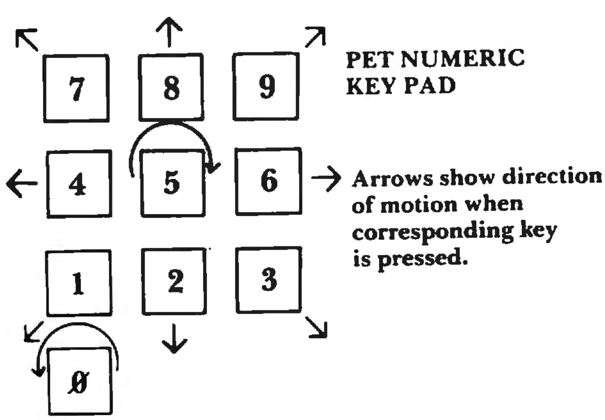


Figure 4.



Program 1. Commodore Microsoft Version

```
100 REM 2-D GRAPHICS SUBROUTINE DEMO
110 REM BY IAN PAULL
120 PRINT"{CLEAR}":REM CLEAR SCREEN
130 DIM XP(10),YP(10):REM ENLARGE FOR MORE ~
    POINTS
140 REM INITIALIZE
150 POKE 59468,12:REM PUTS PET IN STANDARD ~
    CHARACTER MODE
160 NP=10:SS=32768:SE=33767:RL=40:AX=0:AY=0

170 REM NP=NUMBER OF POINTS; SS=SCREEN STAR
    T ADDRESS; SE=SCREEN END ADDRESS
180 REM RL=ROW LENGTH; AX,AY=ACCUMULATED X ~
    AND Y VALUES (NET XLATION SO FAR)
190 FOR KZ=1 TO NP:REM READ FIGURE COORDS
200 READ XP(KZ),YP(KZ)
210 NEXT KZ
220 REM DATA BELOW ARE FIGURE COORDS. FORMA
    T FOR EACH POINT IS X,Y
230 DATA-4,0,-2,0,0,0,2,0,4,0,6,0,0,4,0,2,0
    ,-2,0,-4
240 GOSUB 1060:REM PLOT
250 GETA$:IFA$=""THEN250:REM LOOPS UNTIL AN
     INPUT
260 A=VAL(A$)+1:REM CONVERTS STRING INPUT T
    O NUMERIC VALUE
270 ON A GOTO 290,370,420,470,520,330,570,6
    20,670,720
280 REM CCW ROTATE, FIXED ANGLE 15 DEG
290 IA=-(15/360)*(2*):REM_=PI
300 GOSUB 780:REM ROTATE ROUTINE
310 GOTO 250
320 REM CW ROTATE, FIXED ANGLE 15 DEG
330 IA=(15/360)*(2*):REM_=PI
340 GOSUB 780:REM ROTATE ROUTINE
350 GOTO 250
360 REM XLATE DOWN AND LEFT
370 XT=-1
380 YT=1
390 GOSUB 970:REM XLATE ROUTINE
400 GOTO 250
410 REM XLATE DOWN
420 XT=0
430 YT=1
440 GOSUB 970:REM XLATE ROUTINE
450 GOTO 250
460 REM XLATE DOWN AND RIGHT
470 XT=1
480 YT=1
490 GOSUB 970:REM XLATE ROUTINE
500 GOTO 250
510 REM XLATE TO LEFT
520 XT=-1
530 YT=0
540 GOSUB 970
550 GOTO 250
560 REM XLATE TO RIGHT
570 XT=1
580 YT=0
590 GOSUB 970
600 GOTO 250
610 REM XLATE UP AND LEFT
620 XT=-1
630 YT=-1
640 GOSUB 970
650 GOTO 250
660 REM XLATE UP
670 XT=0
```

```
680 YT=-1
690 GOSUB 970
700 GOTO 250
710 REM XLATE UP AND TO RIGHT
720 XT=1
730 YT=-1
740 GOSUB 970
750 GOTO 250
760 REM ROTATE CONTROL
770 REM FIRST, XLATE TO ORIGIN
780 FOR KZ=1 TO NP
790 XP(KZ)=XP(KZ)-AX
800 YP(KZ)=YP(KZ)-AY
810 NEXT KZ
820 REM DO ROTATION ABOUT 0,0
830 SA=SIN(IA):CA=COS(IA)
840 FOR KZ=1 TO NP
850 KN=XP(KZ)
860 XP(KZ)=KN*CA-YP(KZ)*SA
870 YP(KZ)=KN*SA+YP(KZ)*CA
880 NEXT KZ
890 REM XLATE BACK
900 FOR KZ=1 TO NP
910 XP(KZ)=XP(KZ)+AX
920 YP(KZ)=YP(KZ)+AY
930 NEXT KZ
940 GOSUB 1060: REM PLOT
950 RETURN
960 REM XLATE
970 AX=AX+XT:REM KEEP TRACK OF ABS X
980 AY=AY+YT:REM KEEP TRACK OF ABS Y
990 FOR KZ=1 TO NP:REM SHIFT X AND Y
1000 XP(KZ)=XP(KZ)+XT
1010 YP(KZ)=YP(KZ)+YT
1020 NEXT KZ
1030 GOSUB 1060: REM PLOT
1040 RETURN
1050 REM PLOT
1060 PRINT"{CLEAR}":REM CLR SCREEN
1070 FOR KZ=1 TO NP:REM FIGURE ADDRESSES TO
 BE POKED
1080 LOC=(SS+RL*INT(YP(KZ)+.5)+INT(XP(KZ)+.5
))
1090 IF LOC<SS OR LOC>SE THEN NEXT KZ:RETURN
:REM DON'T POKE OFF SCREEN
1100 POKE LOC,160:REM LIGHT CHARACTER CELL A
T SCREEN LOCATION 'LOC'
1110 NEXT KZ
1120 RETURN
1130 END
```

Program 2. Atari Version

```
110 OPEN #1,4,0,"K"
120 GRAPHICS 3+16
130 DIM XP(10),YP(10):REM ENLARGE DIMENS
ION FOR MORE POINTS
140 REM INITIALIZE
160 AX=0:AY=0:NP=10:PI=3.1415927
170 REM NP=NUMBER OF POINTS
190 FOR KZ=1 TO NP
200 READ T:XP(KZ)=T:READ T:YP(KZ)=T
210 NEXT KZ
220 REM DATA BELOW ARE FIGURE COORDS.
225 REM FORM FOR EACH POINT IS X,Y
230 DATA -4,0,-2,0,0,0,2,0,4,0,6,0,0,4,0
,2,0,-2,0,-4
240 GOSUB 1060:REM PLOT
250 IF PEEK(764)<255 THEN 300
260 ST=STICK(0):IF ST=15 THEN 250
270 XT=-(ST>8 AND ST<12)+(ST>4 AND ST<8)

280 YT=(ST=9 OR ST=5 OR ST=13)-(ST=10 OR
 ST=14 OR ST=6)
290 GOSUB 970:REM XLATE ROUTINE
295 GOTO 250
300 GET #1,A
310 IF A<>60 AND A<>62 THEN 250
320 IA=(-(A=60)+(A=62))*(15/360)*(2*PI)
330 GOSUB 780:REM ROTATE ROUTINE
340 GOTO 250
760 REM ROTATE CONTROL
770 REM FIRST, XLATE TO ORIGIN
780 FOR KZ=1 TO NP
790 XP(KZ)=XP(KZ)-AX
800 YP(KZ)=YP(KZ)-AY
810 NEXT KZ
820 REM DO ROTATION ABOUT 0,0
830 SA=SIN(IA):CA=COS(IA)
840 FOR KZ=1 TO NP
850 KN=XP(KZ)
860 XP(KZ)=KN*CA-YP(KZ)*SA
870 YP(KZ)=KN*SA+YP(KZ)*CA
880 NEXT KZ
890 REM XLATE BACK
900 FOR KZ=1 TO NP
910 XP(KZ)=XP(KZ)+AX
920 YP(KZ)=YP(KZ)+AY
930 NEXT KZ
940 GOSUB 1060:REM PLOT
950 RETURN
960 REM XLATE
970 AX=AX+XT:REM KEEP TRACK OF ABS X
980 AY=AY+YT:REM KEEP TRACK OF ABS Y
990 FOR KZ=1 TO NP:REM SHIFT X AND Y
1000 XP(KZ)=XP(KZ)+XT
1010 YP(KZ)=YP(KZ)+YT
1020 NEXT KZ
1030 GOSUB 1060:REM PLOT
1040 RETURN
1050 REM PLOT
1060 GRAPHICS 3+16:SETCOLOR 0,0,10:COLOR
 1
1070 FOR KZ=1 TO NP
1080 TRAP 1090
1085 PLOT XP(KZ),YP(KZ):TRAP 40000
1090 NEXT KZ
1100 RETURN
1130 END
```

©

A Monthly Column

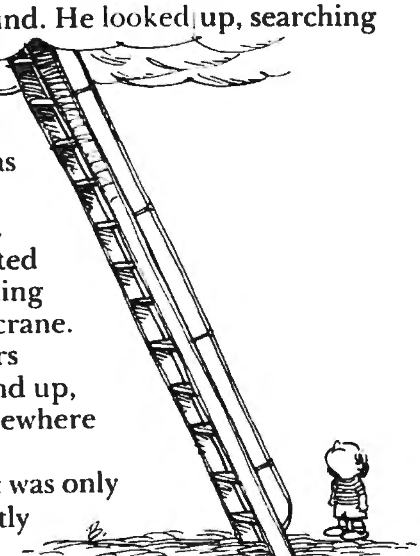
The World Inside The Computer

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.

The Computer Playground

Fred D'Ignazio
Associate Editor

Eric ran over to the royal-blue sliding board, standing like a petrified monolith in the corner of his school playground. He looked up, searching for the top of the sliding board. It was lost somewhere in the drifting clouds. The metal rail glinted in the sun, resembling the arm of a giant crane. The dirty blue stairs seemed to go up and up, maybe ending somewhere in heaven.



Although Eric was only three, he confidently grabbed onto the sliding board's steel handrail and began clinging the steep metal stairs. After he got to the top, Eric stopped, panted like a puppy, and surveyed his playmates, screeching and dashing around the playground. His friends looked strangely small, since Eric was so high up.

Eric stretched on his tip-toes and grabbed the chinning bar above the sliding board. He swung back and forth wildly.

"Karla, watch!" he yelled to his teacher.

Halfway through each arc, he smacked his feet against the top of the sliding board. Loud percussion noises echoed across the playground.

Suddenly, Eric let go and went flying down the sliding board. He slid dizzily round and round the board's corkscrew spirals. He flew off the end of the board, sailed across a mud puddle, and demolished a sand castle being built by two of his friends. His friends went crying to their teacher. Eric howled. When he landed on the sandpile, he had scraped his bottom on the tip of an upturned shovel.

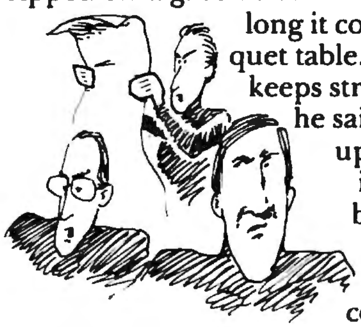
Moments later, the two kids were at work on another sand castle, undisturbed about the prospect of new children falling from the sky.

Eric was far away, on the opposite side of the playground. His cheeks were splashed with gritty tears, and his ears were filled with sand, but he was smiling. He had a new project. He was crab crawling across the green-shingled roof of the school's playhouse, trying to work up his courage to jump into the old rowboat beached under the house's eaves. Eric thought maybe he could do it. After all, what's jumping off a roof to a three-year-old who can climb to heaven and drop from clouds and clobber castles?

The Computer Alchemists

The dark-suited men were frowning and fiercely

intent. They spoke a cryptic, arcane language. "Disk error in sector five," said one. "Uh-huh," said another. "We'll have to bleed the accumulator." A third stared at a wide piece of paper he had just ripped off a green-and-white sheet of paper so long it could have covered a banquet table." The 0S slave program keeps stripping the control bits!" he said angrily. He crumpled up the paper and tossed it into a navy-green wastebasket under his spotless desk.



The men worked in a cool, almost Arctic room, drained of color by the overwhelming bath of white light emanating from the banks of flourescent lamps fastened to the ceiling. Tall panels of lights winked like a city's streetlights viewed from a distant hill. Rows of metal boxes were filled with rotating platters and reels of gleaming tape that spun and whirred.

Staccato, machine-gun noises filled the room, as three of the boxes disgorged endless sheets of paper. The paper looked like a giant, jerking tongue or an impossibly long scarf pulled from a magician's sleeve.

The men were only vaguely aware of the tap-dancing printer and the clacking tape drives. These sounds were muffled by a loud whooshing noise that seemed to cover everything in the room like an invisible volcanic ash. The noise came from air pouring through the wall grates from huge cooling and filtering fans. After several hours in the room, the noise could numb a person's whole body and make it feel wrapped in a cocoon of cotton candy.

When Alien Worlds Collide

Eric on the sliding board and the men in the cotton-candy room seem to be worlds apart. Eric is a toddler having fun on a playground. The men are high-level programmers and analysts designing a new computer system.

Eric is outdoors in the sun. The men are buried deep in the bowels of the Pentagon, in a guarded, windowless computer room.

Eric is covered with dirt and sand, and sweating from his exertions. The men are covered with plastic badges that proclaim their dedication and their utter seriousness about their craft. Dust and dirt are not permitted in their room. Only, sweat – cold sweat.

I used to be one of those men. And Eric is my son. Now Eric's world and the world where I once worked have intersected, like alien worlds colliding.

When did the collision take place? It occurred when I first turned Eric loose on my home computer.

The shock was antic. It was also profound.

Eric is a person of arbitrary, random action. He is a person driven by fantasy, by enormous appetites and desires. His energy seems boundless. His temper is formidable. He vibrates to the tune of unchecked emotions and enthusiasms. He cringes from invisible monsters that scare him half silly, then walks unconcerned past real monsters he still hasn't learned to fear. To my most careful, clearly put instructions, he answers, "What?"

Computers, on the other hand, are creatures of reason and precision. They are powerful tools, the product of advanced technology. Computers are predictable. If we put in the right command, we get the right answer. Or, if we feed the computer garbage, we get garbage back.

Computers and three-year-olds seem worlds apart. Just imagine a computer that acted like a three-year-old. Or imagine a three-year-old that acted like a computer.

Three-year-olds are hardy and tough. Computers are finicky and delicate. Three-year-olds revel in dirt, mud, and grime. Computers like to be clean. They work better that way.

Knives, Forks, And Attila The Hun

Eric has lots of nicknames: Little Hulk, Commando Kid, Tank, and the Lone Ranger, among others. My favorite, though, is Attila the Hun. This is an apt nickname because Eric, at heart, is a barbarian.

Can a barbarian ever learn to eat with a knife and fork? Can he be trusted in the same room as a computer?

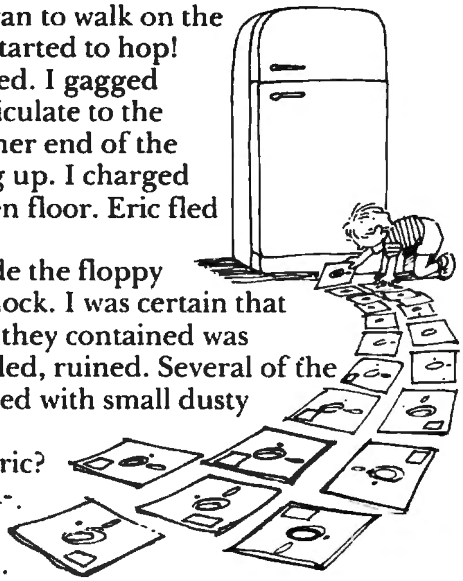
Sometime, over a year ago, I was on the phone in the kitchen talking with someone in California. I was trying to obtain some photos for a book I was writing.

Eric was home with me all day, back then, while his mommy went to work and his sister went to school. I didn't worry about Eric while I talked on the phone, since Eric was busy crayoning on a piece of scrap paper on the floor of my study. Or so I thought.

I was deeply engrossed in the conversation, trying hard to visualize the Californian's robot turtle, which I had never seen. Then I happened to look up, and my heart stopped.

There, on the kitchen floor, Eric was hard at work, creating a flagstone sidewalk made out of my floppy disks. My *floppy disks.* I had stored all my books and programs on those disks. Eric must have known how important the disks were because he had carefully removed each disk from its protective case and gently dropped it on the floor. He had placed all the disks in a neat row that ended at the refrigerator. Now that he was finished building his

sidewalk, he began to walk on the disks. Then he started to hop! "Eric!" I screamed. I gagged something inarticulate to the person at the other end of the phone and hung up. I charged across the kitchen floor. Eric fled to his bedroom.



I knelt beside the floppy disks. I was in shock. I was certain that the information they contained was squished, trampled, ruined. Several of the disks were covered with small dusty footprints.

Did I flog Eric? Was my information destroyed?

Happily, no.

But that was only the first of a brief, but emotional, series of incidents involving Eric and the floppy disks. The problem was he couldn't keep his hands off them. Something perverse inside him told him he had me by the throat. All he had to do was point to a disk, and I flew into a tizzy. Every time I left my study, he would sneak in and rob several disks and hide them under his bed. For awhile, things got so bad, I began sealing my disk drawer shut with strapping tape, just to thwart Eric.

Then I had an inspiration. I gave Eric a disk of his own. It was an old disk that was no longer reliable. I wrote Eric's name on the disk in big green, magic-marker letters. I wrapped the disk up and gave it to him as a present.

He was delighted. He still keeps the disk in a place of honor on his bookshelf.

And he stopped robbing my disks.

Now You See It, Now You Don't!

I have two kids. Eric, of course, and Catie, who is six.

I have several computers, and I have turned my study into a videogame arcade and a computer programming workshop for neighborhood kids.

The kids can use all the computers in the room except one, my writing computer. That computer is supposed to be off limits.

One day, while the kids were banging away at the other computers, I got up from my computer to stretch and make a snack in the kitchen. On my way back into the room, five minutes later, I happened to glance at my computer screen. Before I left, the screen had been filled with words – a section of a new book chapter I was writing. Now the screen was empty.

I panicked. I ran into the room yelling at the kids. "Who messed with my computer?" I hollered. "My chapter's gone. I've lost hours of work."

Then I checked the computer. I was sure it had been turned off, but it was still on. I did some more checking. Finally, I noticed the screen brightness switch. I turned it. Magically, my words reappeared. My chapter was untouched. Nothing was lost.

Just then, a very contrite five-year-old boy came up to me with his head drooping. "I did it, Mr. D'Ignazio," he said. "I didn't mean to kill your computer."

Joystick Tug Of War

Another time, I was in the kitchen eating dinner with my family. Several children were still playing computer games in the study. All of a sudden, from the study came shouts and scuffling noises.

"It's my turn to blast them!" one child cried.

"You just blasted them!" cried another child. "Now it's my turn!"

I ran into the study and found a six-year-old and a nine-year-old (brother and sister) doing their best to rip two joysticks out of a computer.

"What are you two doing?" I shouted.

The kids turned toward me. Frightened, they dropped their joysticks. One broke open on the study's hardwood floor.

"Out! Out!" I yelled.

They dashed past me and ran out the front door of the house.

How Come This Doesn't Fit?

Another time, I was in the kitchen preparing dinner for my family. It was close to five-thirty, and my wife, Janet, would be home soon from the office. As usual, the study was filled with kids playing with the computers.

I was slicing up some carrots for the salad, when I heard strange grunting noises coming from the study. Were the kids in there wrestling among the computers?

Hurriedly I dried my hands and went to see what was the matter.

When I got to the study door, I couldn't believe what I saw. A five-year-old boy was standing in front of one of the floppy disk drives. That was okay. I had taught all the kids – even Eric – how to insert disks properly into the drive.

But this kid wasn't inserting a disk. He was trying to shove a game cartridge into the drive. He was trying to squeeze an inch-thick plastic box into a hole designed for a disk only a sixteenth of an inch thick. He grunted loudly because he was trying so hard. He knew it was a tough thing to do, and he was giving it his best shot.

Strategies For Survival

Given these tales of abused and violated computers, what is the best strategy for mixing young people and computers?

You could put a lock on the computer room door until the kids all turn sixteen. Or until they leave home.

You could keep the computer in the car trunk and only bring it in the house after the kids are asleep.

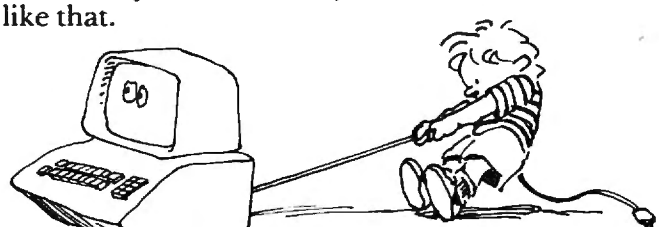
You could even fortify the computer, beef it up, and make it indestructible, the way the military does with computers onboard tanks, guided missiles, and submarines.

You could do these things, but I would advise against it. In their place, I would advise a "Computer Orientation Session," followed by frequent "refresher" courses. I would also recommend a few simple precautions. And trust – lots of trust.

Based on my experience, I know that even toddlers can be taught to use computers properly.

Sure, there are lapses. (Oh, boy, are there lapses.) But, for the most part, Eric and his little friends do an amazing job with the computers. They turn them on and off. They adjust the TV screen. They insert and remove disks and cartridges, even tapes. They know how to call up programs and operate the keyboard, joysticks, and other controls.

But they must be taught first. You need to sit them down and go over the "etiquette" of using a computer. Compare it to the family pet. You shouldn't pull its tail or try to ride on it. It wouldn't like that.



Instead, you need to be gentle with it and take care of it. If you do the right things, it will be nice to you and give you pleasure.

Frequent drills on specific rules help a lot, here.

It would help if you give the kids some responsibilities for the computer's care that are just their own. Like keeping the keyboard free of cookie crumbs. Keeping the screen wiped clean of fingerprints and all sorts of unmentionable objects. Or reporting any irregularities such as wires in the wrong place, or anything that looks dangerous.

Remind the kids frequently that computers run on electricity, and that electricity, like fire, can serve them or hurt them.

Also, if you can afford it, give them their own floppy disk, cartridge, or tape. Help them store their favorite programs and games. Give them a small case to store their printouts, game instructions, and other computer-related materials. This way, the computer will become partly theirs. As its owners, they will try harder to care for the computer and protect it.

The following precautions, like the advice above, are just common sense. For example, I learned to put rugs on my study floor. Then, when computer objects dropped, there was a smaller chance that they'd actually break.

Also, back up *all* your important programs and files. Then keep those backup copies in your office, your bedroom, or in a locked safe. I can tell you, it's a lot easier to keep your cool when your child puts your floppy disk in the toaster, if you have a backup to the disk.

Keep all plugs, wires, and cables (anything electric) hidden and out of the line of fire. My kids could trip on a paper clip. How about yours?

Put all your equipment on top of large, sturdy tables, surrounded by lots of sturdy chairs.

Keep your cartridges, disks, tapes, manuals, programs, etc. organized. Make everything accessible to your children. But make them respect and adhere to your organization.

Teach them standards. All devices must be shut off after use. Disks and tapes must not be left out after use. Everything has a place. (Nag them about this rule, if only to preserve your sanity.)

Two last elements of strategy. First, try hard not to get upset when it looks like your kids have done something to the computer. Most of the time things are a lot better than they seem at first. We've had dozens of accidents with computers here at the "D'Ignazio Arcade," yet we've never broken a computer. At least not completely.

Second, don't push your kids to use the computers. Your three-year-old doesn't have to use educational programs to benefit from the computer. Your six-year-old doesn't have to learn to program.

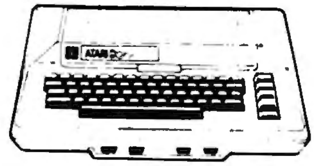
Just the exposure to computers is doing your kids a world of good. Just switching buttons off and on. Or typing letters and shapes on the keyboard and watching them appear on the screen.

If you make kids' computer time synonymous with drudgery, with work, with tension, or with pressure, the kids won't like computers.

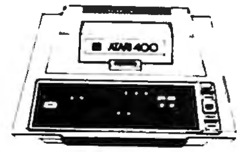
Don't push the kids. Just open the door, and sit back and watch. Computers fascinate kids, and that fascination will motivate them to learn on their own.

And congratulate yourself on your accomplishment. You are bringing together two very different worlds. And the results? The results are anybody's guess. ©

ATARI HOME COMPUTERS



ATARI 800
16K ... $669
32K ... $749
48K ... $779



ATARI 400
16K ... $329
32K ... $419
48K ... $519

410 Recorder	$76.00
810 Disc Drive	$449.00
822 Printer	$269.00
825 Printer	$629.00
830 Modem	$159.00
820 Printer	$269.00
850 Interface	$169.00
New DOS 2 System	$29.00
CX30 Paddle	$18.00
CX40 Joy Stick	$18.00
CX853 16K RAM	$77.95
Microtek 16K RAM	$64.95
Microtek 32K RAM	$99.95
Ramdisk (128K)	$429.95
Intec 48K Board	$199.95
Intec 32K	$99.95

All RAM prices good through July 1982 only

One year extended warranty	$70.00
481 Entertainer	$69.00
482 Educator	$130.00
483 Programmer	$49.00
484 Communicator	$344.00

ATARI HOME COMPUTER PROGRAMS

HOME OFFICE	
CX404 ATARI Word Processor	$119.00
CX8102 Calculator	$29.00
CX412 Dow Jones Investment Evaluator	$99.00
CX4109 Graph It, Joystick optional	$17.00
CX4104 Mailing List	$20.00
CX4115 Mortgage & Loan Analysis	$13.00
CX4103 Statistics 1	$20.00
CX8107 Stock Analysis	$20.00
CXL4015 TeleLink 1	$23.00
HOME STUDY	
CX4101 An Invitation to Programming 1	$20.00
CX4106 An Invitation to Programming 2	$23.00
CX4117 An Invitation to Programming 3	$23.00
CX4107 Biorhythm	$13.00
Conversational Languages (ea.)	$46.00
CX4121 Energy Czar	$13.00
CX4114 European Countries & Capitals	$13.00
CX4108 Hangman, Joystick optional	$13.00
CX4102 Kingdom	$13.00
CXL4007 Music Composer	$34.00
CX4123 Scram, uses joystick	$20.00
CX4112 States & Capitals	$13.00
CX4110 Touch Typing	$20.00
HOME ENTERTAINMENT	
PAC MAN	$35.00
CENTIPEDE	$35.00
CAVERNS OF MARS	$32.00
CXL4013 Asteroids	$29.00
CXL4004 Basketball	$27.00
CX4105 Blackjack	$13.00
CXL4009 Computer Chess	$29.00
CXL4012 Missile Command	$29.00
CXL4008 Space Invaders	$29.00
CXL4011 Star Raiders	$39.00
CXL4006 Super Breakout	$29.00
CXL4010 3-D Tic-Tac-Toe	$27.00
CXL4005 Video Easel	$24.00
PROGRAMMING LANGUAGES AND AIDS	
CXL4003 Assembler Editor	$47.00
CXL4002 ATARI BASIC	$47.00
CX8126 ATARI Microsoft BASIC	$70.00
CXL4018 PILOT	$72.00
CX405 PILOT (Educational)	$105.00

ATARI PROGRAM EXCHANGE:	
Eastern Front '41	$25.50
Avalanche	$15.50
Outlaw	$15.50
747 Landing Simulation	$15.50
Babel	$15.50
Dog Daze	$15.50
Downhill	$15.50
Attack!	$15.50
Blackjack-Casino	$15.50
Reversi II	$15.50
Domination	$15.50
Solitare	$15.50
Disk Fixer	$15.50
Supersort	$15.50
Data Management	$15.50
Chameleon	$15.50
Instedit	$15.50
Insomnia	$15.50
My First Alphabet	$25.50
Mapware	$18.00
Number Blast	$11.50
Family Cash Flow	$15.50
Weekly Planner	$15.50
Bowler's Data Base	$13.00
Banner Generator	$11.50
Crystal Software	
Bermuda Fantasy	$26.00
Beneath Pyramids	$20.00
Galactic Quest	$26.00
House of Usher	$20.00
Forgotten Island	$26.00
Haunted Palace	$33.00
Compumax (Acct. Rec., Gen. Ledger, Inventory, Payroll, ea.)	$110.00
Synapse	
File Manager 800	$69.95
Dodge Racer	$19.00
Chicken	$24.00
Slime	$24.00
Nautilus	$24.00
Disk Manager	$24.00
Fort Apocalypse	$24.00
Assembler	$39.00
Protector	$24.00
EXPY (Automated Simulation)	
Ricochet	$14.50
Crush Crumble & Chomp	$24.00
Star Warrior	$29.00
Rescue at Rigel	$24.00
Datestones	$16.00
Invasion Orion	$18.50
Mission Asteroid	$22.00
OnLine	
Jaw Breaker	$27.00
MouskATTACK	$31.00
The Next Step	$34.00
Softporn	$27.00
Wizzard & Princess	$29.00
Arcade Plus	
Ghost Hunter (cassette)	$24.00
Ghost Hunter (disk)	$30.00
K-Byte	
K-BYTE Krazy Shoot Out (ROM)	$39.00
K-DOS	$69.00
Visicalc	$179.00
Letterperfect (Word Processor)	$109.00
Data Soft Textwizzard	$89.00

Texas Instruments

TI-99/4A $299

PHA2100 R F Modulator	$29.00
PHP 1600 Telephone Coupler	$179.00
PHP 1200 Peripheral Expansion Box	$199.00
PHP 1220 RS 232 Card	$143.00
PHP 1240 Disk Controller	$199.00
PHP 1250 Disk Drive	$319.00
PHP 1100 Wired Remote Controllers (Pair)	$31.00
PHP 1260 32K RAM	$229.00
PHP Printer Solid State	$319.00
PHM 3006 Home Financial Decisions	$26.00
PHM 3013 Personal Record Keeping	$43.00
PHD 5001 Mailing List	$60.00
PHD 5021 Checkbook Manager	$18.00
PHM 3008 Video Chess	$60.00
PHM 3010 Physical Fitness	$26.00
PHM 3009 Football	$26.00
PHM 3018 Video Games I	$26.00
PHM 3024 Indoor Soccer	$26.00
PHM 3025 Mind Challengers	$22.00
PHM 3031 The Attack	$35.00
PHM 3032 Blasto	$22.00
PHM 3033 Blackjack and Poker	$22.00
PHM 3034 Hustle	$22.00
PHM 3036 Zero Zap	$18.00
PHM 3037 Hangman	$18.00
PHM 3038 Connect Four	$18.00
PHM 3039 Yahtzee	$22.00
Tombstone City 21st Century	$34.00
Munch Man	$34.00
T I INVADERS	$34.00
CAR WARS	$34.00

Printers

Smith Carona TP1	$699.00
Centronics 739-1	$519.00
Centronics 739-3	$619.00
Diablo 630 Special	$1799.00
Epson	
MX80 w/Graftrax	$449.00
MX80FT	Call
MX100	Call
NEC	
8023	$549.00
7730	Call
7720	Call
7710	Call
Okidata	
82A	$499.00
83A	$749.00
84	$1129.00
Citoh Starwriter	
F10-40 CPS	$1469.00
F10-55 CPS	Call
Prowriter	$479.00
Talley	
8024-L	$1629.00
IDS	
Prism	$Call
MPC Apple Parallel Board & Cable	$69.00
2 Meter RS232-RS232	$29.95

Cables Available For Most Interfacing Purposes

Computer Covers

ATTRACTIVE DUST COVERS FOR YOUR COMPUTER AND DISK DRIVE.

Atari 400	$6.99
Atari 800	$6.99
Atari 810	$6.99
Commodore VIC-20	$6.99
Commodore 8032	$14.99
Commodore 8050/4040	$10.99

All Atari Covers are Beige.

All Commodore covers are Royal Blue

Most software for Atari 400/800 available on cassette or disk.

computer mail order west

CALL TOLL FREE **800-648-3311**

Franco Habla Espanol

IN NEVADA, CALL (702) 588-5654

P.O. BOX 6689, STATE LINE, NEVADA 89449

With versions for the VIC and Atari computers, this program illustrates how to make games and learning programs easy for preschoolers to use.

Answer Selection With Joysticks

Stephen Levy
Bowie, MD

One of my major objectives in purchasing a computer was to create educational programs for my young son. Realizing that a preschool child might have difficulty using the keyboard, I decided that using the joystick to input responses was the answer. Not that learning to use the keyboard is not important, but if that is not the purpose of the program, it might get in the way and cause frustration.

Al Baker's two articles in *COMPUTE!'s First Book Of Atari* gave me ideas and the little extra push I needed.

The purpose of this educational program is to give a young child practice in deciding which of four words is different from the other three on the screen. The child uses the joystick to make his selection and is allowed as many tries as needed. Once the correct response is picked, a short explanation appears and the child can then decide whether to do another problem or to end the program. All this is done with the joystick.

The program is very straightforward. All data for problems are in lines 20 to 199, with even-numbered lines containing the words and explanations in the odd-numbered lines. I have included only four series of words (lines 20,22,24,26) so that the reader may add his own. You must adjust line 220 to equal the number of even-numbered data lines (total series of words); in my example there are only four.

example: 220 Q = INT(RND(0)*A) + 10:RESTORE Q*2 where A equals the number of problems.

Lines 210-215 randomly select the background colors.

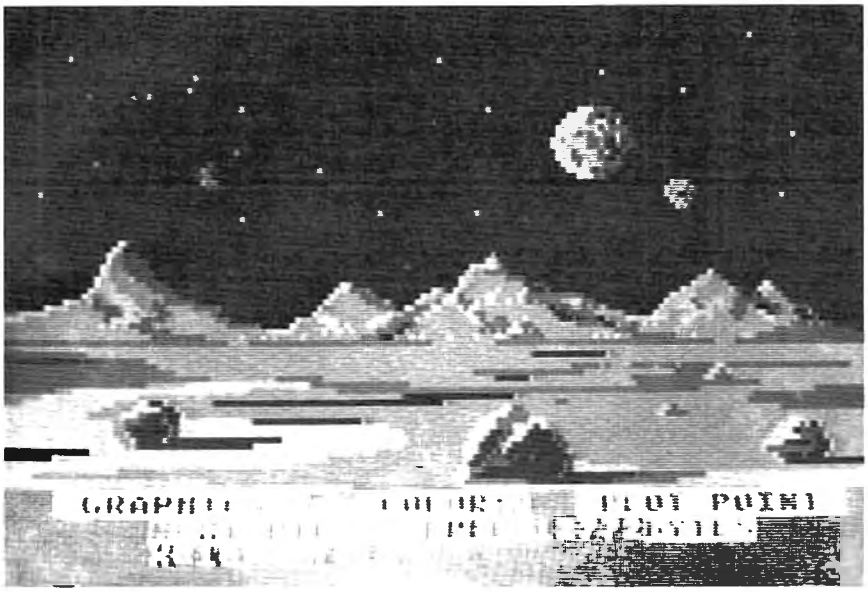
Since my purpose is to demonstrate the use of the joystick in an educational program, I have not added any graphic displays, but I would suggest "dressing up" the program for use with young children in order to maintain their interest. Below is an explanation of the program by line.

Lines	
5-10	Initialization.
20-27	Data for questions.
28-199	To be used for additional data for questions. Data placed as follows: Four choices, then joystick position of correct answer. In the order that they appear in data line. If the first answer is correct then the number should be, 14; second, 11; third, 7; fourth, 13.
200	Sets mode and turns off cursor.
210-215	Randomly selects background color and print.
220	Randomly selects data to be used. The number multiplied by RND(0) should be equal to the number of questions.
230-270	Reads data and positions words on screen.
300-350	Positions @ symbol in the direction of user's answer.
360	Checks for correct answer.
385-390	Clears @ from incorrect answer.
395	Gives user another try at the same question.
400-420	Reads and prints explanations on screen.
440-470	Gives user the option of another problem.

Program 1. Atari Version

```
1 REM D:JOYSTICK
2 REM STEPHEN LEVY
3 REM 3511 MORLOCK LN, BOWIE, MD.20715
4 REM 301-464-2052
5 DIM Q1$(20),Q2$(20),Q3$(20),Q4$(20),E$
(40),B$(1),C$(1),CLEAR$(1)
10 B$=" ":CLEAR$=CHR$(125):C$="@"
15 GOTO 200
20 DATA APPLE,PEAR,HAT,ORANGE,7
21 DATA HAT IS NOT A FRUIT
22 DATA A,E,I,S,13
23 DATA 'S' IS NOT A VOWEL
24 DATA BLUE,GREEN,RED,SAD,13
25 DATA THESE WORDS ARE COLORS
26 DATA APE,APPLE,ATE,FUN,13
27 DATA THESE WORDS BEGIN WITH 'A'
200 GRAPHICS 2:POKE 752,1
210 C=RND(0)*16:SETCOLOR 1,9,2:SETCOLOR
4,C,8:SETCOLOR 2,11,8
215 SETCOLOR 0,C,1
220 Q=INT(RND(0)*4)+10:RESTORE Q*2
230 READ Q1$,Q2$,Q3$,Q4$,A
240 POSITION 10-INT((LEN(Q1$))/2),1:PRIN
T #6;Q1$
250 POSITION 10-INT((LEN(Q4$))/2),9:PRIN
T #6;Q4$
260 POSITION 14,5:PRINT #6;Q3$
270 POSITION 7-LEN(Q2$),5:PRINT #6;Q2$
280 PRINT :PRINT :PRINT "PUSH STICK IN T
HE DIRECTION OF"
```

ARTWORX. IT'S A WHOLE NEW WORLD OF SOFTWARE.



Scene from BETA FIGHTER during creation using the DRAWPIC graphics editor.

HODGE PODGE: by Marsha Meredith (Atari and Apple)
NOW AVAILABLE FOR ATARI!!! This captivating program is a marvelous learning device for children from 18 months to 6 years. HODGE PODGE consists of many cartoons, animation and songs which appear when any key on the computer is depressed. A must for any family containing young children.
PRICE.................... **$19.95** diskette

☐ **PM EDITOR**: by Dennis Zander (Atari, 16K)
Create your own fast action graphics game for the Atari 400 or 800 using its player missile graphics features By using player data stored as strings, players can be moved or changed (for animation) at machine language speed All this is done with string variables (PO$(Y)=SHIP4) This program is designed to permit creation of up to 4 players on the screen store them as string data and then immediately try them out in the demo game included in the program Instructions for use in your own game are included PM EDITOR was used to create the animated characters in **ARTWORX** RINGS OF THE EMPIRE and ENCOUNTER AT QUESTAR IV.
PRICE **$29.95** cassette **$33.95** diskette

☐ **ROCKET RAIDERS** by Richard Petersen (Atari 24K)
Defend your asteroid base against pulsar bombs, rockets, lasers, and the dreaded "stealth saucer" as aliens attempt to penetrate your protective force field Precise target sighting allows you to fire at the enemy using magnetic impulse missiles to help protect your colony and its vital structures
PRICE **$19.95** cassette **$23.95** diskette

☐ **FOREST FIRE TWO:** by Richard Petersen (Atari 24K)
FOREST FIRE has been enhanced and now offers a two player mode for head to head competition to see who can survive, suffer the least damage and put their fire out first. User input now determines landscape, wind and weather conditions, offering limitless game variation. FOREST FIRE's excellent color graphics have been made even better, turning your computer into a super-detailed fire scanner.
PRICE .... **$16.95** cassette **$20.95** diskette

☐ **GIANT SLALOM:** by Dennis Zander (Atari, 16K)
Bring the Winter Olympics to your computer anytime of the year! Use the joystick to guide your skier's path down a giant slalom course consisting of open and closed gates Choose from three levels of difficulty Take practice runs or compete against from two to eight additional skiers
PRICE **$15.95** cassette **$19.95** diskette

☐ **THE PREDICTOR** by Thomas Barker (Apple, Atari TRS 80 North Star and CP/M (M BASIC)
This is a complete package that covers least squares fitting of parameters for two or more variables THE PREDICTOR can be used for predicting sales and process behavior, trend analysis, model building and many other uses calling for multilinear regression techniques Each option in the program is prompted with simple YES/NO commands making it **very** easy to use
PRICE **$29.95** diskette

☐ **PILOT:** by Michael Piro (Atari, 16K)
Pilot your small airplane to a successful landing using both joysticks to control throttle and attack angle PILOT produces a true perspective rendition of the runway, which is constantly changing Select from two levels of pilot proficiency
PRICE ... .. **$16.95** cassette **$20.95** diskette

☐ **TEACHER'S PET:** by Arthur Walsh (Atari, Apple, TRS-80, PET, North Star and CP/M (MBASIC) systems)
This is an introduction to computers as well as a learning tool for the young computerist (ages 3-7). The program provides counting practice, letter word recognition and three levels of math skills.
PRICE **$14.95** cassette **$18.95** diskette

☐ **MAIL LIST 3.0:** (Atari, Apple and North Star)
The very popular MAIL LIST 2 2 has now been upgraded Version 3 0 offers enhanced editing capabilities to complement the many other features which have made this program so popular. MAIL LIST is unique in its ability to store a maximum number of addresses on one diskette (typically between 1200 and 2500 names!). Entries can be retrieved by name, keyword(s) or by zip codes They can be written to a printer or to another file for complete file management The program produces 1, 2 or 3-up address labels and will sort by zip code (5 or 9 digits) or alphabetically (by last name) Files are easily merged and MAIL LIST will even find and delete duplicate entries! The address files created with MAIL LIST are completely compatible with **ARTWORX** FORM LETTER SYSTEM
PRICE .. .. **$49.95** diskette

☐ **THE VAULTS OF ZURICH:** by Felix and Greg Herlihy (Atari, 24K, PET)
Zurich is the banking capital of the world. The rich and powerful deposit their wealth in its famed impregnable vaults But you, as a master thief, have dared to undertake the boldest heist of the century You will journey down a maze of corridors and vaults, eluding the most sophisticated security system in the world. Your goal is to reach the Chairman's Chamber to steal the most treasured possession of all: THE OPEC OIL DEEDS!
PRICE .. **$21.95** cassette **$25.95** diskette

☐ **BRIDGE 2.0** by Arthur Walsh (Atari (24K), Apple TRS-80, PET, North Star and CP/M (MBASIC) systems)
Rated #1 by Creative Computing, BRIDGE 2 0 is the only program that allows you to both bid for the contract and play out the hand (on defense or offense!). Interesting hands may be replayed using the "duplicate" bridge feature This is certainly an ideal way to finally learn to play bridge or to get into a game when no other (human) players are available
PRICE **$17.95** cassette **$21.95** diskette

☐ **ENCOUNTER AT QUESTAR IV:** by Douglas McFarland (Atari, 24K)
As helmsman of Rikar starship, you must defend Questar Sector IV from the dreaded Zentarians Using your plasma beam, hyperspace engines and wits to avoid Zentarian mines and death phasers, you struggle to stay alive. This BASIC/Assembly level program has super sound, full player missile graphics and real time action
PRICE **$21.95** cassette **$25.95** diskette

NEW PROGRAMS!

HAZARD RUN: by Dennis Zander (Atari, 16K)
The sheriff has spotted you and you must make the treacherous run through Crooked Canyon past Bryan's Pond to the jump at Hazard Creek and safety. You can even put the joystick-controlled GEE LEE car upon two wheels to make it through some tight spots. A lead foot is not always the answer as you dodge trees, rocks and chickens in this nerve-racking game. HAZARD RUN employs full use of player/missile graphics, re-defined characters and fine scrolling techniques to provide loads of fast action and visual excitement.
PRICE........**$27.95** cassette **$31.95** diskette

BETA FIGHTER: by Douglas McFarland (Atari, 16K)
See who will be the ace gunner in this action game set on a spectacular Martian landscape. BETA FIGHTER can be played with one or two players and uses player/missile graphics and delightful sound effects.
PRICE........**$16.95** cassette **$20.95** diskette

DRAWPIC: by Dennis Zander (Atari 16K)
DRAWPIC provides the user with an unbelievably easy way to create screens in graphics modes 3-7. Just sit back with your joystick and use POINT PLOT, DRAW LINE, RUBBER BAND fill and COLOR SET to create beautiful images on your Atari. Full or partial screen images are saved as string data in the program and can be instantly recalled and combined into new images using machine language subroutines. These graphic images can be easily incorporated into your own programs. The images of HODGE PODGE and the landscape of BETA FIGHTER were made using DRAWPIC.
PRICE ....**$29.95** cassette **$33.95** diskette

POKER TOURNEY: by Edward Grau (Atari 32K, Northstar)
You are entered in a high stakes Draw Poker Tournament facing six opponents including Lakewood Louie, Shifty Pete and Dapper Dan. Each has his own style of play and of bluffing. POKER TOURNEY utilizes the Joker, has true table stakes play and each hand is played based on pot odds. The Atari version's graphics and sound are superb of course (programmed by Jerry White) making POKER TOURNEY the class program of its type.
PRICE ....**$18.95** cassette **$22.95** diskette

ORDERING INFORMATION

Call ARTWORX toll-free number to order direct:
800-828-6573
In New York, Alaska, Hawaii call:
(716) 425-2833

All orders are processed and shipped within 48 hours.

Shipping and handling charges:
Within North America: Add $2.00
Outside North America: Add 10% (Air Mail)
New York State residents add 7% sales tax

Quantity Discounts:
Deduct 10% when ordering 3 or more programs

Ask for ARTWORX at your local computer store.

Write for FREE Catalogue listing more information about these and other quality ARTWORX programs.

Artworx

150 North Main Street Fairport, NY 14450

```
290 PRINT :PRINT "      CORRECT ANSWER."

300 IF STICK(0)=15 THEN POSITION 10,5:PR
INT #6;C$:GOTO 300
310 IF STICK(0)<>15 THEN POSITION 10,5:P
RINT #6;B$
315 SOUND 0,90,10,10:FOR S1=1 TO 20:NEXT
 S1
320 IF STICK(0)=14 THEN POSITION 10,3:PR
INT #6;C$
330 IF STICK(0)=11 THEN POSITION 8,5:PRI
NT #6;C$
335 SOUND 0,60,10,8:FOR S1=1 TO 20:NEXT
S1
340 IF STICK(0)=7 THEN POSITION 12,5:PRI
NT #6;C$
345 SOUND 0,0,0,0
350 IF STICK(0)=13 THEN POSITION 10,7:PR
INT #6;C$
360 IF STICK(0)=A THEN GOTO 400
370 PRINT CLEAR$:PRINT :PRINT "PLEASE TR
Y ANOTHER ANSWER"
372 FOR X=1 TO 100:NEXT X
375 FOR S1=1 TO 150:SOUND 0,200,10,8:NEX
T S1
380 FOR S1=1 TO 100:SOUND 0,230,10,8:NEX
T S1
381 SOUND 0,0,0,0
385 FOR X=3 TO 7 STEP 4:POSITION 10,X:PR
INT #6;B$:NEXT X
390 POSITION 8,5:PRINT #6;B$:POSITION 12
,5:PRINT #6;B$
395 GOTO 280
400 RESTORE (Q*2)+1
410 READ E$
420 PRINT CLEAR$;"GOOD, ";E$
425 FOR S2=1 TO 70:S3=INT(RND(0)*50)+50:
SOUND 0,S3,10,8:FOR S1=1 TO 7:NEXT S1
427 SOUND 0,0,0,0:NEXT S2
440 PRINT :PRINT "PRESS BUTTON FOR ANOTH
ER PROBLEM"
450 PRINT :PRINT "      PUSH STICK TO END
"
460 IF STRIG(0)=0 THEN GOTO 200
470 IF STICK(0)=15 THEN GOTO 460
480 PRINT :PRINT :PRINT "               G
AME OVER"
490 FOR X=1 TO 1000:NEXT X
500 GRAPHICS 0
```

Program 2: VIC Version

```
10 DD=37154:P1=37151:P2=37152
15 GOTO 200
20 DATA APPLE,PEAR,HAT,ORANGE,1
21 DATA HAT IS NOT A    FRUIT
22 DATA A,E,I,S,2
23 DATA 'S' IS NOT A    VOWEL
24 DATA SAD,BLUE,GREEN,RED,-2
25 DATA THESE WORDS ARE COLORS
26 DATA APE,FUN,ATE,APPLE,-1
27 DATA THESE WORDS     BEGIN WITH 'A'
200 PRINT"{CLEAR}"
210 Q=INT(4*RND(1)+1)
220 RESTORE
230 FOR I=1 TO Q
240 READ Q1$,Q2$,Q3$,Q4$,A,AN$
250 NEXT
260 PRINT"{RED}";TAB(11-LEN(Q1$)/2);Q1$;"{0
    5 DOWN}"
270 PRINT"{GRN}"TAB(11-LEN(Q4$)/2);Q4$;"{04
     UP}"
280 PRINTTAB(14);"{PUR}"Q3$
290 PRINT"{UP}{CYN}"TAB(7-LEN(Q2$));Q2$;"{B
    BLU}"
300 PRINT"{04 DOWN}PUSH STICK IN":PRINT"DIR
    ECTION OF THE"
310 PRINT"CORRECT ANSWER
320 PRINT"{HOME}{04 DOWN}";TAB(10);"Q"
330 GOSUB 500:IFH=0ANDV=0THEN330
340 PRINT"{HOME}{04 DOWN}";TAB(10);" "
350 PRINT"{HOME}{03 DOWN}";LEFT$("{02 DOWN}
    ",1+V);TAB(10+H);"Q"
360 IF(V*2+H)=A THEN 400
370 PRINT"{CLEAR}PLEASE TRY AGAIN"
380 GOTO 220
400 PRINT"{07 DOWN}":PRINT"{REV}GOOD, ";AN$

410 PRINT:PRINT
415 GOSUB500:IF(V*2+H)=ATHEN415
420 PRINT"{DOWN}PUSH STICK FOR ANOTHERPROBL
    EM"
430 PRINT"PUSH {RED}BUTTON{BLU} TO END"
440 GOSUB500:IF(HORV)THEN200
450 IFFB=0THEN440
460 PRINT"{DOWN}   GAME OVER"
499 END
500 REM JOYSTICK SUB
510 POKEDD,127:P=PEEK(P2)AND128
520 H=-(P=0)
530 POKEDD,255:P=PEEK(P1)
540 V=-((PAND8)=0)
550 H=((PAND16)=0)+H
560 V=((PAND4)=0)+V
570 FB=-((PAND32)=0)
580 RETURN
```

©

Learning With Computers

Teaching Johnny To Program

Glenn Kleiman
Teaching Tools: Microcomputer Services
Palo Alto, CA

I expect within the next few years someone will write a book charging many schools with a misguided approach to teaching children about computer programming. This book, probably entitled *Why Johnny Can't Program,* will claim that teaching LOGO is the only proper approach, and that teaching BASIC is almost evil. Such extreme claims have already been made. For example, Seymour Papert, a leading advocate of LOGO, has written in *Mindstorms: Children, Computers and Powerful Ideas:*

> [It is] as unacceptable for children to enter the computer culture by learning computer languages such as BASIC as it would be to confine their access to English poetry to pidgin English translations (p. 211).

BASIC is the standard language for all personal computers. LOGO is a newer language, available for Apple II and TI 99/4 computers and being developed for several others. The advocates of LOGO claim it should, and will, replace BASIC – especially for teaching children. In order to evaluate this claim, we first must consider what features we want in a computer language taught to children.

What Children Need

1. The language should make small but interesting programs easy to write. This is useful for motivating students and alleviating anxieties about programming.

2. The language should make it easy to debug programs. Beginners quickly realize the need for exactness in programming but have trouble getting all the details right. Clear error messages and other features that facilitate debugging prevent a lot of frustration in teaching and learning.

3. The language should be designed for working with pictures and words. Most people, particularly young people, would rather use computers to create pictures and dialogues than to solve mathematical problems.

4. Learning the language should provide a good basis for learning more advanced programming. The students should master concepts common to all computer languages and develop good programming techniques.

5. The language should be compatible with the ways children think, and working with the language should facilitate the development of general thinking skills.

Comparing BASIC and LOGO

Now that we have some criteria, how do BASIC and LOGO compare?

Ease of getting started. BASIC and LOGO were both designed with novice programmers in mind. In either language, you can write small programs with a few simple commands. The programs will be of different types. A typical first BASIC program performs simple mathematical operations such as converting between centigrade and fahrenheit degrees. A typical first LOGO program creates shapes such as squares and triangles.

Ease of debugging. BASIC and LOGO are both interactive or conversational languages. When using an interactive language, you can get immediate responses from the computer without having to go through any other steps (such as the compiling step required with many languages). This makes it easier to find and correct errors in programs. With either BASIC or LOGO you can stop a program at any time, ask the computer questions such as the values of variables, and then continue the program from where it was stopped. The two languages also give helpful information about exactly where errors occur.

Working with pictures. LOGO contains an excellent Turtle Graphics system which makes it easy to create pictures. The child commands a "turtle" marker to move and draw on the screen. The commands include move forward or backward a number of steps, turn left or right a number of degrees, select a pen of a specified color, and raise or lower the pen to control whether the turtle draws as it moves. Here is a simple program to draw a square:

```
TO SQUARE
  FORWARD 40
  RIGHT 90
  FORWARD 40
  RIGHT 90
  FORWARD 40
  RIGHT 90
  FORWARD 40
  RIGHT 90
END
```

This program can be condensed to:

```
TO SQUARE
  REPEAT 4 [FORWARD 40 RIGHT 90]
END
```

Turtle graphics, originally developed as part of LOGO, has since been incorporated into other languages, including Atari PILOT which I will review in next month's column. See Dave Thornburg's Friends of the Turtle column for more examples of turtle graphics.

BASIC typically uses a coordinate graphic system in which you specify the horizontal and vertical locations of things to appear on the screen. Here is a BASIC program to draw a square:

```
100 DRAW 0,0 TO 40,0
110 DRAW 40,0 TO 40,40
120 DRAW 40,40 TO 0,40
130 DRAW 0,40 TO 0,0
```

Each DRAW command draws a line between the coordinates specified.

There are many dialects of BASIC. In some the graphic commands are limited to plotting points and drawing lines. Others, such as the Radio Shack Color Computer Extended BASIC and the IBM Personal Computer BASIC, include more powerful graphic commands for drawing circles, arcs and boxes, filling areas with color, and moving pictures on the screen to create animations. The ease of creating pictures in BASIC depends to a large extent on which version is used.

Working with words. In BASIC, sets of letters are grouped together to form strings. Commands are available for testing whether strings match, for combining strings and for selecting parts of strings.

In LOGO, letters combine to form words, which are analogous to strings in BASIC. In addition, LOGO has lists. A list consists of an ordered set of words or simpler lists. That is, you can have a list of lists, or a list of lists of lists, and so on. This is very useful for working with language – a list of words forms a sentence, a list of sentences forms a paragraph, and a list of paragraphs forms a discourse. LOGO contains procedures for checking whether a list contains a specified word, testing whether two lists match, combining lists and selecting parts of lists. The commands for working with language in LOGO are far more powerful than those in BASIC.

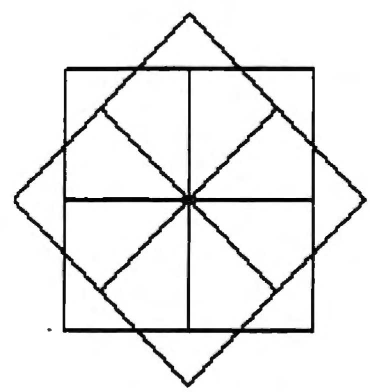
Learning general programming concepts. When learning either BASIC or LOGO, children become familiar with many concepts important in all programming. These concepts include variables, branching, iteration (repetition of sets of commands), conditionals (if/then decisions), and modules (subroutines or procedures). LOGO adds the concept of recursion (procedures that call themselves).

Learning good programming techniques. Programming techniques have advanced since BASIC was developed, and BASIC is often faulted for not encouraging good programming practices. As computers have become faster and memory less expensive, emphasis has shifted from writing programs which execute quickly and take little memory, to programs that are easy to write, understand, test, debug and modify.

An important part of good programming is dividing the overall goals of the program into simpler subgoals, each of which can be handled in its own part of the program. This is called modular programming. While modular programming is possible in BASIC, the language does not facilitate this approach.

LOGO was designed for modular programming. You define procedures, each of which tells the computer how to do something. Once a procedure is defined, it can be used in other procedures in exactly the same way as any of the "primitive" procedures built into the language. That is, each procedure you create can be used as a module in a larger program. For example, once the SQUARE procedure given above is created, it can be used in a procedure to create a more complex picture such as:

```
REPEAT 8 [SQUARE RIGHT 45]
```



Since a created procedure can be used just like a built-in one, LOGO lets you design your own language by creating a set of procedures to suit your purposes.

Compatibility with children's ways of thinking. Children are best able to understand things in terms of concrete images and their own actions, rather than in terms of abstract concepts. Turtle graphics therefore provides an excellent means of introducing programming to children. Commands to the turtle are in terms of processes children can act out. A child might create a SQUARE procedure by first drawing a square with a pencil and paper (or walking the shape of a square) and observing his own movements. The child can then tell the turtle how to follow the same pattern of movement. In contrast, the graphics commands typical of

BASIC require an understanding of the coordinate system and do not reflect the processes of drawing.

While the turtle graphics component of LOGO suits children very well, I am less certain about the rest of LOGO. The commands for working with words are powerful but may not be simple for children to use. Most teachers using LOGO have so far focused on turtle graphics. I am waiting for more information from teachers and children who are exploring the non-graphics aspects of LOGO.

In BASIC, children generally find it easy to understand the individual string processing commands and to write small programs with them. However, the difficulty of BASIC programming increases rapidly as the size of the program increases.

Developing thinking skills. To program in any language, one must analyze the task the computer is to perform, divide it into sub-steps, carefully communicate the sub-steps to the computer, and test and debug the program. Many teachers report that when children learn programming they also acquire ways of approaching many types of problems and gain an appreciation of the need for careful work. This leads to improvements in much of their school work.

I believe the critical factor in whether learning to program facilitates general thinking skills is not which language is taught, but whether the teaching encourages careful task analysis, problem solving, and testing. However, LOGO does have some advantages over BASIC. Since it encourages modular programming, it also encourages careful analysis of problems and good programming practices. LOGO contains features well adapted to certain problem solving strategies (such as progressively reducing a problem to similar but simpler problems). And working with turtle graphics can lead children to an intuitive understanding of concepts of geometry and symmetry.

Which Language Should Children Be Taught?

More people know BASIC than any other computer language and there are a greater number of books and programs available to help teach BASIC. I know many children and adults who have gotten a great deal out of learning BASIC – knowledge about computers, an understanding of general concepts of programming, appreciation of the need for careful thinking and attention to details, and joy and pride in being able to control the computer with their own programs.

LOGO has been used in a number of test projects and is beginning to be tried in many classrooms. As I've discussed, it has several excellent features for teaching children, particularly in the turtle graphics component. I expect materials to help learn and teach LOGO will become available rapidly.

Children enjoy and benefit from learning either BASIC or LOGO. Why shouldn't they learn both?

Versions Of LOGO For Apple II Computers

Two versions of LOGO are now available for Apple computers, one developed at MIT and the other developed by Logo Computer Systems, Inc. (LCSI). LCSI LOGO is marketed by Apple through its dealers. Two companies market the MIT version: Terrapin Inc. (678 Massachusetts Ave. #205, Cambridge MA 02139), and Krell Software (21 Millbrook Drive, Stony Brook, NY 11790). Both versions require a 48K Apple II with a 16K RAM card (or language card) and one disk drive. The two versions have far more similarities than differences, but each has several features not found in the other.

The MIT version has three features which are particularly useful with children: (1) The ability to change the shape of the turtle. This is useful for animations – you can create your own shape and move it on the screen with turtle commands. (2) A trace function which lets you run a program step-by-step. This is useful for carefully analyzing programs and for finding bugs. (3) The ability to save pictures to disk and retrieve them from a program. MIT LOGO also has a feature for machine language programmers: it lets you add your own machine language routines.

LSCI LOGO, while not having the features mentioned above, has several features not found in the MIT version. These include more advanced list processing capabilities, convenient ways of working with sets of procedures, the ability to re-define primitive commands, and good error trapping capability. These features are particularly useful for advanced programmers.

An important part of a LOGO package is the documentation. LCSI LOGO comes with two manuals, an excellent turtle graphics tutorial and a good reference manual. Terrapin has its own tutorial, which covers both turtle graphics and language processing. I only have a draft available for review, but it looks like a very useable manual. Krell provides a minimal amount of written documentation and a tutorial on disk. The Krell tutorial is severely lacking – it demonstrates many of the capabilities of LOGO but does not teach anyone how to use them.

The prices of the three LOGO packages, with documentation and back-up copy, are: LCSI – $175; Terrapin – $165 ($150 + $15 for back-up disk); Krell – $179. I recommend either LCSI or Terrapin. Krell cannot be recommended due to the lack of adequate documentation.

LOGO Books

Four books that tell you more about LOGO are available:

Logo by Harold Abelson (Byte/McGraw Hill Publishers, 1982). A guide to using LOGO which is a good addition to the LCSI and Terrapin manuals.

Special Technology for Special Children by Paul Goldenberg (University Park Press, 1979). Discusses uses of LOGO with severely handicapped children.

Mindstorms: Computers, Children and Powerful Ideas by Seymour Papert (Basic Books, 1980). The case for LOGO, strongly stated.

Turtle Geometry by Harold Abelson and Andrea diSessa (MIT Press, 1980). A technical book on using turtle graphics to teach concepts of plane geometry, vectors and topology and other areas.

An Inexpensive Turtle Graphics Program

Many people want to try turtle graphics without investing in the full package and the necessary memory expansion. An excellent program which lets you do so was published in *Nibble* magazine (Vol. 3, No. 1, 1982). This program, written in Applesoft by David Krathwohl, provides the main turtle graphics commands, the ability to write procedures and call them from other procedures, iteration, some minimal use of variables, and saving programs to disk. You can find a copy of the magazine and type the program into your Apple or order a disk (with documentation and two other programs) for $29.95 plus $1.50 postage from Nibble: Box 325, Lincoln MA 01773. ©

Apple Game Paddles

John K. Elberfeld
Rochester, NY

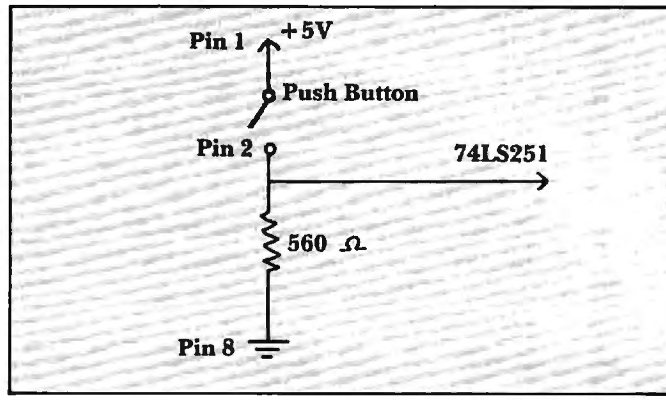
Understanding the functions of the Apple game paddles was my first step toward interfacing the Apple with the "real world." The simplicity of the paddles is a result of the amazing versatility of the game In/Out connector (where the game paddles are plugged in) on the computer mother board. The Apple Reference Manual, pages 23, 24, 100, and 114, provides the basic facts about the paddles and the connector. This article will attempt to unify these facts into an understandable explanation of the Integrated Circuit electronics used every time Little Brick Out is RUN.

The push button control on the paddles is a normally open momentary switch. No electricity may flow through this switch until it is pushed down. As soon as it is released, the flow of electricity stops. The source of voltage is pin 1 on the game I/O connector, which supplies +5 volts through the connecting wires to one side of the switch. When the button is pressed and the switch is closed, this +5 volts passes through the switch and back through the wires to pin 2 on the I/O connector. Pin 2 is connected to a standard Transistor-Transistor Logic (TTL) Integrated Circuit (IC) on the Apple mother board. When +5 volts is applied to pin 2, this IC causes the memory location of (-16287) or ($C061) to be greater than 128. The actual circuitry allows three push buttons to be used, but this article will supply details for button 0 only.

When pin 2 is grounded, the value of (-16287) will fall below 128. However, *not* applying +5 volts to pin 2 is *not* the same as grounding the pin. To make the pin grounded when no voltage is applied, a 560 ohm resistor is permanently connected between pin 2 and ground. Any electricity which might deceive pin 2 into thinking that 5 volts was being connected to it disappears through the resistor and into the ground. This resistor is located in the 16-pin DIP Header – the black box at the end of the paddle wires. This resistor is small enough to effectively ground pin 2 when no voltage is applied, but large enough so the current through it is very small when +5 volts is applied.

Figure 1 shows this information.

Figure 1. Push Button



The game controller analog inputs (knobs which turn) are physically very simple. The following details apply to PADDLE 0, but the concept is the same for all paddles. Just the pin numbers are different.

Pin 1 again supplies +5 volts through the connecting wires to one end of a 150,000 ohm variable resistor. The other end of the resistor is connected through the wires to pin 6. Turning the game control varies the resistance between the 5 volt supply voltage and pin 6 on the game I/O connector. When the resistor is turned so the resistance in the circuit is large, very little current can flow back to pin 6. A low resistance setting of the game control allows a large current to flow back to pin 6.

Pin 6 is a direct connection to a capacitor on the mother board. A capacitor is designed to store electric charge, but this capacitor starts out with *no* charge. Electric current which flows through pin 6 to the capacitor will build up the charge on the capacitor until it reaches a predetermined level. When the Apple is instructed to read the value of the game control through the BASIC command PDL(0), it uses a machine language monitor routine. This command starts a counting routine, consisting of a loop and a counter, to time how long it takes the electric current through the game controller resistor to charge the capacitor back up from its zero starting charge. The number of times that the computer executes this timing loop is the value reported when PRINT PDL(0) is RUN.

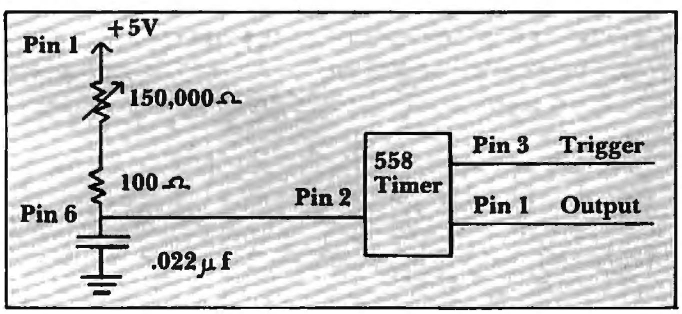
To limit the electric current flow into the capacitor when the game control is applying zero resistance, a small 100 ohm resistor is connected in series between the capacitor and the game controller. If too much current flowed into the capacitor in a very short time, the power supply or other circuits could be damaged.

How can the computer tell when a capacitor is charged? The Apple uses a 558 quad timer Integrated Circuit which is designed specifically to accomplish this. This circuit is similar to four 555 timers on one chip and allows up to four game paddles to be used at one time. The 558 IC is carefully designed to have a high output when the capacitor is charged to *less* than 2/3 of the supply voltage, and a low output after it reaches 2/3 of the supply voltage.

The timing cycle is started when the 558 is triggered – accomplished by PEEKing at location 49264 (-16272 or $C070). After the 558 is triggered, the output goes high, and the capacitor is allowed to charge up from the current through the game controller. When the capacitor reaches 2/3 of the supply voltage, the output goes low and remains low until triggered again. The 558 then automatically discharges the capacitor and waits for another trigger. Memory location $C064 is high (greater than 128) while the output of the 558 timer is high, and low (less than 128) when the output of the timer is low. How a high output from the 558 affects the value in a memory location is a good topic for some other author.

Figure 2 shows the circuit used for Paddle 0 on the Apple. The other game controllers have similar circuits but different memory locations.

Figure 2. Game Control Paddle



The Apple Reference Manual, Page 144, lists the following monitor routine for reading the value of the game controller. These directions are permanently stored in the Read Only Memory (ROM) of the Apple and are used during the execution of BASIC commands in regular programs. The following explanation assumes some knowledge of machine language programming.

```
PREAD  LDA  PTRIG
       LST  ON
       LDY  #$00
       NOP
       NOP
PREAD  LDA  PADDL0,X
       BPL  RTS2D
       INY
       BNE  PREAD2
       DEY
RTS2D  RTS
```

PTRIG is location $C070. LDA to this location triggers all 4 timers on the 558 chip on the mother board. This forces all the outputs of the 558 chip to go high and remain high until the capacitors are charged. This trigger also allows the capacitor to start charging.

LST ON is not a machine language command, but is probably a note to indicate that the timer has been triggered.

LDY #$00 loads the Y register with a value of zero to initialize it for the timing loop.

NOP means no operation. These are dummy commands used to take up some time before the timing loop is entered.

LDA PADDL0,X stores in the accumulator (A) the value of the memory location indicated by PADDL0, with the location address increased by the value stored in the X register. This allows a single routine to time all four allowable game controllers. The number of the controller (0 through 4) must be loaded in the X register before the routine is entered. PADDL0 is the memory location $C064, the memory location which is high when the current through paddle zero has not charged the capacitor. When register X has 0 stored in it, this location is read. When register X has the value "1" stored in it, the value in memory location $C064 + 1 = $C065 is loaded in the accumulator. $C065 is the memory location which is high when paddle 1 is in the process of charging its capacitor.

BPL RTS2D commands the computer to branch to location RTS2D if the value loaded into A is positive. RTS2D contains the command RTS which stops the timing loop and returns control to the BASIC commands which first called in the routine. A byte is considered negative when bit 7, the eighth bit from the right, is "1," and positive when this bit is "0." This seventh bit is also used as the flag to indicate that the output of the timer is still high. While the capacitor is in the process of being charged by the current through the game controller, this seventh bit in location $C064 is set to "1," the computer considers the value of the entire byte negative, and no branching occurs. The computer continues down the list of instructions. When the seventh bit is "0," it means the capacitor is charged enough, the timing stops with the value in Y indicating the number of times the program executed the loop, and the program branches to RTS2D which returns control back to the BASIC program. This is the normal END for this monitor subroutine.

INY commands the computer to increase the value of the Y register by 1. The Y register is now a counter which indicates how many times the computer passes this point in the loop. It is this value stored in Y which is the "value" of the game controller setting.

BNE PREAD2 commands the computer to branch back in the loop to the location of PREAD2 (the LDA PADDL0,X command) if the value of the Y register is *not* equal to zero. If the resistance is very high in game controller, the Y register may have counted up to 255, the largest number it can store. (All 8 bits are set to 1). Adding 1 more to 255 does *not* result in 256, but changes all the 1's to 0's (255 + 1 = 0). At this point the computer realizes it has gone too far, and it does not branch back to continue the loop.

DEY decreases the value of Y by one. Since Y must be zero to reach this point in the program, Y changes from 0 back to 255 because 000 - 1 = 255. This is the largest number that may be a paddle reading. At this point the timing loop is halted.

RTS then returns control to the BASIC program which called the subroutine. The value in Y is the value of the paddle reading.

The loop of:

```
READ2 LDA PADDL0,X   (5 CYCLES)
      BPL RTS2D      (2 CYCLES-NO BRANCH)
      INY            (2 CYCLES)
      BNE PREAD2     (3 CYCLES-SUCCESSFUL
                      BRANCH)
```

is repeated until the capacitor is charged or until Y reaches a reading of 256 (000). The loop uses 12 clock cycles or about 12 microseconds. The maximum amount of time this program can measure is 255 loops or 3060 microseconds. The time needed to charge sufficiently the .022 mfd capacitor on the mother board through 150,000 ohm game controller is 1.1 * 150,000 * .022 E-06 = .0036 seconds = 3600 microseconds. This extra time guarantees that the maximum value can be reached when the game controller is set for maximum resistance.

Where can this information lead? It is possible to substitute a thermistor in place of the game controller so the reading of a paddle will indicate the temperature. A game controller with a resistance differing from the 150,000 ohm Apple Controller can be adapted to this circuit. A tilt sensitive switch could replace the pushbutton to indicate changes in position. The possibilities are endless and challenging. For more information on these circuits, see:

Mims, Forrest M.
Engineer's Notebook
Radio Shack, 1979

Lancaster, Don
TTL Cookbook
Howard W. Sams, 1974

Semiconductor Reference Guide
Radio Shack, 1981

©

In this interesting article, the author, a quadriplegic, describes techniques he has learned to effectively work with his computer. His suggestions may be of interest to other handicapped programmers and, at the end, he has a program-typing hint for everyone.

Computing Techniques For The Handicapped

George Leotti
Glenolden, PA

Being a physically disabled individual I faced some unique problems in making my computer, and peripherals, usable. First, I'd like to briefly describe my disability and then pass along some hints that may help other people, disabled and able alike.

I'm a C5-C6 quadriplegic. What that basically means is that my legs are totally paralyzed. My arms are partially paralyzed. I can't move my fingers or thumbs at all, but I can flex my wrists. Now, on to the hints.

There are many devices available to hold pencils, pens, and paintbrushes that will hold a stick which may be used to press keys on your keyboard. I use a universal holder with an L bar that holds a pencil, eraser-side down. Use whatever is comfortable for you.

To protect the faces on my small keyboard PET, I have someone cut out the fingers of a surgical glove, put them over the eraser, and tape it securely to the pencil. This way the eraser will not fragment and get stuck between the keys. Plus, wear and tear on the keys is reduced.

A New On-Off Switch

The first problem I encountered when I bought my PET was how to turn it on and off by myself. As you know, the PET and Commodore peripherals have their power switches located on the back panels. Why? I don't know. But I do know it is impossible for me to reach them.

The solution was simple. I had another switch mounted on the front panel, above the keyboard. The new switch "jumpers" the power switch in the rear. The Commodore switch now stays in the on position. The new switch turns power on and off.

Its switch is a toggle switch which can be purchased at any Radio Shack, part number 275-651.

I had the same done on my 2023 printer. The switch is mounted an inch to the right of the paper advance button.

Who will install switches for you? My brother, an electronics engineer, installed them for me. If you have no one able to perform this operation for you I suggest you: contact a local electronics/computer store, appliance repair shop, or television repair shop. If you go this route be sure the person is insured and knows exactly what you need. If your machine is still under warranty, contact the manufacturer first. They (or the store where you bought it) may install a switch for you.

Another solution to the problem would be to have a switching center installed by a competent electrician. In this way you would not deface your machine and you could have all the power switches in one convenient location.

Operating Tape And Disk Drives

The next problem was how to push the PLAY and RECORD keys at the same time. I found that if you put something in the jaws of a nutcracker to space the ends the proper width, and then tape the ends to hold the spacer, you have a tool that works on the tape drive. This, when held in the mouth, makes pushing those buttons easy. Since most nutcrackers are made of metal, however, it would be easier on your teeth to contact your friendly occupational therapist for something made of plastic or wood.

Just recently, I purchased a 2031 single disk drive because of an overflow of cassettes. It is a remarkable, machine, of course, very fast and efficient, but when I tried to change its diet (remove one diskette and put in another) I found I could not pull out the diskette.

Some tape and toothpicks solved this one. On the left side (front) of my diskettes I have between five and six inches of tape folded in half and fixed to the top and bottom of each diskette with a toothpick at the fold in the tape. Now I can pull out a diskette with one finger.

I used strapping tape (tape with fibers) and round toothpicks. Instead of toothpicks you could use small plastic rings that you can put a finger, or your pushstick, through. Be sure not to cover the write-protect notch on the disk.

Shopping For Machines

If you are disabled and deciding which computer to buy, go to a computer store. Try each model or brand you are interested in. Look for accessibility. I have found that the PET has very accessible keyboards. For me, the separate numeric keypad reduces arm fatigue when entering programs manually.

Also, anyone who has a disability that affects the hands should consider the tractor feed type printer. I have the problem of my paper coming out of alignment almost every time I tear a sheet out of the printer. I have to use my mouth to grip the paper and pull it. So far, I've come up with no practical solution to this problem...

Something For Everyone

Here's a hint which might solve a general problem. When entering a program from **COMPUTE!**, do your eyes sometimes wander the page looking for your place? In the back of **COMPUTE!** you will find a card that is used for foreign subscriptions. If you live in the US, or subscribe, tear this card out.

Use a sharp hobby knife to cut out a "window" in the card. Cut out, on the address side, the words (sorry about this) **COMPUTE!** Magazine. You may need to widen it to equal a whole line of a program. Place the card on the page that has the program, leaving the lines you want, showing through the window. As you enter the program, just move the card down the page.

These ideas are offered as suggestions. Neither **COMPUTE!** nor Mr. Leotti can be held responsible for any damages resulting from their use. ©

For Commodore computers, this program allows you to add very large numbers together.

Multidigit Addition

Zoltan Szepesi
Pittsburgh

A microcomputer generally cannot handle more than 8 to 12 digits of precision. However, special programs can be written in BASIC or machine language to increase the precision of the calculations.

In this program, Multadd, the addition is made in eight-digit groups. The two numbers to be added together (the *addenda*) are entered as two strings between lines 70 to 120, using a modified version of Gary Greenberg's simulated input routine (**COMPUTE!**, May/June, 1980, #4). This way, up to 254 digits can be entered for each number. The two addenda can have different numbers of digits. Lines 140 to 160 make them equal in length by placing zeros in front of the shorter number.

Lines 170 to 250 divide the strings into eight-digit groups and transform these substrings into numeric values. Then lines 260 to 300 do the addition, and 310 to 380 reconvert the groups to eight-digit strings, taking care to fill up the empty places with zeros, and eliminating the spacing between the groups. The result is printed out as a continuous string.

```
10 PRINT"{CLEAR}           MULTIDIGIT ADDITI
   ON"
40 PRINT"{DOWN}THE NUMBERS TO BE ADDED CAN
     HAVE MAX."
50 PRINT"254 DIGITS.TYPE THE DIGITS CONTIN
   UALLY, THEN PRESS 'RETURN'."
60 PRINT"CORRECT WRONG NUMBER WITH 'DEL'."

70 PRINT"{DOWN}        FIRST ADDENDUM:A=":PR
   INT"{RIGHT}";
80 GOSUB 500
90 A$="0"+Z1$
100 PRINT:PRINT"       SECOND ADDENDUM:B=":P
    RINT"{RIGHT}";
110 GOSUB 500
120 B$="0"+Z1$
130 T1=TI:PRINT:PRINT"          {REV}WORKING
    "
140 LA=LEN(A$):LB=LEN(B$):D$=""
150 IF LB<LA THEN LD=LA-LB:FOR I=1 TO LD:D$
    =D$+"0":NEXTI:B$=D$+B$
160 IF LA<LB THEN LD=LB-LA:FOR I=1 TO LD:D$
    =D$+"0":NEXTI:A$=D$+A$:LA=LEN(A$)
170 M=INT(LA/8):Q=LA-8*M:IF Q>0 THEN M=M+1
180 DIM A(M),B(M),C(M),D(M),R(M),R$(M),F$(M
    )
190 FOR I=0 TO M:D(I)=0:C(I)=0:R(I)=0:NEXT
    I
200 IF M=1 THEN A(1)=VAL(A$):B(1)=VAL(B$):G
    OTO260
210 IF Q>0 THEN A(1)=VAL(LEFT$(A$,Q)):B(1)=
    VAL(LEFT$(B$,Q)):X=1:GOTO 230
220 A(1)=VAL(LEFT$(A$,8)):B(1)=VAL(LEFT$(B$
    ,8)):X=0
230 FOR I=2 TO M
240 A(I)=VAL(MID$(A$,(I-1-X)*8+1+Q,8)):B(I)
    =VAL(MID$(B$,(I-1-X)*8+1+Q,8))
250 NEXT I
260 FOR I=M TO 1 STEP -1
270 C(I)=A(I)+B(I)+D(I)
280 D(I-1)=INT(C(I)/1E8)
290 R(I)=C(I)-D(I-1)*1E8
300 NEXT I
310 PRINT:PRINT"  THE SUM:A+B=":R$(1)=STR$(
    R(1)):IF R(1)=0THENR$(1)="{RIGHT}"

320 PRINT R$(1);:IFM=1 GOTO390
330 FOR I=2 TO M:R$(I)=STR$(R(I))
340 P=LEN(R$(I)):R$(I)=RIGHT$(R$(I),P-1):F$
    (I)=""
350 FOR J=8 TO 2 STEP -1
360 IF P<=J THEN F$(I)=F$(I)+"0":NEXTJ
370 R$(I)=F$(I)+R$(I)
380 PRINT R$(I);:NEXT I:PRINT
390 PRINT:PRINT"          EXECUTION TIME:";IN
    T((TI-T1)*100/60+.5)/100;"SEC"
400 PRINT:PRINT"DO YOU WANT TO CONTINUE? (Y
     OR N)"
410 GET V$:IF V$=""GOTO 410
420 IF V$="Y" THEN CLR:PRINT"{CLEAR}{DOWN}"
    :GOTO 70
430 IF V$<>"N" GOTO 400
440 IF V$="N" GOTO620
500 Z$="":Z1$=""
510 PRINT "${LEFT}";
520 GET Z$:IF Z$="" GOTO 510
530 PRINT " {LEFT}";
540 IF Z$<>CHR$(20) GOTO 580
550 IF Z$="" GOTO 510
560 ZZ=LEN(Z1$):IF ZZ<1 GOTO 510
570 Z1$=LEFT$(Z1$,ZZ-1):PRINT "{LEFT}";:GOT
    O510
580 IF Z$=CHR$(13) OR Z$=CHR$(141) GOTO 610
590 PRINT Z$;
600 Z1$=Z1$+Z$:GOTO 510
610 RETURN
620 END
```

©

For PET/CBM or Atari disk users, any memory size. This is a handy program if you ever need to update files.

A Direct Access File Editor

Charles Brannon
Editorial Assistant

What's in a file? A payroll program might keep a list of employees, social security numbers, status, etc., on a disk file. A check balancing program would probably save the information gathered during the run of the program such as check number, payee, and amount paid on a file.

Most files are very similar – a list of numbers and strings (words, names, etc.). Each line of the file is "sent" to the file with some variant of the PRINT statement, such as PRINT#1,A$ or PRINT#1, ZX(I,J). [*PRINT#1; A$ on Atari.*] Each line (also called a *field*) ends with a carriage return.

This carriage return is used to separate each line in the file. Otherwise, the file would be just a long block of characters, each line running into the next. These separators are called *delimiters*. On many computers, the comma and colon can also be used (but can cause trouble) as delimiters. This is important, as we will see later.

A File Editing Tool

You probably use files every day, even if you're not a programmer. If you use any commercial software such as a word processor, or a prepackaged accounting program, you are unknowingly generating files. Such software writes files to the disk to save the results of calculations and your input, then later *reads* the file, when you run the program anew, to "restore" the original conditions and information.

Unfortunately, few programs let you *edit* this information (excepting a word processor, of course). You may have a check-balancing program that writes to disk all checks that you enter in order to use this information for taxes at the end of the year. This file simply accumulates information. If you later discover that you've made a mistake on a check entered, there is no easy way to find or correct that check. What you need is a program that can read the entire file into memory, let you view it and make selective changes, then rewrite the corrected file back out to the disk.

The program here is a *file editor*. Basically, it asks you for the name of the file you want to edit, reads it in, and lets you "page" through the file 19 lines at a time. You can edit, replace, insert, and delete lines, as well as search the file for any sequence of characters.

This program only works with a file organized in the common "list" structure previously mentioned (which limits it to *sequential* files on the PET/CBM).

The two versions presented here, for the Atari and the PET/CBM, are different, but they both perform similar functions. After you enter the file name, the file is read into a string array. If the file is too large to fit into memory, the message "FILE TOO LONG" will appear. You can change the variable MAX (maximum number of lines in file) if you have enough memory. The file is displayed as numbered lines in the top 19 lines of the screen. Your input is taken at the bottom. When prompted for a line, always enter the *number* of the line.

Miscellaneous

If you enter any command that you don't want to perform (such as Replace when you meant to type Edit), you can usually exit the command with a null input (just press <RETURN>) on the PET/CBM, or by pressing the ESC (escape) key on the Atari.

Since commas and colons are used as delimiters on the PET/CBM, you can't enter or edit a line containing commas. To enter such a line, hold down SHIFT when you press the command key such as SHIFT-E for Edit, SHIFT-I for Insert, etc. A quote will automatically be printed at the start of the line, permitting you to enter commas and colons without the infamous ?EXTRA IGNORED message. After you enter the line, all commas (CHR$(44)'s) are converted to "fake commas" that look like commas, but don't work as delimiters (CHR$(108)).

Create A "Fake File"

You could even use this program to *create* files if you can create a "fake file" containing only one item such as:

```
OPEN 1,8,8,"0:filename" (on the PET)
PRINT#1,"X":CLOSE1
OPEN#1,8,0,"D:filename" (Atari)
PRINT#1;"X":CLOSE#1
```

You can then Replace the first line (the "X"), and use Append to add to the file.

I'll leave you with one important reminder: remember to re-SAVE a file after you edit it or your changes will have been in vain.

Editing Commands

PET/CBM	Atari	Command	Description
<RETURN>	N	Next page	Displays the next 19 lines.
E	E	Edit	The primary "change" command. You enter the number of the line to be changed. The line is displayed at the bottom of the screen with the cursor on the first character. You can use your computer's editing keys to change the line (just like in an INPUT statement). Press <RETURN> when finished.
R	R	Replace	You just enter a completely new line to replace any line in the file.
I	I	Insert Line	For Atari: you enter the number of the line you want to insert a new line at. That line and all following lines are "pushed down," and the word "*** INSERT ***" is inserted. You can then use Replace to add a new line. For PET/CBM: you enter the number of the line you want to insert a line at, and then you enter the line you want inserted.
D	D	Delete Line	Enter the number of the line to be deleted. The following lines will be "pulled up" to fill the gap. For PET/CBM: If you use SHIFT-D (shifted D), you can enter the starting and ending lines of a block of lines you'd like deleted.
A	A	Append Line	After you type "A" you can enter a new line which will be added to the end of the file.
P	P	Print	Enter the starting and ending lines of the block of lines you would like printed to the printer. For the PET/CBM, this is device number four. You can just hit <RETURN> to both prompts to print the entire file (Atari version).
S	W	Save/Write	Stores the file on disk. PET/CBM: scratches the old file, then writes the new file. Atari: Replaces the old file, unless you save the file under another name.
X	Q	Exit/Quit	Lets you end the program.
0/1	[START]	Re-run	Atari version: If you press the START key at the prompt "Which?" you can RUN the program over again. PET/CBM version: Entering zero or one specifies which drive to use for the next file you want to edit. You can then enter the new file name.
G	G	Go to line	Lets you "skip around" in the file. Enter the number of the line you want to see.
n/a	O	Other menu	There are two menus in the Atari version, since the bottom of the screen can only hold so many commands. The other menu contains the commands Length, Search, and Next Search. You can add your own commands to this menu.
H	n/a	Help	Unlike the Atari version, the PET/CBM version doesn't constantly display all choices. Just type H to see a list of all options.
K	n/a	Kill File	Lets you scratch a file from within the program.
Z	n/a	—	Flips the character set.
F Q	S* N*	Find/Search Quick Find Next Search **Use (O)ther Menu*	You enter a series of characters you want to find in the file. In the PET/CBM version, all matching lines are displayed. Press any key to abort the search. The Atari version only displays the first occurrence of the match. The Next Search command is used to find the next occurrence of the same search string. You can search either by character or by line. The character search will search the entire file character-by-character for the string, where the line search only matches the search string with the beginning of the line, and is naturally faster. The PET/CBM version has a separate command, Quick Find, to search by line.
L	L	Length	Displays the number of lines in the file.

Program 1. PET/CBM Version

```
100 PRINT"{CLEAR}{REV}"TAB(15-20*(PEEK(213)
    =79))"*FIXFILE*"
110 PRINTCHR$(14);:POKE59468,12
120 MAX=1000:REM MAXIMUM NUMBER OF LINES AL
    LOWED
130 DIMA$(MAX):REM CHANGE TO SUIT MEMORY/FI
    LE REQUIREMENTS
140 PRINT"{DOWN}ON WHICH DISK WILL WE WORK?
    ";
150 GETD$:IFD$<"0"ORD$>"1"THENK=1-K:PRINTMI
    D$("{REV}{OFF}",K+1,1);" {LEFT}";:
    GOTO150
160 PRINT"{OFF}"D$
170 OPEN15,8,15:FILE$="   *"
180 INPUT"WHAT IS THE FILE'S NAME?_*{03 LEF
    LEFT}";F$:IFF$="*"THENF$=MID$(FI$,
    3)
190 FIL$=D$+":"+F$
200 OPEN8,8,8,FIL$+",S,R":J=0
210 INPUT#15,EN,EM$
220 IFENTHENPRINTEM$:GOTO1420
```

```
230 CLOSE0:REM RESETS ST TO 0
240 IFST=64THEN280
250 J=J+1:INPUT#8,A$(J)
260 IFJ<MAXGOTO240
270 PRINT"{DOWN}FILE TOO LARGE":CLOSE8:END
280 CLOSE8
290 K=0:SK=1
300 PRINT"{CLEAR}":K=SK-1
310 K=K+1:IFK>JTHENK=0:GOTO360
320 IFPEEK(216)=1THENSK=K
330 N=K:GOSUB1300:PRINTN$;A$(K)
340 IFPEEK(216)>=20THEN360
350 GOTO310
360 PRINT:PRINT"COMMAND? ";
370 GETCM$:IFCM$=""THENZ=1-Z:PRINTMID$("{RE
    REV}{OFF}",Z+1,1);" {LEFT}";:GOTO3
    70
380 SH=0:CM=ASC(CM$):IF(CMAND127)<>CMTHENSH
    =1:CM$=CHR$(CMAND127)
390 PRINT"{OFF}";CM$
400 IFCM$=CHR$(13)THENPRINT"{CLEAR}":GOTO31
    0
410 IFCM$<>"I"THEN460
420 NS=1:PRINT"{UP}INSERT AFTER WHICH ITEM"
    ;:GOSUB1310:IN=VAL(IN$):IFRTTHEN45
    0
430 J=J+1:FORI=JTOIN+2STEP-1:A$(I)=A$(I-1):
    NEXT
440 PRINT"ENTER NEW LINE:":GOSUB1310:IFRT=0
    THENA$(IN+1)=IN$
450 GOTO300
460 IFCM$="X"THENPRINT"{CLEAR}":GOTO830
470 IFCM$<>"K"THEN510
480 INPUT"{UP}ARE YOU SURE?__{03 LEFT}";X$:
    IFX$<>"Y"THEN290
490 PRINT"{CLEAR}NOW KILLING THE FILE...":P
    RINT#15,"S"+FIL$
500 PRINT FIL$" HAS NOW BEEN SCRATCHED.":GO
    TO1420
510 IFCM$<>"D"THEN590
520 NS=1:IFSH=0THENPRINT"{UP}DELETE WHICH I
    TEM";:GOSUB1310:DL=VAL(IN$):D2=0
530 IFSH=0ANDDL=0THEN300
540 IFSH=1THENINPUT"{UP}DELETE WHICH ITEMS?
    __{03 LEFT}";IN$,D2:DL=VAL(IN$)
550 IFDL=0THEN300
560 Z=1:DI=D2-DL:IFDI<0THENDI=1:Z=0
570 J=J-DI-Z:FORI=DLTOJ:A$(I)=A$(I+DI+Z):NE
    XT
580 GOTO300
590 IFCM$<>"S"THEN640
600 PRINT#15,"S"+FIL$
610 OPEN8,8,8,FIL$+",S,W"
620 FORI=1TOJ:PRINT#8,A$(I);CHR$(13);:NEXT:
    CLOSE8
630 GOTO300
640 IFCM$<>"G"THEN680
650 PRINT"{UP}GOTO WHICH LINE";:NS=1:GOSUB1
    310:IFRTTHEN300
660 L=VAL(IN$):IFL<1ORL>JTHEN640
670 K=L-1:PRINT"{CLEAR}":GOTO310
680 IFCM$<>"R"THEN730
690 PRINT"{UP}REPLACE WHICH ITEM";:NS=1:GOS
    UB1310:R=VAL(IN$):IFRTTHEN300
700 IFR<1ORR>JTHEN690
710 PRINT"ENTER NEW LINE:":GOSUB1310:IFRT=0
    THENA$(R)=IN$
720 GOTO300
730 IFCM$<>"P"THEN800
740 IFSHTHENFR=1:TU=J:GOTO760
750 DN=4
760 INPUT"FROM LINE,TO LINE";FR,TU
770 IF(FR<1ORTU<1)OR(FR>JORTU>J)OR(FR>TU)TH
    EN750
780 OPEN1,DN:FORI=FRTOTU:N=I:GOSUB1300:PRIN
    T#1,N$;A$(I):NEXT:CLOSE1
790 GOTO300
800 IFCM$<>"1"ANDCM$<>"0"THEN840
810 PRINT"{CLEAR}":IFCM$<>D$THEND$=CM$:GOTO
    180
820 GOTO180
830 GOTO1420
840 IFCM$<>"F"THEN920
850 PRINT"{UP}FIND WHAT";:GOSUB1310:S$=IN$:
    PRINT"{CLEAR}":Z=0:IFRTTHEN300
860 FORI=1TOJ:FORK=1TOLEN(A$(I))
870 IFMID$(A$(I),K,LEN(S$))=S$THENN=I:GOSUB
    1300:PRINTN$;A$(I):Z=1:GOTO890
880 NEXTK
890 GETA$:IFA$<>""THENI=J
900 NEXTI:IFZ=0THENPRINT"NOT FOUND."
910 GOTO360
920 IFCM$<>"E"THEN970
930 PRINT"{UP}EDIT WHICH LINE";:NS=1:GOSUB1
    310:L=VAL(IN$):IFRTTHEN300
940 IFL<1ORL>JTHEN930
950 V=1:Z$=CHR$(SH*29):PRINT"? ";Z$;A$(L);"
    {UP}":GOSUB1310:IFRT=0THENA$(L)=IN
    $
960 GOTO300
970 IFCM$="L"THENPRINT"FILE LENGTH:";J;"LIN
    ES.{DOWN}":GOTO1290
980 IFCM$="Q"THEN1430
990 IFCM$<>"A"THEN1020
1000 PRINT"ENTER NEXT LINE:":GOSUB1310:IFRTT
     HEN300
1010 J=J+1:A$(J)=IN$:GOTO300
1020 IFCM$<>"H"THEN1270
1030 UL=PEEK(59468):POKE 59468,12
1040 PRINT"{CLEAR}"TAB(7)"{REV}LIST OF FIXFI
     LE COMMANDS{DOWN}"
1050 PRINT"A = APPEND"
1060 PRINT"D = DELETE"
1070 PRINT"D = (SHIFT) DELETE RANGE"
1080 PRINT"E = EDIT A LINE"
1090 PRINT"F = FIND A STRING"
1100 PRINT"G = GO TO LINE #"
1110 PRINT"I = INSERT LINE"
1120 PRINT"K = KILL (SCRATCH) FILE"
1130 PRINT"L = PRINT LENGTH OF FILE"
1140 PRINT"P = PRINT FILE TO PRINTER"
1150 PRINT"Q = QUICK-FIND"
1160 PRINT"R = REPLACE LINE"
1170 PRINT"S = SAVE FILE (DON'T FORGET!)
1180 PRINT"X = EXIT FIXFILE"
1190 PRINT"Z = SWITCH UPPER/LOWERCASE
1200 PRINT"0 = EDIT A FILE FROM DRIVE 0"
1210 PRINT"1 = EDIT A FILE FROM DRIVE 1
1220 PRINT"{DOWN}HOLD DOWN SHIFT WITH COMMAN
     DS":PRINT"A,E,F,I,Q,R"
1230 PRINT"TO ENTER COMMAS OR COLONS."
1240 PRINT"PRESS A KEY TO CONTINUE..."
1250 GETA$:IFA$=""THEN1250
1260 POKE59468,UL:GOTO300
1270 IFCM$="Z"THENPOKE59468,12+2*(PEEK(59468
     )=12):PRINT"{02 UP}";:GOTO360
1280 PRINT"{REV}TRY{OFF} A,D,E,F,G,I,K,L,P,Q
     ,R,S,X,Z,1 OR 0":PRINT"OR {REV}H{O
     OFF} FOR HELP"
1290 PRINT"{03 UP}COMMAND?     {UP}";:GOTO360

1300 N$="{REV}"+RIGHT$("     "+MID$(STR$(N),2
     ),4)+"{OFF} ":RETURN
1310 RT=0:Z$=CHR$(160+SH*126*(NS=0))
```

The Hard & Soft of It: ATARI from ASAP

If you're into Atari, get into ASAP. When it comes to Atari, ASAP has it all — computers, a full line of accessories, and one of the most complete lines of software on the market. Whether you use your Atari for work or play, make ASAP your source. Call today.

The **Atari® 800™** Computer features color graphics and English characters with truly high resolution, high quality sound, expandable memory and sleek modular appearance.

Atari® 400™ — 16K also available: **$349.00.**

OPTIONAL ACCESSORIES	PRICE
ATARI® 410™ Program Recorder	80.00
ATARI® 810™ Disk Drive	455.00
ATARI® 822™ Thermal Printer	299.00
ATARI® 820™ 40-column Dot Matrix Impact Printer	279.00
ATARI® 825™ 80-column Dot Matrix Impact Printer	625.00
ATARI® 830™ Acoustic Modem	159.00
ATARI® 850™ Interface Module	175.00
ATARI® Paddle (CX30-04) and Joystick (CX40-04) (Pair)	17.95
ASAP 16K RAM Module	55.00

COMPLETE SOFTWARE LIBRARY INCLUDES THESE POPULAR UNITS:

Atari®

- Video Easel ROM 25.00
- Music Composer ROM ... 42.00
- Assembler/Editor ROM .. 45.00
- Mortgage Loan Analysis Cassette 15.95
- Stock Analysis Disk 19.95
- Stock Charting 22.95
- Bond Analysis Disk 22.95
- Mailing List Cassette 19.95
- Touch Typing (2 Cass) ... 19.95
- Graph It (2 Cass) 17.95
- Word Processor 119.00
- Personal Finance 64.95
- Microsoft Basic 75.00
- Basketball ROM 27.00
- Super Breakout ROM 30.00
- Computer Chess ROM ... 32.00
- 3D Tic Tac Toe ROM 25.00
- Star Raiders ROM 36.00
- Kingdom Cassette 12.95
- Blackjack Cassette 12.95
- Biorhythm Cassette 12.95
- Energy Czar Cassette 12.95
- Telelink ROM 24.00
- Space Invaders ROM 30.00
- Scram 18.95
- Asteriods 32.00
- Missile Command 32.00
- Caverns of Mars Disk ... 30.00
- Centipede Disk 35.00
- Pac-Man 35.00

Arcade Plus

- Ghost Hunter Cassette ... 25.00
- Ghost Hunter Disk 30.00

Datasoft

- Atari Mailing List Disk ... 19.95
- Atari Character Generator Disk 16.95
- Text Wizard Disk 89.95
- Micropainter Album 1 Disk 16.95
- Micropainter Album 2 Disk 16.95
- Le Stick Accessory 30.00

On-Line Systems

- HI-RES Adv #0 — Mission: Asteroid Disk 19.95
- HI-RES Adv #2 — Wiz & Princess Disk 25.00
- Roadwork Disk 29.95
- Jawbreaker Disk 25.00
- Softporn Adventure (X-rated) Disk 25.00
- The Next Step Disk 29.95

Personal Software

- Visicalc Disk 169.00

FOR HOME ENTERTAINMENT SYSTEMS:

Activision

- Dragster Cartridge 18.50
- Boxing Cartridge 18.50
- Checkers Cartridge 18.50
- Fishing Derby Cartridge.. 18.50
- Skiing Cartridge 18.50
- Bridge Cartridge 23.50
- Tennis Cartridge 18.50
- Laser Blast Cartridge 18.50
- Freeway Cartridge 18.50
- Kaboom! Cartridge 18.50
- Stampede Cartridge 18.50
- Ice Hockey Cartridge 23.50

asap computer products, inc.

1198 E. Willow St., Signal Hill, CA 90806

ALL PRICES SUBJECT TO CHANGE WITHOUT NOTICE. CALL FOR BEST PRICE

ASAP offers a 15-day buyer protection policy: full money-back guarantee if not totally satisfied.

Ordering Information: name, address, phone; ship by: UPS or Mail. Shipping charge: add $2.90 up to 1 lb. (UPS blue). U.S. Mail add $1.50 (U.S. only) ($25.00 minimum order).

Terms: We accept cash, check, money orders, Visa and Master Charge (U.S. funds only). Tax: 6% Calif. res., COD's and terms available on approval (School PO's Accepted).

Toll free outside California: **(800) 421-7701.** Inside California: **(213) 595-6431 (714) 891-2663**

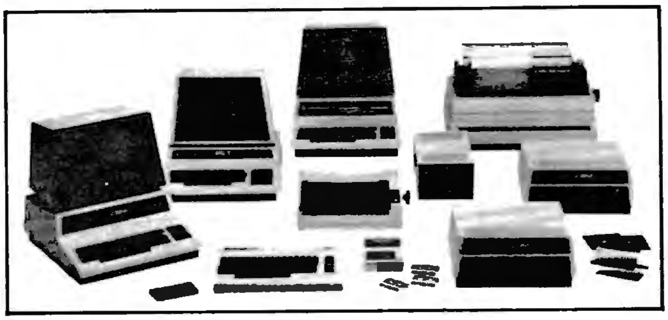
Inside Canada call **ASAP Computer Products, LTD.,** 116 Viceroy Rd., D-12 Concord, Toronto, Ontario, Canada L4K 1A9 **(416) 738-0500 (800) 268-1996**

```
1320 IFV=1ANDSH=0THENZ$="{RIGHT}":V=0
1330 PRINT"? ";Z$;:POKE205,0:PRINT"{03 LEFT}
     ";
1340 IFNS=0ANDSH=1THENPOKE158,1:POKE623,ASC(
     "{RIGHT}")
1350 INPUTIN$
1360 IFIN$=Z$THENNS=0:RT=1:RETURN
1370 IFSH=0ORNS=1THENNS=0:RETURN
1380 FORI=1TOLEN(IN$):IFMID$(IN$,I,1)<>","TH
     EN1410
1390 REM CHR$(108) IS FAKE COMMA.  SAVEABLE
     TO DISK
1400 IN$=LEFT$(IN$,I-1)+CHR$(108)+MID$(IN$,I
     +1)
1410 NEXT:RETURN
1420 CLOSE8:CLOSE15:END
1430 PRINT"{UP}QUICK-FIND WHAT";
1440 FOUND=0:GOSUB1310:IFRTTHEN300
1450 S$=IN$:L=LEN(S$):PRINT"{CLEAR}";
1460 FORI=1TOJ:IFLEFT$(A$(I),L)<>S$THEN1490
1470 N=I:GOSUB1300:PRINTN$;A$(I):FOUND=1
1480 IFPEEK(216)>20THENWAIT158,1:GETA$:PRINT
     "{CLEAR}"
1490 GETA$:IFA$<>""THENI=J
1500 NEXT
1510 IFFOUND=0THENPRINT"NOT FOUND."
1520 GOTO360
```

Program 2. Atari Version

```
100 REM  FIXFILE-File Editor v1.0
110 REM Charles Brannon, 1982
120 REM
130 GRAPHICS 0:SETCOLOR 4,9,4
140 DLIST=PEEK(560)+256*PEEK(561)+4
150 POKE DLIST-1,7+64:POKE DLIST+2,6
160 POSITION 6,0:? "FIXFILE":POSITION 3+
20,0:? "a file editor"
170 DIM TEMP$(36),FILE$(14)
180 MAX=200:REM Maximum number of lines
in the file
190 DIM DISK$(MAX*35),LN(MAX),KEY$(35)
200 OPEN #1,4,0,"K:":POKE 752,1
210 ? :? "(Press |RETURN| alone for dire
ctory.)":?
220 ? :? "Enter Filename - D:
 {12 LEFT}";
230 GOSUB 1340:SETCOLOR 0,10,12
240 IF TEMP$=" " THEN GOSUB 1080:RUN
250 FILE$="D:":FILE$(3)=TEMP$
260 ? :? "{2 UP}Reading ";FILE$,
270 TRAP 280:OPEN #2,4,0,FILE$:GOTO 290
280 CLOSE #2:? :? "{UP}Error - ";PEEK(19
5),,:GOTO 220
290 TRAP 350:LINES=0
300 INPUT #2;TEMP$
310 L=LEN(TEMP$):LN(LINES)=L
320 DISK$(LINES*35+1,LINES*35+L)=TEMP$
330 LINES=LINES+1:IF LINES<MAX THEN 300
340 ? :? "{UP}Error - File too Large!";:
GOTO 0
350 IF PEEK(195)<>136 THEN 280
360 CLOSE #2
370 CURR=0:POKE 82,0:LINES=LINES-1
380 GRAPHICS 0:SETCOLOR 4,9,4:POKE 752,1
:POKE 766,1
390 FOR I=CURR TO CURR+18
400 IF I>LINES THEN POP :GOTO 450
410 TEMP$="000":TEMP$(4-LEN(STR$(I)))=ST
R$(I)
420 FOR J=1 TO 3:TEMP$(J,J)=CHR$(ASC(TEM
P$(J))+128):NEXT J
430 ? TEMP$;" ";DISK$(I*35+1,I*35+LN(I))

440 NEXT I
450 POKE 766,0:POSITION 0,19:? "{40 R}";

460 POKE 703,4
470 ? "{CLEAR}|E|dit |R|eplace |I|nsert
|D|elete |A|ppend |P|rint |W|rite |Q|uit
 |N|ext Page |G|oto |O|ther |START|"
480 ? :? "Which? ";
490 IF PEEK(53279)=6 THEN 1300
500 IF PEEK(764)=255 THEN 490
510 P=PEEK(702):POKE 702,64:GET #1,A:POK
E 702,P
520 IF A=ASC("O") THEN 1450
530 IF A<>ASC("E") THEN 580
540 ? "{CLEAR}|EDIT|: Which Line? ";:GOS
UB 1200:IF N<0 OR N>LINES THEN 540
550 ? "{CLEAR}Enter changes, press |RETU
RN|"
560 ? "    ";DISK$(N*35+1,N*35+LN(N)):?
"{UP}";:GOSUB 1180
570 GOTO 610
580 IF A<>ASC("R") THEN 620
590 ? "{CLEAR}|REPLACE|: Which Line? ";:
GOSUB 1200:IF N<0 OR N>LINES THEN 590
600 ? "Enter new line:":GOSUB 1180
610 LN(N)=LEN(TEMP$):DISK$(N*35+1,N*35+L
N(N))=TEMP$:GOTO 380
620 IF A<>ASC("I") THEN 670
630 ? "{CLEAR}Insert a line at line: ";:
GOSUB 1200:IF N<0 OR N>LINES THEN 630
640 ? "Patience...":LINES=LINES+(LINES<M
AX):DISK$(LINES*35)=" "
650 FOR I=LINES TO N+1 STEP -1:TEMP$=DIS
K$((I-1)*35+1,I*35):DISK$(I*35+1,I*35+35
)=TEMP$:LN(I)=LN(I-1):NEXT I
660 DISK$(N*35+1,N*35+14)="*** INSERT **
*":LN(N)=14:GOTO 380
670 IF A<>ASC("D") THEN 730
680 ? "{CLEAR}Delete which line? ";:GOSU
B 1200:IF N<0 OR N>LINES THEN 680
690 IF N=LINES THEN LINES=LINES-(LINES>0
):GOTO 380
700 ? "Patience...":LINES=LINES-(LINES>0
):DISK$(LINES*35+71)=" "
```

SJB DISTRIBUTORS. THE MOST COMPETITIVE PRICES ON COMMODORE PRODUCTS.



commodore

AUTHORIZED SALES AND SERVICE FOR THE FOLLOWING PRODUCTS —

HARDWARE:

SuperPET SP9000, 96K	$1,599	(New)
CBM 8032 Computer, 80 Col.	1,069	
CBM 8050 Disk Drive, 1 Mb	1,299	
CBM 8250 Disk Drive, 2 Mb	1,760	(New)
CBM D9060 Hard Drive, 5 Mb	2,795	(New)
PET 4032 Computer, 40 Col.	950	
CBM 4040 Disk Drive, 340K	969	
CBM 2031 Disk Drive, 170K	525	
CBM Memory Expansion, 64K	399	
CBM 4022 Printer, Graphics	595	
CBM 8023P Printer, 150 CPS	769	
CBM 8300P Printer, Ltr. Qual.	1,750	
Commodore 64 Computer, 64K	595	(New)
VIC 20 Computer, 5K	259	
VIC Datasette Recorder	69	
VIC 1540 Disk Drive, 170K	467	(New)
VIC 1515 Printer, 30 CPS	320	
VIC 8K Memory Expansion	50	

MONITORS:

Panasonic CRT, 13" Color	$ 405
Panasonic CRT, 9" B & W	144

PRINTER PAPER:

9½" x 11" Continuous, Blank 2700 Sheets, Carton	$ 32

SOFTWARE:

OZZ Data Base System (8050)	$ 289	
Silicon Office	995	(New)
Wordcraft 80	289	
Wordcraft Ultra	429	(New)
VisiCalc	199	
Personal Tax Calculator	59	
Dow Jones Portfolio Management System (RS232)	120	
WordPro 4+ or 5	299	
The Manager	199	
Galaxy One	39	
Legal Time Accounting	449	
I.R.M.A.	380	
B.P.I. A/R, G/L, Job Cost, Inventory, Payroll	325/pkg	
VIC Games Cartridges	23 ea	
VIC Cassette Recreation Pack	44	
VIC Cassette Calculation Pack	44	

DISKETTES:

Verbatim #18158, 5¼", SS/DD, Hub Ring, Box of 10	$ 30
3M "Scotch" #744-0, 5¼", SS/SD, Hub Ring, Box of 10	33
Memorex #3481, 5¼", SS/DD, Hub Ring, Box of 10	28

SJB carries many other lines of equipment and products. Call us for a free catalogue.

REGISTER FOR OUR NEW SOFTWARE SUPPORT SERVICE!

ORDER TOLL-FREE

800-527-4893 (Outside Texas)
800-442-1048 (Within Texas)

MASTERCARD, VISA, MONEY ORDER, BANK CHECK, COD (ADD $5) ACCEPTED.
Add 3% Surcharge For Credit Cards.
In Stock Items Shipped Within 48 Hours, F.O.B., Dallas, Texas.
All Products Shipped With Manufacturer's Warranty.

SJB DISTRIBUTORS, INC.
10520 Plano Road
Suite 206
Dallas, Texas 75238
(214) 343-1328

Prices are subject to change without notice.

```
710 FOR I=N TO LINES:TEMP$=DISK$((I+1)*3
5+1,(I+1)*35+35)
720 DISK$(I*35+1,I*35+35)=TEMP$:LN(I)=LN
(I+1):NEXT I:GOTO 380
730 IF A<>ASC("A") THEN 790
740 IF LINES=MAX THEN ? "{CLEAR}Not enou
gh room!":FOR W=1 TO 200:NEXT W:GOTO 470

750 CURR=(LINES-17)*(LINES>18)
760 ? "{CLEAR}Enter line to be appended:
":GOSUB 1180:IF TEMP$=" " THEN 470
770 LINES=LINES+1
780 LN(LINES)=LEN(TEMP$):DISK$(LINES*35+
1,LINES*35+LN(LINES))=TEMP$:GOTO 380
790 IF A<>ASC("P") THEN 870
800 ? "{CLEAR}|PRINT|: Turn on printer,
press |RETURN|":GET #1,A:IF A=27 THEN 47
0
810 TRAP 820:CLOSE #2:OPEN #2,8,0,"P:":G
OTO 830
820 ? "{CLEAR}{BELL}Printer not ready.":
? "{DOWN}Press |RETURN|":GET #1,A:CLOSE
#2:GOTO 470
830 ? "Starting Line?";:GOSUB 1200:S=N
840 ? "Ending Line?";:GOSUB 1200:IF N=0
THEN N=LINES
850 FOR I=S TO N:PRINT #2;DISK$(I*35+1,I
*35+LN(I)):NEXT I
860 LPRINT :CLOSE #2:TRAP 0:GOTO 470
870 IF A<>ASC("W") THEN 970
880 ? "{CLEAR}|WRITE|: Save file to disk
"
890 ? "Use same file name? (Y/N)";
900 GET #1,A:IF A<>ASC("Y") AND A<>ASC("
N") AND A<>27 THEN 900
910 IF A=27 THEN 470
920 IF A=ASC("N") THEN ? "{CLEAR}Enter f
ile name: ";:GOSUB 1340:FILE$="D:":FILE$
(3)=TEMP$
930 ? "{CLEAR}SAVING FILE TO ";FILE$
940 TRAP 950:OPEN #2,8,0,FILE$:GOTO 960
950 ? "{CLEAR}{BELL}Disk Error - ";PEEK(
195):CLOSE #2:? :? "Press |RETURN|":GET
#1,A:GOTO 470
960 FOR I=0 TO LINES:PRINT #2;DISK$(I*35
+1,I*35+LN(I)):NEXT I:CLOSE #2:TRAP 0:GO
TO 470
970 IF A<>ASC("Q") THEN 1030
980 ? "{CLEAR}|QUIT|: Are you sure? (Y/N
)";
990 GET #1,A
1000 IF A=ASC("N") OR A=ASC("n") OR A=27
 THEN 470
1010 IF A=ASC("Y") OR A=ASC("y") THEN GR
APHICS 0:END
1020 GOTO 990
1030 IF A=ASC("N") THEN CURR=CURR+19:IF
CURR>=LINES THEN CURR=0
1040 IF A=ASC("N") THEN 380
1050 IF A<>ASC("G") THEN 470
1060 ? "{CLEAR}Go to which line? ";:GOSU
B 1200:IF N<0 OR N>MAX THEN 1060
1070 CURR=N:GOTO 380
1080 REM |Print directory|
1090 ? "{CLEAR}":POSITION 5,0:? "DIRECTO
RY":POSITION 20,0:? "__________________
"
1100 SETCOLOR 0,INT(16*RND(0)),10
1110 TRAP 1120:OPEN #2,6,0,"D:*.*":GOTO
1130
1120 ? "Can't read directory":GOTO 1160
1130 TRAP 1160:INPUT #2;TEMP$
1140 ? "{TAB}  ";TEMP$:IF PEEK(84)>15 TH
EN POSITION 0,1:? "{DEL LINE}":POSITION
2,15:FOR W=1 TO 100:NEXT W
1150 GOTO 1130
1160 ? :? :? "Press |RETURN| to continue
..."
1170 CLOSE #2:GET #1,A:RETURN
1180 POKE 752,0:? "   ";:INPUT TEMP$:POK
E 752,1:IF TEMP$="" THEN TEMP$=" "
1190 RETURN
1200 REM GET A NUMBER
1210 TEMP$="0":FOR I=1 TO 4
1220 ? "|#|{LEFT}";
1230 GET #1,A:? " {LEFT}";:IF A=27 THEN
POP :GOTO 470
1240 IF A=126 AND I>1 THEN ? CHR$(A);:TE
MP$(I-1)=" ":I=I-1:GOTO 1220
1250 IF A=155 THEN POP :GOTO 1280
1260 IF A<48 OR A>57 OR I=4 THEN 1220
1270 TEMP$(I)=CHR$(A):? CHR$(A);:NEXT I
1280 N=0:P=1:FOR I=LEN(TEMP$) TO 1 STEP
-1:N=N+(ASC(TEMP$(I,I))-48)*P:P=P*10:NEX
T I
1290 ? :RETURN
1300 ? "{CLEAR}START OVER: Are you sure?
 (Y/N)";:GET #1,A
1310 IF A=27 OR A=ASC("N") OR A=ASC("n")
 THEN 470
1320 IF A=ASC("Y") OR A=ASC("y") THEN RU
N
1330 GOTO 1300
1340 REM GET FILE NAME
1350 TEMP$=" ":FOR I=1 TO 12
1360 FOR I=1 TO 12
1370 ? "_{LEFT}";:POKE 702,64
1380 IF PEEK(764)=255 THEN SETCOLOR 0,IN
T(16*RND(0)),10+2*INT(3*RND(0)):GOTO 138
0
1390 GET #1,A:? " {LEFT}";:IF A=155 THEN
 POP :GOTO 1430
```

```
1400 IF A=126 AND I>1 THEN TEMP$(I-1)="
":I=I-1:? CHR$(A);:GOTO 1370
1410 IF (A<48 OR (A>57 AND A<65) OR A>90
) AND A<>46 THEN 1370
1420 TEMP$(I)=CHR$(A):? CHR$(A);:NEXT I:
I=I-1
1430 POKE 764,255
1440 RETURN
1450 REM OTHER COMMANDS
1460 REM ADD YOUR COMMANDS HERE
1470 ? "{CLEAR}|L|ength  |S|earch  |N|ex
t Search"
1480 ? "Any other key--Other commands"
1490 ? "Which?";
1500 P=PEEK(702):POKE 702,64:GET #1,A:PO
KE 702,P
1510 IF A<>ASC("L") THEN 1540
1520 ? "{CLEAR}Length of File: ";LINES+1
;" lines."
1530 ? :? "Press |RETURN|":GET #1,A:GOTO
 470
1540 IF A<>ASC("S") THEN 1770
1550 ? "{CLEAR}|SEARCH|: Enter character
s to search for:"
1560 GOSUB 1180:IF TEMP$=" " THEN 470
1570 KEY$=TEMP$:NX=0
1580 KL=LEN(KEY$)
1590 ? "{CLEAR}By |C|haracter or |L|ine?
";:GET #1,A:IF A=27 THEN 470
1600 TYPE=-1
1610 IF A=ASC("C") OR A=ASC("c") THEN TY
PE=1
1620 IF A=ASC("L") OR A=ASC("l") THEN TY
PE=0
1630 IF TYPE=-1 THEN 1590
1640 ? "{CLEAR}Searching..."
1650 FOR I=NX TO LINES
1660 TEMP$=DISK$(I*35+1,I*35+LN(I)):TEMP
$(LEN(TEMP$)+1)="
                  ":REM 35 SPACES
1670 IF TYPE THEN 1700
1680 IF TEMP$(1,KL)=KEY$ THEN 1750
1690 GOTO 1730
1700 FOR J=1 TO 35-KL
1710 IF TEMP$(J,J+KL-1)=KEY$ THEN 1750
1720 NEXT J
1730 NEXT I
1740 ? "{CLEAR}Not found.":? "{DOWN}Pres
s |RETURN|":GET #1,A:GOTO 470
1750 CURR=I:I=LINES+1:NEXT I
1760 GOTO 380
1770 IF A<>ASC("N") THEN 470
1780 IF LEN(KEY$)<1 THEN 470
1790 NX=CURR+(CURR<LINES)
1800 ? "{CLEAR}|SEARCH| for: "
1810 ? KEY$:GOTO 1580
```
©

This extraordinarily useful Atari Disk technique lets you eliminate DUP.SYS and MEM.SAV from most of your disks and to call a function menu directly from BASIC.

MicroDOS

Dennis Keathley
Webster, TX

It's late at night and you've been working on major revisions to a huge BASIC program for four hours. You finally finish and want to save it back on disk as ULTIMATE.003 (versions 001 and 002 are already on the disk). Upon SAVEing it, you get ERROR 162 (disk full). You think to yourself, "I'll save it as version 001 since that version is out of date anyway." You get another error, ERROR 167 (file locked). "Well I'll just type DOS and unlock the file. After all, MEM.SAV will preserve my program." So you type "DOS" and sit back to twiddle your thumbs for a minute or so. Wait! There's something wrong. There was no MEM.SAV file on the disk. You deleted it because you needed more free sectors. Four hours of work are lost forever.

Save Yourself 88 Sectors And 75 Seconds

Would you like to eliminate the need for DUP.SYS and MEM.SAV on most of your disks?

The Disk Operating System, version 2.0, is a powerful and well-designed piece of software. It also allows you more free RAM for programming use than DOS 1 by moving many of the utility functions into a separate file called DUP.SYS. The Disk Utility Programs are loaded into memory only when requested by typing DOS. The trade-off is that the file DUP.SYS (42 sectors) must be on your disk in drive #1, and it requires 15 to 20 seconds to load into RAM.

Another trade-off is that the BASIC program in RAM will be lost unless you have a MEM.SAV file (45 sectors) on your disk in drive #1, and using MEM.SAV requires almost a minute longer. Eliminating the need for DUP.SYS and MEM.SAV saves you 88 sectors and 75 seconds.

When DOS is booted into RAM at power-on, the portion that remains memory-resident at all times is called the File Management System. The FMS allows you to perform some file manipulation and directory modifications. The most common way to do this is to type DOS which gives you the DOS 2.0 Menu contained in DUP.SYS. Some schemes have been developed to access the RMS functions from BASIC, but they require you to either remember the exact XIO command or to have numerous short programs on your disk that remember them for you (more time and sectors wasted).

Wouldn't it be nice to be able to call a function menu from BASIC without using additional memory, losing your BASIC program, or performing disk I/O?

MicroDOS is a machine language program that is loaded into page six of RAM (Decimal 1536 to 1791), an area conveniently left to the user by the wizards that designed the Atari Operating System. When executed, MicroDOS gives you a menu of FMS functions, which allows you to select the function and desired filename. You may perform as many functions as you wish, and then return to BASIC. And your BASIC program is still there, intact.

Program 1 is the assembly listing. For those of you who do not have an assembler, Program 2 provides a BASIC program to accomplish the same result. If using an assembler, you may designate "D:AUTORUN.SYS" as the binary output file. This will perform automatic loading of MicroDOS when DOS is booted into RAM. If you are presently using an AUTORUN.SYS file, you may append it onto this one using the /A (append) feature of the copy (C) option of the DOS 2.0 menu. The comments in the listings make them self-explanatory.

For ease of use, we "steal" the DOS jump vector, DOSVEC ($0A or decimal 10) and point it to MicroDOS. Perform POKE 10,0 and POKE 11,6. When you type "DOS", BASIC jumps to the address specified in DOSVEC and DOSVEC+1 (low byte, high byte). Now it will jump to MicroDOS instead. One more detail: when SYSTEM RESET is pressed, DOS restores DOSVEC to the original value. To prevent this, POKE 5446,0 and POKE 5450,6. DOS will now reload DOSVEC with $0600 (the start of MicroDOS) when SYSTEM RESET is pressed.

Note that the last six lines of the assembler listing before the .END statement will cause this to be done automatically when the file is loaded. In the BASIC listing, the last 16 numbers perform this. Omit them if you desire and subtract 16 from the 342 in line two. If you should desire to reset DOSVEC so that DUP.SYS may be loaded, POKE 10,159 and POKE 11,23. Now, typing "DOS" will load DUP.SYS, but SYSTEM RESET will cause DOSVEC to point to MicroDOS.

Upon typing "DOS", MicroDOS will display the menu:

LOCK
UNLOCK
DELETE
RENAME

SWEET SIXTEEN FOR 16K ATARI COMPUTERS

FLIP'N'FILE

FLIP'N'FILE is a new functional and attractive storage system for ATARI 400/800 Computer cartridges. Each desktop unit holds 10 cartridges for easy access.
Cat No. 3790 $29.95

DYSAN
5¼" DISKETTES

Dysan diskettes, long known for their exceptional quality, are ideally suited for use on your Atari computer (or any 5¼" soft sector computer). And with HW's low, low price, it's time to stock up. Dysan #104/1
Cat No.3966 Box of 10 $39.95

SYNCRO
WORDMANIA

Learning is not supposed to be this much fun. In WORDMANIA, one or two players compete against a rapidly burning dynamite fuse in order to unscramble anagrams. If you take too long... KABOOM! Random selection from over 1800 words and auto-adjustment of difficulty levels make WORDMANIA a game you won't soon tire of. Requires BASIC cartridge. CAUTION: This game is extremely addicting!
Cat No.3907 cassette $14.95
Disk version (3908) available requires 24K 19.95

SYNCRO
ALIEN HELL

Lucky you... you're enslaved by Aliens to work in their crystal mines. Escape is your only hope. You must search the vast underground corridors in order to find the Magic Crystals and escape. Beware! Strange creatures and deadly fireballs could be around the next corner. Alien Hell is a real-time adventure played in high resolution graphics. Requires BASIC cartridge.
Cat No.3780 cassette, jystk. $14.95
Disk version (3781) available - requires 24K 19.95

DATASOFT
CANYON CLIMBER

You're at the bottom of the Grand Canyon and trying to scale the toughest canyon wall. You must face three difficult challenges before you reach the top. Angry sheep, threatening Indians, and attacking birds all stand in your way.
Cat No.3957 cassette, jystk. $29.95
Cat No.3958 diskette, jystk. 29.95

DATASOFT
SHOOTING GALLERY

Some of the most superb graphics ever created on the Atari Computer. With a limited amount of bullets, you must shoot the elephants, ducks, rabbits, happy & sad faces, and finally the bears. If you can make it to the tenth round, a surprise awaits you.
Cat No.3959 cassette, jystk. $29.95
Cat No.3960 diskette, jystk. 29.95

DATASOFT
PACIFIC COAST HWY

PCH is a fast action game in which you must safely cross PCH without getting squashed by the rush hour traffic. Once you survive the highway, you must leap from boat to raft in order to cross the mighty Pacific. One or two may play. Features excellent graphics.
Cat No. 3961 cassette, jystk. $29.95
Cat No. 3962 diskett, jystk. 29.95

APOGEE SOFTWARE
ASTROWARRIORS

A fast action player vs. player game of space combat. ASTROWARRIORS features realistic gravity and orbital mechanics as well as extensive use of Atari graphics and sound. Attack with your Photon missiles. Protect yourself with shields. Maneuver with your thrusters or enter hyperspace to avoid damage or destruction. Up to four players may battle in one of four skill levels and battle modes.
Cat No.3886 cassette, jystk. $29.95
Cat No.3885 diskette, jystk. 29.95

GAMMA SOFTWARE
HOCKEY

HOCKEY is a hi-res, fast action game for two to four players. An offensive player can carry the puck, pass and shoot. A defensive player can steal the puck and intercept passes. An advanced feature of HOCKEY is the inclusion of "Smart" players who perform automatically. In case of a tie after regulation time, there is even sudden-death overtime.
Cat No.3588 cassette, jystk. $29.95
Cat No.3587 diskette, jystk. 29.95

SYNAPSE
CHICKEN

There's trouble in the barnyard! Ma Hen and a pesky old fox battle it out, as you try to save the local chicken population. Seems that the fox has found Ma Hen's eggs, and she is trying to save them from his deadly clutches. The action gets faster and faster as eggs turn into chicks, feathers fly, chickens squawk, and all bedlam breaks loose!
Cat No.3906 cassette, jystk. $29.95
Cat No.3905 diskette, jystk. 29.95

ANALOG SOFTWARE
RACE IN SPACE

RACE IN SPACE is a very different space game: you must not only avoid the myriad of tiny asteroids about you, but you must contend with comets zooming past, negative universes, and missiles shot by your opponent's ship. Written in machine language, RACE IN SPACE features 128 options, multi-color player/missile graphics & "outer space" sound effects.
Cat No.3934 cassette, jystk(2) $21.95
Cat No.3935 diskette, jystk(2) 24.95

COMPUTER CORE SOFTWARE
YAHTMAN

YAHTMAN is the computerized version of a dice game (of a similar name) which has popular for years. One to four players, school age and up, may complete for the highest score. Invite your friends over and see who can beat YAHTMAN. Requires BASIC cart.
Cat No.3405 cassette, jystk. $19.95

HOW TO ORDER

WRITE OR PHONE. Pay by check, M/C, Visa, or C.O.D. (add $1.50 for COD.)

SEND ORDERS TO:
HW ELECTRONICS, INC.
19511 BUSINESS CTR. DR.
Dept G7
NORTHRIDGE, CA. 91324

TOLL-FREE USA (800) 423-5387
INSIDE CALIF (213) 886-9200

HW ELECTRONICS

WHEN IN SOUTHERN CALIFORNIA, VISIT OUR RETAIL STORES

19511 Business Center Dr.
Northridge, Ca. 91324

2301 Artesia Blvd.
Redondo Beach, Ca. 90277

444 S. Indian Ave.
Palm Springs, Ca. 92262

Offer expires July 31, 1982

Mention this ad and we pay shipping (UPS ground only)

FORMAT
MENU
ADOS
BASIC
Note: Underlined letters are inverse video.

Enter the command you desire (only the first letter is necessary). You will then see:

FN?

Enter your filename and press RETURN. You must enter the "D:" or "D2:" at the beginning of your filename so that FMS will know which drive to access (do *not* use quotes, as in BASIC). The function will be performed and the menu will reappear. Note: if you selected RENAME, the correct filename input is "D1:fn1,fn2". Only one "Dn:" is needed and the comma is required.

If you selected FORMAT, instead of FN? you will see:

FD#

to warn you that the information on the disk will be destroyed. Type "D1",RETURN (or D2,D3, etc.) to format, or anything else to abort formatting.

If you selected MENU, you will see:

D#?

as a prompt. Enter the drive number (single digit) from which you wish to see a disk directory, and it will scroll across the screen. Use CTRL 1 to momentarily halt the scrolling, and CTRL 1 to resume. The directory will stay on the screen until you press RETURN once more.

Making Use Of The 6502 Stack

Selecting ADOS will allow you to load DUP.SYS in the normal way if you have a need to do that (and you shouldn't often!). As a safety feature, to select this function you must type the first *two* letters. If you enter an invalid filename or if FMS is unable to perform the function, MicroDOS will just return to the ">" prompt. To return to BASIC, type "B" (or any other letter that is not a selection letter).

The most difficult part about writing MicroDOS was fitting it into just one *page* (256 bytes) of RAM. Due to this constraint, a Disk Directory option could not be included in page six. In order to add this feature, the MicroDOS selection menu and input buffer had to be moved to the beginning of page one. What? You thought page one of RAM was used by the 6502 microprocessor as a stack area? Well, it is, but the 6502 will never need more than the top one-third. So we will use the bottom one-third.

If you spend a portion of your time programming in BASIC, you will probably find MicroDOS well worth the time spent typing it in. It has saved me much time and frustration.

Program 1. Source Listing

```
0100  .TITLE "MICRO-DOS"
0110 REM COPYRIGHT 1982
0120 REM ALL COMMERCIAL RIGHTS
0130 REM RESERVED BY DENNIS KEATHLEY
0140  .OPT NOEJECT
0150 ;
0160 ;       DEFINE EQUATES
0170 ;
0180  *=$0342
0190 ICCMD *=*+2
0200 ICCMD1=ICCMD+16
0210 ICBAL *=*+1
0220 ICBAL1=ICBAL+16
0230 ICBAH *=*+3
0240 ICBAH1=ICBAH+16
0250 ICBLL *=*+1
0260 ICBLL1=ICBLL+16
0270 ICBLH *=*+1
0280 ICBLH1=ICBLH+16
0290 ICAX1 *=*+1
0300 ICAX11=ICAX1+16
0310 ICAX2 *=*+1
0320 CIOV=$E456
0330 OPEN=3
0340 CLOSE=12
0350 GETC=7
0360 PUTC=11
0370 GETR=5
0380 PUTR=9
0390 EOL=155
0400 CLR=125
0410 TAB=127
0420 ;  ******** MICRODOS SOURCE CODE ** ******
0430  *=$0600
0440 ;
0450 ;       DISPLAY MENU ON SCREEN
0460 ;
0470 INIT
0480  LDY #MBUF&255
0490  LDX #EOM-MBUF
0500  LDA #PUTC
0510  JSR IOE2
0520 ;
0530 ;       ACCEPT COMMAND
0540 ;
0550 GRAB
0560  JSR INPUT
0570 ;
0580 ;       EXECUTE COMMAND
0590 ;
0600  JSR IOP
0610  BPL INIT
0620  BMI GRAB If CIOV error, Y>127
```

```
0630 ;
0640 ;        INPUT CONTROL ROUTINE
0650 ;
0660 INPUT
0670 ;        PUT PROMPT ON SCREEN
0680  LDY #QBUF&255
0690  LDX #2
0700  JSR Q
0710 ;        GET COMMAND CHOICE
0720  JSR IN
0730  LDY #6
0740  STY ICAX11 To "open directory" lat
er
0750 LOOP
0760  DEY
0770  BEQ EXIT
0780  LDA ASCII-1,Y
0790  CMP KBUF
0800  BNE LOOP
0810  LDA CMD-1,Y
0820  PHA       Save command #
0830  CMP #$FE
0840  BNE CONT
0850 ;        VERIFY FORMAT COMMAND
0860  LDY #QBUF+5&255
0870  LDX #3
0880  JSR Q
0890  JSR IN
0900  LDA KBUF
0910  CMP #68  Is first letter a "D"?
0920  BNE EXIT No, then leave MICRO-DOS
0930  BEQ RT
0940 CONT
0950 ;        PUT PROMPT ON SCREEN
0960  LDY #QBUF+2&255
0970  LDX #3
0980  JSR Q
0990 ;        GET FILENAME
1000  JSR IN
1010 RT
1020  PLA   Retrieve FMS command
1030  RTS
1040 ;
1050 ;        PROMPT ON SCREEN ROUTINE
1060 ;
1070 Q
1080  LDA #PUTC
1090  JSR IOE
1100  RTS
1110 ;
1120 ;        EDITOR LINE IN ROUTINE
1130 ;
1140 IN
1150  LDY #KBUF&255
1160  LDX #40
1170  LDA #GETR
1180  JSR IOE2
1190  RTS
1200 ;
1210 ;        EXIT AND I/O ROUTINES
1220 ;
1230 EXIT
1240  LDA KBUF
1250  CMP #77 Directory Request?
1260  BNE DOS No, go to DOS
1270  LDY #QBUF+6&255
1280  LDX #3
1290  JSR Q Put D#? prompt on screen
1300  LDY #SBUF+1&255
1310  LDX #1
1320  LDA #GETR
1330  JSR IOE2 Get drive #
1340  LDY #58
1350  STY SBUF+2 Restore colon
1360  LDY #SBUF&255
1370  LDA #OPEN
1380  JSR IOP2 Open directory
1390 AGAIN
1400  LDA #19
1410  STA ICBLL1
1420  LDA #GETR
1430  JSR IOP Get directory entry
1440  BMI THRU Last entry? Go to THRU
1450  LDY #KBUF-1&255 Tab is first chara
cter
1460  LDX #20
1470  LDA #PUTR
1480  JSR IOE2 Put directory entry on sc
reen
1490  BPL AGAIN Go get another entry
1500 THRU
1510  LDA #CLOSE
1520  JSR IOP Close directory
1530  JSR IN  Wait for RETURN pressed
1540  JMP INIT Go to selection menu
1550 ;
1560 DOS
1570  CMP #65 Is DOS desired?
1580  BNE BASIC No, exit MICRODOS
1590  LDA #68
1600  CMP KBUF+1 Additional check for sa
fety
1610  BNE BASIC
1620  JMP 6047 Load DUP.SYS
1630 ;
1640 BASIC
1650  JMP $A04D Warm start entry
1660 ;
1670 IOE
1680  PHA
1690  LDA #QBUF/256
1700  STA ICBAH
```

```
1710  PLA
1720 ENT
1730  STY ICBAL
1740  STX ICBLL
1750  LDX #0
1760  STX ICBLH
1770  STX ICBLH1
1780  BEQ IO
1790 IOE2
1800  PHA
1810  LDA #KBUF/256
1820  STA ICBAH
1830  STA ICBAH1
1840  PLA
1850  BNE ENT
1860 ;
1870 IOP
1880  LDY #KBUF&255
1890 IOP2
1900  STY ICBAL1
1910  LDX #$10
1920 IO
1930  STA ICCMD,X
1940  JSR CIOV
1950  RTS
1960 ;
1970 ;        DEFINE MENU AND BUFFERS
1980 ;
1990 ASCII
2000  .BYTE 76,85,68,82,70
2010 CMD
2020  .BYTE $23,$24,$21,$20,$FE
2030 QBUF
2040  .BYTE 29,">FN?",198,"D#?"
2050 ;        PUT THE REST IN PAGE 1
2060  *=$100
2070 SBUF
2080  .BYTE "D1:*.*",EOL,TAB
2090 KBUF
2100  *=*+29
2110 MBUF
2120  .BYTE CLR,204,"OCK",EOL
2130  .BYTE 213,"NLOCK",EOL
2140  .BYTE 196,"ELETE",EOL
2150  .BYTE 210,"ENAME",EOL
2160  .BYTE 198,"ORMAT",EOL
2170  .BYTE 205,"ENU",EOL
2180  .BYTE 193,196,"OS",EOL
2190  .BYTE 194,"ASIC",EOL
2200 EOM
2210  *=10
2220  .BYTE 0,6
2230  *=5446
2240  .BYTE 0
2250  *=5450
2260  .BYTE 6
2270   END
```

Program 2. BASIC Loader

```
1 OPEN #1,8,0,"D:MICRODOS.OBJ"
2 FOR I=1 TO 342
3 READ A
4 PUT #1,A
5 NEXT I
6 END
10 DATA 255,255,0,6,253,6,160,37,162,50,
169,11,32,209,6,32,19,6,32,221,6,16,239,
48,246,160,245,162,2,32,84,6,32,90
20 DATA 6,160,6,140,90,3,136,240,63,185,
234,6,205,8,1,208,245,185,239,6,72,201,2
54,208,19,160,250,162,3,32,84,6,32
30 DATA 90,6,173,8,1,201,68,208,30,240,1
0,160,247,162,3,32,84,6,32,90,6,104,96,1
69,11,32,186,6,96,160,8,162,40
40 DATA 169,5,32,209,6,96,173,8,1,201,77
,208,62,160,251,162,3,32,84,6,160,1,162,
1,169,5,32,209,6,160,58,140,2
50 DATA 1,160,0,169,3,32,223,6,169,19,14
1,88,3,169,5,32,221,6,48,11,160,7,162,20
,169,9,32,209,6,16,233,169,12
60 DATA 32,221,6,32,90,6,76,0,6,201,65,2
08,10,169,68,205,9,1,208,3,76,159,23,76,
77,160,72,169,6,141,69,3,104
70 DATA 140,68,3,142,72,3,162,0,142,73,3
,142,89,3,240,19,72,169,1,141,69,3,141,8
5,3,104,208,228,160,8,140,84,3
80 DATA 162,16,157,66,3,32,86,228,96,76,
85,68,82,70,35,36,33,32,254,29,62,70,78,
63,198,68,35,63,0,1,7,1,68
90 DATA 49,58,42,46,42,155,127,37,1,86,1
,125,204,79,67,75,155,213,78,76,79,67,75
,155,196,69,76,69,84,69,155,210,69
100 DATA 78,65,77,69,155,198,79,82,77,65
,84,155,205,69,78,85,155,193,196,79,83,1
55,194,65,83,73,67,155,10,0,11,0,0
110 DATA 6,70,21,70,21,0,74,21,74,21,6
```
©

The past, present, and predicted future of the VIC. Jim Butterfield, an internationally recognized expert on Commodore Computers, discusses VIC software, add-ons, and VIC's place in the world of computing.

Whither VIC?

Jim Butterfield
Associate Editor

VIC has been around for about a year now. It is catching on? Old hat? Will it survive in the turbulent marketplace?

A Little History

The first VIC units had problems. Some were physical in nature. The unit ran too hot – you could hardly touch the ROM plug-in area. The picture was of poor quality. Radiation emission standards were not met. These were cleaned up in a redesign.

VIC had another problem, too. People didn't perceive it as a computer. There were few programs available, and the only peripheral was cassette tape. Memory was limited, and memory expansion was a distant promise.

And to top it all off, VIC was an unknown quantity. The VIC-20 User Guide – the "friendly computer guide" was brightly written and by far the best user documentation that Commodore had yet put out, but it was limited in scope. A Programmer's Reference Guide was hinted at, but seemed slow in making an appearance.

By late summer of 1981, the hardware problems had been resolved, but the VIC had still not established itself as a mature product. Disk, printer, and modem were still promises. Software was still limited.

VICs sold in quite respectable numbers during the Christmas season. A number of games had materialized, allowing the machine to be nicely demonstrated. The economical price and educational potential seemed to appeal to many buyers.

In early spring of 1982, VIC suddenly started to grow up, and a new class of users developed an interest in the machine, not as a game, but as a computer.

New Hardware

The 1540 Single Drive Floppy Disk unit made a dramatic appearance. It looked good and behaved well. It proved to be compatible with the Commodore 4040 units, allowing easy transfer of programs and data files. Speed isn't up to the PET/CBM standards, but for most applications it's quite usable.

Printers are now available in limited quantities. The style is familiar to the Commodore user, but the price is more economical.

Other devices are on the horizon. An IEEE-488 interface, which allows connection to the PET/CBM range of machines, has been demonstrated. An inexpensive modem is almost ready.

Several sizes of memory expansion are on the shelf, from the tiny 3K to the huge 16K. A motherboard system is promised soon which will allow the expansion of the VIC to a full 32K system. On the full system, BASIC will have about 27,500 bytes free; not quite as much as a full PET, but enough for most of us. The remaining memory isn't available to BASIC, but can be used for high-resolution screens and other special applications.

New hardware is appearing from non-Commodore sources. It looks like there will be two types: some will extend the capabilities of the VIC, making new things possible; and others will compete with existing products, hopefully keeping prices interesting.

New Software

Commodore has now made available three major software support modules: a Monitor to allow machine language program writing and debugging; a Super Expander, to make graphics easy; and a Programmer's Aid to help in the writing and debugging of BASIC programs.

These three modules are not essential to the VIC's operation. You can write machine language without the Monitor; you can draw graphics without the Super Expander; and you can certainly write BASIC without the Programmer's Aid. But a job can be made much easier with their help.

The Super Expander, in particular, seems to be very popular among programmers with artistic inclinations. Computer art, geometric figures, and animated sketches have been enthusiastically produced by users.

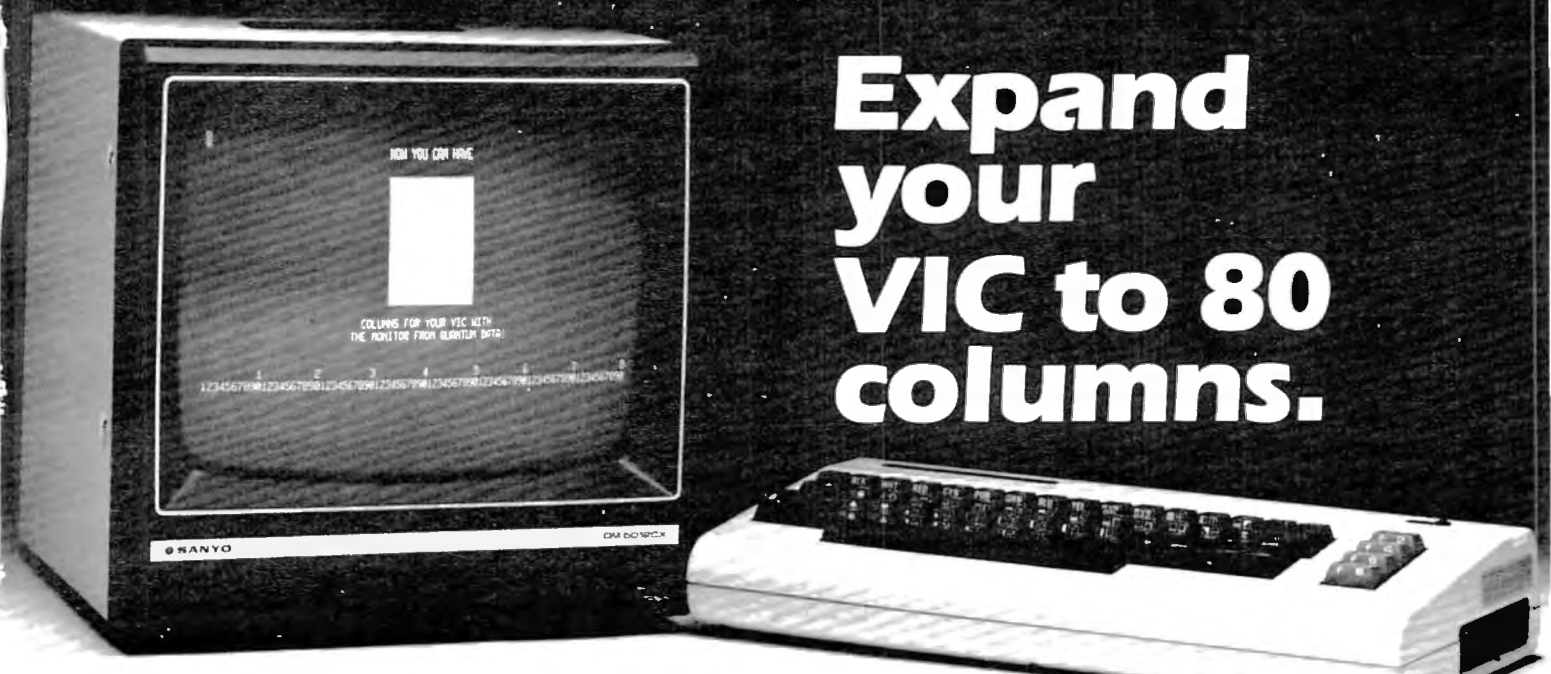
The initial games library is being beefed up with new offerings. I have seen several prototype games which are nearing release and they look good – an order of magnitude better than the satisfactory, existing games on plug-in ROM cartridges.

I'm pleased to see good quality programs coming from non-Commodore sources. There's a lot of good stuff already out (see Table 1) and high-quality programs are a vote of confidence in the VIC by the software houses.

New Information

Commodore has now released the *Programmer's Reference Guide*, which contains a great deal of reference information on the VIC. That's the kind

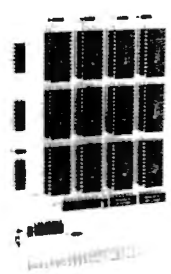
22-40-80 HIKE!



Expand your VIC to 80 columns.

Quantum Data's new Video Combo Cartridge brings you: 40 or 80 column display, plus 16K RAM and PROM socket. $299.00

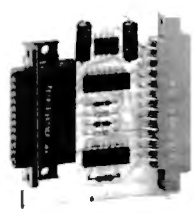
With the Video Combo Cartridge from Quantum Data you can now have 40 or 80 column display, 16K RAM and PROM all in one cartridge. It comes set for 40 column Display compatible with the VIC video modulator and your home T.V. Then, when you are ready to upgrade to 80 columns and a video monitor, just make a simple, no-cost change inside the cartridge. Instructions are provided. Also provided is a socket for a PROM, 16K of memory and AC adaptor. If you don't need memory, then 80 columns can be yours for only $199.50. A listing of the driver software is provided at no charge. A programmed PROM containing this software is also available for $19.95.



QDI expandor:

- Expands Basic user memory up to 24K in 8K steps
- PROMS may be mixed with RAM in 8K blocks
- 8K can be assigned to machine language area
- Plugs directly into VIC expansion port
- Low power, no additional power supply required
- Professional Quality, full buffering on all signals
- Small size: 6 x 4.5 inches.

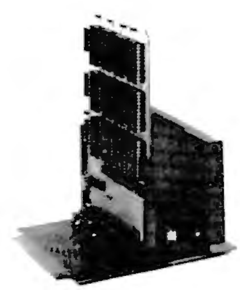
16K expandor........ $149.95
24K expandor....... $199.95



QDI Printor RS-232 interface:

- Provides RS-232 voltage conversion for VIC serial port
- Allows use of a wide variety of RS-232 peripherals including printers, modems and voice synthesizers
- Low power CMOS circuitry requires no external power supply
- Small size: 2½ x 3 inches

Printor.............. $49.95



QDI Minimother:

- Adds 3 slots to the memory expansion port
- Removable card guides allow either boards or cartridges
- Requires no additional power supply
- Fused to protect VIC power supply from overload
- Simple plug-in installation

Minimother.......... $69.95

VISA and MasterCard accepted.

QUANTUM DATA, INC. (714) 966-6553
DEALER HOT LINE (714) 754-1945
3001 Redhill Bldg. 4, Suite 105, Costa Mesa, CA 92626

Non-Commodore Software

This is software that I've seen that looks good. There's undoubtedly lots more. All games listed run in the 5K VIC.

Amok – Nicely animated joystick chase game. Similar to the arcade game Berserk.
United Microware Industries, Inc.
3431 H Pomona Blvd.
Pomona, CA 91768

VIC Pics – Many high resolution pictures derived from TV camera. With explanatory booklet. Nice slide show for your friends.
Un-Word Processor – Simple, but does the job. Needs printer, of course.
Midwest Micro Associates
P.O. Box 6148
Kansas City, MO 64110

Snakman – Fast joystick graphics game. Similar to the arcade game Pacman.
Microdigital
752 John Glenn Blvd.
Webster, NY 14580

The Qube – Draws, scrambles, and solves the Rubic's Cube.
Qumax
GRW Laboratories
P.O. Box 17010
Rochester, NY 14617

of resource we need to exploit the capabilities of the machine.

On the learning level, Commodore has released *An Introduction to BASIC.* This is a beginner-level text which comes as a book and two cassette tapes. At first glance, it looks quite attractive.

What Next?

It looks good. Users are now seeing a computer system rather than a toy. There's a new level of interest in the VIC.

Are there clouds on the horizon? Commodore has announced new products – in particular, the Ultimax and the Commodore 64 – which have new capabilities (and some of the new features won't be VIC-compatible). Will VIC be obsolete before it really gest started?

I think not. In their first incarnations, Ultimax and Commodore 64 won't be direct competitors. Ultimax is too small, and the 64 is twice the price. More to the point: it will be some time before these products mature, just as it took the VIC some time to reach its present state.

I suspect that it will be a couple of years before VIC begins to fade away from the retail market. To a buyer interested now, that makes it a worthwhile investment. Less than $300 for a couple of years' computing? A bargain price. Indeed, in this technology-driven era, it would be optimistic to expect any machine to have an economic life that's much longer.

The box isn't the only part of the question, however. If you move on in two years, how compatible will your software be? The Commodore line has been relatively stable for some time. There are occasional grumblings from long-time users (myself included) when Commodore introduces an improved line which calls for adjustments in some specialized programs, but most programs move fairly easily. Commodore's adoption of the "Kernal" standardized program system is likely to help.

Some things aren't likely to transport. Those plug-in games, for example, are in ROM memory, and can't be trimmed up for a new machine. That's OK ... keep the VIC for the games, or delight the kid who eventually buys your VIC by tossing in the games as part of the package.

At a computer club meeting in 1976, I asked a member what type of system he had. He replied, "I'm waiting for prices to drop to zero." I don't know if he's still waiting. But prices are down to the point where obsolescence in a couple of years shouldn't be an important factor in the decision.

Exciting things are happening with the VIC. There are user groups, there are lots of programs, and new hardware is appearing very quickly. It's going to be fun. ©

*QuadraPET (**COMPUTE!**, June 1981, #13) is a machine language program that lets you "partition" the memory of a 32K PET with Upgrade BASIC into four, 8K areas. A BASIC program can reside in each partition. Each program can be run independently. It's like having four computers in one.*

Super QuadraPET converts QuadraPET to run on 4.0 machines (4032 and 8032) with 32K. Also included is a handy utility program that makes QuadraPET easy to use. It can even relocate itself into one of the partitions – ready to go, but out of the way.

To use QuadraPET 4.0, type in and save Program 1. It places the machine language program in upper memory. Program 1 must be RUN before *you load and RUN Program 2.*

Super QuadraPET

David Sale
Acton, Ontario

Several articles have appeared in **COMPUTE!** dealing with memory partition for the PET computer. I could see the value of this procedure for program development, especially since "Dungeon" could be neatly tucked away in one of the partitions to keep my son happy when I took a break.

Unfortunately, the very useful program by Charles Brannon (**COMPUTE!**, June 1981, #13, p. 102) was written for Upgrade ROMs and will not operate on 4.0 machines. In revising his program, I made several changes:

1) The program has been relocated to the top of memory to leave the cassette buffer available for machine language program development.

2) Calls to BASIC routines are revised for use with 4.0 ROMs.

3) Memory partitions have been changed to make areas more nearly equal in size rather than having the first partition (the most easily accessible) be the smallest.

To create the partitions, load Program 1 and type RUN. Four 8K partitions will be created. By choosing PET #1, you can then load the utility program (Program 2) that will simplify many tasks involved in program development and in "QuadraPETting." I use the utility program to relocate itself into partition four where it is available when needed to perform the following tasks:

1) Converting decimal numbers to hex and vice versa.

2) Moving programs to and from the first partition.

This is necessary to avoid having to use machine language SAVEs or awkward loading procedures.

3) Printing a directory to show which areas are available for use to avoid accidentally destroying a program.

Moving Around In The Partitions

In relocating programs, the program to be moved is duplicated in the desired partition with all pointers reset to enable it to operate correctly. It is *not* erased from the original area. This makes it possible to revise a program in one area and then compare the results with the original in another.

If you are working in the partition that contains the utility program you can move to another area easily using its routine #6. If you are in another area, you can move using SYS 32513. Once you have moved to a new area, the program contained in it will operate normally. To erase the program or enter a new one, simply type NEW.

One word of caution: the relocation routines will move only BASIC programs. Programs containing machine language must be left in partition one or relocated using "Supermon" (**COMPUTE!**, Dec. 1981, #19, p. 134). If you want to pack "Supermon" into the top of partition four follow this procedure:

1) Load and run QuadraPET.
2) Choose PET #1.
3) POKE 53,127 to temporarily remove the partition.
4) Type NEW.
5) Load and run "Supermon."
6) POKE 53,35 to restore the partition.
7) POKE 32627,121 to protect "Supermon."

If you have saved "Supermon" at the top of memory rather than at 0600 you will need to follow a slightly different procedure:

1) Load "Supermon."
2) Relocate it to 7933 hex using its transfer function.
3) Load QuadraPET and run it choosing PET #1.
4) POKE 32627,121 to protect "Supermon."

"Supermon" is now safely tucked away at the top of memory under the QuadraPET program ready for use with SYS 31027.

No doubt you will discover more uses for these programs once you begin to explore their possibilities.

References:

Brannon, Charles, "QuadraPET: Multitasking On Your PET?," **COMPUTE!**, June 1981, #13, p. 102.

Butterfield, Jim, et al, "Supermon: A Primary Tool For Machine Language Programming," **COMPUTE!**, Dec. 1981, #19, p. 134.

Program 1.

```
10 REM
20 REM  QUADRA PET HIGH MEMORY
30 REM        VERSION 4.0
40 REM     COMPUTE!JUNE 81
50 REM  REVISED BY DAVID SALE
60 REM        NOV.17 1981
70 REM
80 POKE 52,255: POKE 53,127
100 FOR I = 32513 TO 32645
110 READ A
120 POKE I,A
130 NEXT
140 SYS 32586
150 PRINT"{CLEAR}{05 RIGHT}QUADRA PET DIVID
    ES THE AVAILABLE"
160 PRINT"{DOWN}MEMORY INTO FOUR 8K AREAS T
    HAT CAN EACH"
170 PRINT"{DOWN}STORE A SEPARATE PROGRAM."
180 PRINT"{02 DOWN}{05 RIGHT}YOU ARE NOW IN
     PET"PEEK(32611)+1". TO MOVE TO"
190 PRINT"{DOWN}ANOTHER AREA TYPE {REV}SYS
    32513{OFF}."
200 PRINT"{02 DOWN}{05 RIGHT}PROGRAMS MAY B
    E LOADED OR SAVED IN"
210 PRINT"{DOWN}THE FIRST AREA IN THE NORMA
    L MANNER."
220 PRINT"{02 DOWN}{05 RIGHT}PROGRAMS IN OT
    HER AREAS MUST EITHER"
230 PRINT"BE SAVED BY MACHINE LANGUAGE OR M
    OVED"
240 PRINT"{DOWN}TO PET #1 USING THE UTILITY
     PROGRAM."
250 END
995 REM  CLEAR END OF STRING
1000 DATA 169,0,141,133,127
1005 REM  STORE PRESENT POINTERS
1010 DATA 174,99,127,165,42,157,104,127,165,
     43,157,108,127
1015 REM  SELECT NEW AREA
1020 DATA 169,116,160,127,32,29,187,32,228,2
     55,41,15,240,249,201,5,176,245
1030 DATA 170,202,142,99,127
1035 REM  SET ALL POINTERS & CLR
1040 DATA 169,1,133,40,189,100,127,133,41,18
     9,104,127,133,42,189,108,127,133
1050 DATA 43,169,0,133,52,189,112,127,133,53
     ,32,238,181,96
1055 REM  INITIALIZE EACH AREA FIRST
1060 DATA 162,0,169,0,24,157,0,4,157,0,35,15
     7,0,66,157,0,97
1070 DATA 232,224,3,144,238,76,1,127
1075 REM  POINTER DATA
1080 DATA 0,4,35,66,97,3,3,3,3,4,35,66,97,35
     ,66,97,127
1085 REM  WHICH PET QUESTION
1090 DATA 147,87,72,73,67,72,32,80,69,84,63,
     32,40,49,45,52,41,0
```

Program 2.

```
100 PRINT"{CLEAR}PET UTILITY PROGRAMS"
110 PRINT"####################"
120 REM  VERSION  JANUARY 10, 1982
130 REM          DAVID SALE
140 REM       ACTON  ONTARIO
150 REM   FOR USE WITH QUADRA PET
160 REM   VERSION 4.0 HIGH MEMORY
170 PRINT"{DOWN}{REV}1{OFF} DECIMAL TO HEX"
180 PRINT"{DOWN}{REV}2{OFF} HEX TO DECIMAL"
190 PRINT"{DOWN}{REV}3{OFF} RELOCATE TO PET
     #1"
200 PRINT"{DOWN}{REV}4{OFF} RELOCATE FROM P
    ET #1"
210 PRINT"{DOWN}{REV}5{OFF} PRINT DIRECTORY
    "
220 PRINT"{DOWN}{REV}6{OFF} MOVE TO ANOTHER
     PET"
230 PRINT"{02 DOWN}TYPE IN THE NUMBER OF YO
    UR CHOICE"
240 GET A:IF A<1 OR A>6 THEN 240
250 ON A GOTO 260,440,590,1040,1490,1670
260 PRINT"{CLEAR}DECIMAL HEX CONVERSION  TY
    PE {REV}M{OFF} FOR MENU"
270 PRINT"##################################
    ######"
280 PRINT"{DOWN}TYPE IN DECIMAL NUMBER (MAX
     65535){DOWN}"
290 INPUT A$:IF A$="M"THEN 100
300 A=VAL(A$)
310 IFA>=16^4THENPRINT"ILLEGAL QUANTITY":GO
    TO280
320 IFA>=16^3THENB(1)=INT(A/(16^3)):A=A-B(1
    )*(16^3):GOTO340
330 B(1)=0
340 IFA>255THENB(2)=INT(A/256):A=A-B(2)*256
    :GOTO360
350 B(2)=0
360 IFA>15THENB(3)=INT(A/16):A=A-B(3)*16:GO
    TO380
370 B(3)=0
380 B(4)=A
390 C$="0123456789ABCDEF"
400 PRINTTAB(8)"{UP}= {REV}";
410 FORX=1TO4:B=B(X)+1:B$(X)=MID$(C$,B,1)
420 PRINTB$(X);:NEXT
430 PRINT"{OFF} HEX":GOTO290
440 PRINT"{CLEAR}HEX TO DECIMAL  TYPE {REV}
    M{OFF} FOR MENU"
450 PRINT"################################"
460 PRINT"{DOWN}TYPE IN HEX NUMBER (MAX FFF
    F){DOWN}"
470 DC=0:INPUT HX$:IF HX$="M"THEN 100
480 IF LEN(HX$)>4 THEN 530
490 IF LEN(HX$)<4 THEN TM$="000"+HX$:HX$=RI
    GHT$(TM$,4)
500 FOR I=1 TO 4:IN=ASC(MID$(HX$,I,1))
510 IF IN>47 AND IN<58 THEN IN=IN-48:GOTO 5
    40
520 IF IN>64 AND IN<71 THEN IN=IN-55:GOTO 5
    40
530 PRINT"ILLEGAL QUANTITY":GOTO 460
540 DC=DC+IN*(16^(4-I))
550 NEXT
560 TM$="     "+STR$(DC):DC$=RIGHT$(TM$,5)
570 PRINT"{UP}"TAB(8)"= {REV}"DC$TAB(16)"{O
    OFF}DECIMAL"
```

```
580 GOTO 470
590 PRINT"{CLEAR}RELOCATION PROGRAM TO PET
    #1"
600 PRINT"###############################"
610 IF PEEK(1026)=0 THEN 670
620 PRINT"{DOWN}PET #1 CONTAINS A PROGRAM"
630 PRINT"DO YOU WANT TO DESTROY IT ?"
640 GET R$:IF R$="" THEN 640
650 IF R$<>"Y" THEN 100
660 PRINT TAB(30)"{UP}{REV}YES{OFF}"
670 PRINT"{DOWN}WHICH PET HAS PROGRAM TO BE
     MOVED ?"
680 GET N$:IF N$="" THEN 680
690 NO=VAL(N$):IF NO<2 OR NO>4 THEN 680
700 PRINT TAB(37)"{UP}{REV}"NO
710 PRINT"{DOWN}ARE YOU SURE ?"
720 GET R$:IF R$="" THEN 720
730 IF R$<>"Y"THEN 100
740 PRINT TAB(17)"{UP}{REV}YES{OFF}"
750 PRINT"{DOWN}TRANSFERING CONTENTS OF PET
     #"NO
760 PRINT"INTO PET #1"
770 REM  SAVE POINTERS IF IN PET NO
780 IF PEEK(32611)<>NO-1 THEN 810
790 POKE 32616+NO,(PEEK(42))
800 POKE 32620+NO,(PEEK(43))
810 REM  CORRECT POINTERS IN PET #1
820 NO=NO-1:V1=PEEK(32616+NO)
830 POKE 32616,V1:  REM  VAR. LOW
840 V2=PEEK(32620+NO)-31*(NO)
850 POKE 32620,V2:  REM  VAR. HIGH
860 MH=PEEK(32624+NO)-31*(NO)
870 POKE 32624,MH:  REM  MEMORY LIMIT
880 SU=NO*7936:     REM  SUBTR. FACTOR
890 SA=SU+1025:     REM  START ADDRESS
900 REM  CORRECT POINTER TO NEXT ADDRESS
910 PA=PEEK(SA)+256*PEEK(SA+1)
920 BY=PEEK(SA):POKE(SA-SU),BY
930 SA=SA+1:BY=PEEK(SA):POKE(SA-SU),(BY-SU/
    256)
940 REM  TRANSFER REST OF LINE
950 SA=SA+1:FOR A=SA TO (PA-1)
960 BY=PEEK(A):POKE(A-SU),BY
970 NEXT
980 REM  CHECK FOR END OF BASIC
990 IF PEEK(PA+1)<>0 THEN SA=PA:GOTO 910
1000 POKE PA-SU,0:POKE PA+1-SU,0
1010 PRINT"{DOWN}PROGRAM TRANSFERRED"
1020 FOR A=1 TO 3000: NEXT
1030 GOTO 100
1040 PRINT"{CLEAR}RELOCATE FROM PET #1"
1050 PRINT"####################"
1060 PRINT"{DOWN}TO WHICH PET ? (2-4)"
1070 GET N$:IF N$="" THEN 1070
1080 NO=VAL(N$):IF NO<2 OR NO>4 THEN 1070
1090 PRINT TAB(22)"{UP}{REV}"NO
1100 PRINT"{DOWN}ARE YOU SURE ?"
1110 GET R$:IF R$="" THEN 1110
1120 IF R$<>"Y" THEN 100
1130 PRINT TAB(16)"{UP}{REV}YES{OFF}"
1140 IF PEEK(1026+(NO-1)*7936)=0 THEN 1200
1150 PRINT"{DOWN}PET"NO"CONTAINS A PROGRAM"
1160 PRINT"DO YOU WANT TO DESTROY IT ?"
1170 GET R$:IF R$=""THEN 1170
1180 IF R$<>"Y"THEN 100
1190 PRINT TAB(30)"{UP}{REV}YES{OFF}"
1200 PRINT"{DOWN}TRANSFERING PROGRAM TO PET
     #"NO
1210 PRINT"FROM PET #1"
1220 REM  SAVE POINTERS IF IN PET 1
1230 IF PEEK(32611)<>0 THEN 1260
1240 POKE 32616,(PEEK(42))
1250 POKE 32620,(PEEK(43))
1260 REM  CORRECT POINTERS IN PET NO
1270 NO=NO-1:V1=PEEK(32616)
1280 POKE 32616+NO,V1:  REM  VAR. LOW
1290 V2=PEEK(32620)+31*(NO)
1300 POKE 32620+NO,V2:  REM  VAR. HIGH
1310 MH=PEEK(32624)+31*(NO)
1320 POKE 32624+NO,MH:  REM  MEM. LIMIT
1330 AD=NO*7936:        REM  ADD. FACTOR
1340 SA=1025:           REM  START ADDRESS
1350 REM  CORRECT POINTER TO NEXT ADDRESS
1360 PA=PEEK(SA)+256*PEEK(SA+1)
1370 BY=PEEK(SA):POKE(SA+AD),BY
1380 SA=SA+1:BY=PEEK(SA):POKE(SA+AD),(BY+AD/
     256)
1390 REM  TRANSFER REST OF LINE
1400 SA=SA+1:FOR A=SA TO (PA-1)
1410 BY=PEEK(A):POKE(A+AD),BY
1420 NEXT
1430 REM  CHECK FOR END OF BASIC
1440 IF PEEK(PA+1)<>0 THEN SA=PA:GOTO 1360
1450 POKE PA+AD,0:POKE PA+1+AD,0
1460 PRINT"{DOWN}PROGRAM TRANSFERRED"
1470 FORA=1TO3000:NEXT
1480 GOTO 100
1490 PRINT"{CLEAR}DIRECTORY FOR PROGRAMS IN
     PARTITIONS"
1500 PRINT"####################################
     ###"
1510 FOR A=1 TO 4
1520 PRINT:PRINT"{DOWN}PET #"A
1530 PRINT"########"
1540 READ B
1550 IF PEEK(B)=0 THEN PRINT "EMPTY";:GOTO 1
     610
1560 PRINT "CONTAINS PROGRAM";
1610 NEXT
1620 DATA 1025,8961,16897,24833
1630 RESTORE
1640 PRINT:PRINT"{02 DOWN}PRESS {REV}M{OFF}
     TO RETURN TO MENU"
1650 GET R$:IF R$=""THEN 1650
1660 GOTO 100
1670 SYS 32513
1680 PRINT"{CLEAR}"TAB(3)"{02 DOWN}YOU ARE N
     OW IN PET #" PEEK(32611)+1 "{LEFT}
     ."
1690 PRINTTAB(3)"{02 DOWN}ANY PROGRAM IN THI
     S PARTITION IS"
1700 PRINT"{DOWN}UNDISTURBED AND MAY BE RUN
     OR MODIFIED."
1710 PRINTTAB(3)"{02 DOWN}TO ENTER A NEW PRO
     GRAM FIRST TYPE {REV}NEW{OFF}"
1720 PRINT"THEN PROCEED IN THE NORMAL FASHIO
     N."
1730 PRINTTAB(3)"{02 DOWN}TO MOVE TO ANOTHER
      PARTITION TYPE"
1740 PRINT"{DOWN}{REV}SYS 32513{OFF}. THE PR
     OGRAM IN THIS AREA"
1750 PRINT"{DOWN}WILL BE PROTECTED UNTIL NEE
     DED LATER."
```

©

A disaster-prevention technique for Apple Disk systems. Teachers should find this especially valuable. For DOS 3.3 with 48K RAM.

Apple DOS Changer

Robert Swirsky
Cedarhurst, NY

If there is an Apple in a classroom, there is usually a student who gets immense pleasure out of typing "INIT" and watching the drive whir and click as it erases the disk!

There is a strange phenomenon which occurs when you mix microcomputers and young people. Most of them will treat the computer with reasonable care, but there is always someone who feels compelled to try to "mess up" the computer's files.

I know from experience; I used to be such a student. What can be done to stop people from tampering with dangerous DOS commands? The answer is simple – just *change* the names of commands that you don't want available to users. Since the DOS is stored in RAM, it is relatively easy to find the table of commands, and modify it accordingly.

This program will change the DOS commands. When run, it displays each command followed by a prompt. If you want to change the command, just type in your revision. To leave it unmodified, hit the RETURN key. The next DOS command will be displayed, and the same procedure is followed.

A few words of caution are needed. First of all, don't make the new commands so that they match the beginnings of other commands. For example, if you change "CATALOG" to "C", conflicts will arise if you try to use the CLR, CLOSE, or other commands that begin with C. Second, don't put spaces in your command. This tends to make strange things happen. Also, avoid making your commands excessively long. There is only so much room in memory for the command table. Note, however, that the commands can be of different lengths from the original ones.

How To HAHA A Disk

The program will work on DOS 3.3 with 48K of RAM. It will not work with less than 48K, without changing the addresses that the changes are POKEd to. Since the vast majority of Apple owners have the 48K, this is hardly a limitation.

After the program is RUN, the changes can be saved as a permanent part of DOS. Insert a *blank* disk, and type the INIT command, followed by the correct parameters. Of course, if you have changed the INIT *itself* to something else, you would type that in instead. When the initialization is completed, the disk will have your DOS modifications stored on it. Whenever DOS is booted from that disk, your custom commands will be the ones that the Apple knows.

A great deal of protection is offered with this program. If you decide to change INIT to HAHA, nobody can init the disk. Anyone who tries will be greeted with a ?SYNTAX ERROR (beep). However, when you go to HAHA your disk, you will be successful.

```
0 ONERR  GOTO 1000
1 DIM CO$(28)
2 P = 1
10 FOR X = 43140 TO 43271
20 CO$(P) = CO$(P) + CHR$ ( PEEK (X))
31 IF  PEEK (X) > 128 THEN P = P + 1
32 IF P = 28 THEN 70
40 NEXT
70 HOME
80 PRINT "DOS CHANGER BY ROBERT A. SWIRSKY
   "
160 VTAB 24: PRINT "HIT 'C' TO CONTINUE";:
    GET R$: IF R$ < > "C" THEN 160
170 HOME
180 PRINT : PRINT "YOU WILL SEE ALL 28 DOS
    COMMANDS"
190 PRINT "IF YOU WISH TO LEAVE IT UNCHANGE
    D,        PRESS RETURN"
200 PRINT "TO CHANGE THE COMMAND, SIMPLY RE
    TYPE IT WITH YOUR CHOICE."
210 PRINT : PRINT
220 LL = 43139
221 HL = 43271
230 FOR Q = 1 TO 28
235 PRINT CO$(Q);
240 PRINT  TAB( 20);" --> ";
250 INPUT R$
260 IF R$ = "" THEN R$ = CO$(Q)
261 IF  LEN (R$) = 1 THEN  PRINT "PLEASE --
     TWO OR MORE LETTERS": GOTO 240
270 FOR W = 1 TO LEN (R$) -1
280 POKE LL + W, ASC ( MID$ (R$,W,1))
290 NEXT
295 IFASC(RIGHT$(R$,1))>127THENPOKELL+W,ASC
    (RIGHT$(R$,1)):GOTO 501
300 POKE LL + W,ASC(RIGHT$(R$,1))+128
501 REM
510 LL=LL+W
520 IF LL > HL THEN 600
530 W=0
599 NEXT
600 PRINT "CHARACTER LIMIT EXCEEDED"
1000 PRINT "RUN TERMINATED"
```

A Monthly Column

This month, columnist Michael Day tackles some of the practical aspects of adding a modem to your computer. A modem will let your computer talk to other computers over the telephone lines. If you're shopping for a modem, you'll need to answer some questions first. Will you want the computer to be able to "pick up the phone" by itself and answer incoming calls automatically? To place its own calls *automatically? How can you use a modem with a "Princess" phone? Will you need an extra interface to connect the modem to your computer or will it just plug in?*

Telecommunications:

Choosing A Modem

Michael E. Day
Chief Engineer, Edge Technology

There are a number of different types of modems on the market. In order to understand each modem's capability in each application, we need to break them down into different groups.

One aspect to consider is the way the modem attaches to the phone line. There are three basic methods:

1. Acoustic coupler: a modem that communicates through a telephone by sound.
2. Inductive coupler: a modem that communicates through a telephone by electromagnetic induction.
3. Direct connect: a device which allows a modem to connect directly to the phone line instead of using the telephone to communicate through.

The acoustic coupler and direct connect are the most commonly used methods. The inductive coupler is actually a modification of the acoustic coupler, and generally still requires that the transmitter portion of the modem be acoustically coupled. The usually stated advantage of the inductive coupler is its avoidance of the distortion and interference caused by the extra translation to sound and back occuring in the acoustic coupler.

This is of little help, however, since the majority of problems with acoustic coupling involve the transmitter portion, which needs to be acoustically coupled anyway. The inductive coupler also has its own set of problems which can often make it more difficult to work with if care has not been exercised in its design.

Each telephone has its own quirks with regard to inductive coupling, and this can sometimes make it difficult to get a modem to work properly. Also, because of the sensitivity involved in properly picking up the weak electromagnetic signals, the inductive coupler is more sensitive to stray magnetic fields, fields generated by a computer or terminal can interfere with the operation of the modem.

Portability Of The Acoustic Coupler

The acoustic coupler modem has been around for some time. The advantage of the acoustic coupler is its portability. It can be used with any telephone that has a standard handset (referred to as a *500 style phone*). This means that you are not limited as to the electrical connections to the phone line, only to the physical construction of the phone handset, which must fit into the rubber or plastic "cups" of the acoustic coupler.

This can be a problem if you have a princess or flip phone or some other style which does not use the standard 500 series handset. Other problems with the acoustic coupler are a result of the acoustic coupling itself. Because of the signal's conversion from electrical to acoustic and back to electrical again, some amount of distortion is generated. This can be held to a minimum with proper modem design, but cannot be eliminated entirely. The telephone itself is a major contributor to this problem. A great portion of the distortion can be cured simply by replacing the carbon microphone in the telephone handset with a condensor (capacitor) microphone.

The standard acoustic coupler used to communicate over the telephone is referred to as a 103 compatible modem. This is a full duplex type modem – that is, it can send and receive data at the same time. Because of the method used to encode and decode the data to be transmitted by this type of modem, the maximum communication speed that can be used reliably is 300bps (30 characters per second).

This limitation is the result of many things, including the bandwidth of the phone line, the encoding scheme, and the signal distortion caused by the acoustic coupling. Although it is possible to push the speed a bit higher, reliability will suffer. A very good modem might let you push it to 400 or 450 bps, but, because of acoustic distortion, this is about the maximum.

Another problem with the acoustic coupler is its limited sensitivity. Most are not able to operate at a receive level of less than -30db to -36db. This is equivalent to a whisper. About 90% of the phone calls placed will provide a communications link allowing a signal through stronger than this. Therefore, this is not a great problem. However, the strength of the transmitted signal is of major importance, since the weaker the transmitted signal is, the weaker the received signal at the other end is going to be. Generally, it is desirable to have a transmitted signal strength of between -6db and -10db. This is equivalent to loud talking.

Direct Connect Method

The other method of attaching a modem to the phone line is the direct connect method. Here you attach the modem directly to the phone line, rather than acoustically through a telephone handset. This eliminates the problem of distortion by not going through the translation to sound and back.

There are many types of direct connect modems. The type that most computer users need is the 103 compatible type. It uses the same conversion scheme to communicate as the 103 compatible acoustic coupler, although the translation to sound and back is, of course, not necessary.

The main advantages of the direct connect modem are reduced signal distortion and improved sensitivity, (since the sound translation is eliminated). Some direct connect modems are able to communicate at up to 600bps (60 characters per second). However, some reduction in reliability should be expected at these high speeds. Other modems are designed so that they cannot operate at speeds above 300bps – in order to achieve a higher degree of reliability at the normal speeds.

Another attractive aspect of the direct connect modem is that it allows the computer to access the phone line directly to answer, and even make, calls *by itself*. This cannot easily be done with the acoustic coupler because it goes through a phone which must be physically lifted off a receiver to start dialing. The disadvantage of a direct connect modem, however, is that it must be physically attached to the phone line. This can be a problem with a business phone setup, pay phone, etc. As a result, the direct connect method is usually used in a fixed location where the phone line is easily attached. Often a phone line is solely for the use of a computer.

Three Types Of Direct Connect Modems

The three main types of direct connect modems are the "auto answer," "auto answer/manual originate," and "auto answer/auto originate."

An auto answer modem can answer calls, but is unable to place any. An auto answer/manual originate modem can answer calls as well as allowing you to manually make a call (after which control is transferred to the modem). Finally, an auto answer/auto originate modem lets the computer answer calls automatically as well as place them by itself without operator intervention.

While most acoustic couplers are separate devices that attach to a computer or terminal through some sort of interface (usually an RS232 serial interface), a few are built into the equipment. If the acoustic coupler is to be purchased as a separate device, be sure to find out if the separate interface will be needed and include it in the cost of the modem. Often a computer already has an RS232 port available to which to attach the modem. If this is the case, the additional interface may not be needed.

Direct connect modems come as either a separate unit that attaches to the computer, or as a device that can be mounted inside the computer. If it is a separate external unit, the interface must be taken into account. Generally, modems which are designed to be installed inside the computer provide their own interface to the computer, and the extra cost need not be considered.

Although the final decision must be made according to your own needs, a general rule of thumb is that if you have no need for automatic operations, an acoustic coupler will probably serve best. If, however, you do desire automatic functions, then you will need the direct connect type modem. ©

In this, the first of a two-part series on the inner workings of Atari Graphics, the author reviews the computer's system of screen management and defines several important terms including color clock, playfield, mode line, *and* display list. *Next month, the article concludes with techniques for using* color indirection, *a powerful graphics tool, and explores the new GTIA chip in detail. This new chip costs nothing if your Atari is still under warranty. If you have an older machine, the nearest authorized service center should be stocking it by now and will install it for about $60.*

Part I:

Atari Video Graphics And The New GTIA

Craig Chamberlain
Birmingham, MI

The GTIA is an exciting new graphics chip now being shipped in Atari 400/800 computers. Among its special features are a sixteen color mode with a resolution eight times better than the Apple's, and the capability of generating 256 color variations. The GTIA chip provides three new graphics modes in addition to the normal fourteen, totally different, full-screen modes. This article defines a few terms relating to graphics, explains the normal graphics modes, then introduces the new modes provided by the GTIA.

ANTIC Is A Busy Chip

We all know that the Atari 400 and 800 have superior graphics capabilities. This has been achieved by designing special chips to handle video display tasks, taking that burden off the main microprocessor. In Atari computers these special chips are known as ANTIC and CTIA.

The ANTIC chip is actually an advanced DMA (direct memory access) controller that qualifies as a true microprocessor. It has an instruction set (mode lines and "load memory scan" operation), a program (the good 'ole display list), and data (display memory and character sets).

This special chip is a rather busy fellow. Its responsibilities include doing DMA for the display list, the display data (playfields), the character set, and player/missile buffers. Besides that, it sets the playfield width, controls horizontal and vertical fine scrolling, keeps track of the vertical position of the scan beam, and handles NMI interrupts. It also supports a light pen.

The GTIA: Three New Modes

The other chip is the CTIA, or Computer Television Interface Adapted integrated circuit. This is the chip which handles all color and luminance (brightness) information to send to the television screen. This is a complicated process, but the chip designers at Atari got carried away and created whole new functions which we know as the player/missile graphics system. It is the CTIA which processes the horizontal position, size, priority, and color of the players. The CTIA also watches for player/playfield collisions, joystick triggers, and console keys. Like the ANTIC, it is a busy chip.

The new GTIA chip replaces the CTIA. Rumor has it that the "G" stands for George. Apparently some fellow named George was still not satisfied with all the special functions of the CTIA, and gave it the ability to generate three totally new graphics modes. When you find out what these new modes can do, I think you will appreciate "George" and his GTIA.

The three new modes are 9, 10 and 11. The operating system and, therefore, Atari BASIC, supports these new modes. But before describing all the features of these new modes, I want to define a few essential terms and review the normal graphics modes 0 through 8.

In order to fully understand Atari graphics, one must have a solid concept of how a television display is generated. And no discussion on "television theory" would be complete without a definition of the "color clock." The term *color clock* derives from the fact that there is a problem in measuring distances on a television screen. Different television sets have different screen sizes, with 9", 13" and 19" being common diagonal measurements. All television sets, however, have a scanning beam which translates a signal from the computer into a picture on the screen.

The signal coming from the computer contains two characteristics. It has a frequency, which defines a color, and it has an amplitude, which defines the *luminance* of that color, often referred to as the brightness or intensity. These qualities of the computer signal affect the way in which the scanning beam shoots electrons at the phosphors on a television screen. This electron shooting process is done horizontally, one line at a time, but it is done so quickly that it is not noticeable to the human eye.

When drawing a line, the scanning beam starts at the left edge of the screen and proceeds to the right edge, shooting electrons the whole time. Since the beam has a finite amount of time it can

ATARI SAYS ITS FIRST WORD

WITH A VOICE BOX BY THE ALIEN GROUP!

THE ALIEN GROUP has emerged from the underground, daring to offer a full-featured speech synthesis system that is flexible, low in cost, and needs no accessory devices. No Interface, Cables, Speaker, Amp, or External Power Required! VOICE BOX has been designed and programmed by Atari users to become the integral voice of a *400* or *800* computer. Simply plugged into the serial port, VOICE BOX automatically routes all speech into the speaker of your television monitor. With the menu-driven operating system supplied, you'll be creating original, intelligible speech within moments after loading disk or cassette. No lengthy or obscure instructions to wade through.

The system includes a dictionary which translates typed text into VOICE BOX's phonetic language. The dictionary can be expanded to include as many as *5,000* words of your own custom vocabulary. Unlimited speech can be produced by straightforward phonetic definiton at any time, even if the dictionary should be full.

The speech routines can be called from other programs for any purpose you can conceive. Here are a few suggested uses:

VOICE BOX
Speech Synthesizer

SOUND EFFECTS
- Access 64 phonemes at any of 4 pitch levels to add filtered, contoured sound to the Atari audio repertoire.

GAMES
- Program aliens to hiss threats, moan when destroyed.
- Devise weird, non-human tongues for dungeon dwellers.
- Insert cryptic spoken clues in maze games.

COMPUTER OPERATION
- Code verbal prompts and error messages that command attention and leave the current display intact.

EDUCATION
- Gain an introduction to the principles of phonetics.
- Learn touch typing through spoken feedback from the keyboard.

In addition, the *Random Sentence Generator* included in the operating system, which prints and speaks endlessly startling, amusing, even poetic combinations of words supplied by the user, helps teach school children to identify parts of speech and recognize a variety of sentence structures.

A minimum of 16K RAM is required by the operating system. Either disk or cassette includes both 16K and 32K versions. Try VOICE BOX for up to 10 days, and if it isn't the finest value you've ever seen in a computer peripheral, the most challenging and provocative addition you've ever made to your system, return it in its original condition for a full refund.

When ordering specify disk- or cassette-based operating system, and enclose check or money-order for $169, or state your VISA or MASTERCARD number. Send mail orders to: THE ALIEN GROUP
27 West 23rd Street
Dept. PT-1
New York, New York 10010

or telephone orders from
10 AM to 6 PM New York time
(212) 924-5546

ALSO AVAILABLE AT LEADING COMPUTER STORES THROUGHOUT THE WORLD.

spend drawing one line, the beam will seemingly have to move faster to cover more area on a larger screen. Thus the problem of trying to measure horizontal distances is further complicated by the fact that different scanning beams not only travel different amounts, but also at different rates. Our unit of measurement cannot really be a distance; it must be a unit of time. The hint I gave a moment ago was that the scanning beam has a certain amount of time it can spend on one scan line. How fast or how far the beam travels is insignificant.

The ANTIC knows how to handle fourteen different kinds of mode lines.

Understanding Color Clock

The fact that our unit of measurement is based on time explains the word *clock* in the term *color clock*. A color clock is the amount of time the computer needs in order to sufficiently change the frequency of the signal it generates so as to produce a different color. What a mouthful! This is my own personal definition; it has worked for me, but some people may not agree with it. Here's another definition. A scan line is the horizontal path of the scanning beam from the left edge of the screen to the right edge.

Scan lines extend horizontally across the screen, but it takes a lot of them stacked vertically to fill up the screen from top to bottom. Therefore, horizontal resolution is usually expressed in terms of color clocks while vertical resolution is expressed in scan lines. Of course, on different television sets the actual lengths will differ, but the resolution horizontally to vertically is always proportionate. It turns out that, on any screen, one color clock appears to be equal in length to two scan lines.

Now we have to get even more technical for a moment. The scanning beam starts at the upper left corner of the screen and travels horizontally to the right. By the time it hits the right edge it has drawn one scan line that is 228 color clocks wide. The team then shuts off for a short period while it returns to the left edge, only one scan line lower. This period is called the "horizontal blank" for obvious reasons. The beam then turns on again and starts drawing the next scan line. This sequence of drawing scan lines continues 262 times. At that point, the scanning beam, at the lower right corner of the screen, shuts off and returns to the upper left corner of the screen during a period known as the (guess what!) "vertical blank."

This whole process of drawing 262 scan lines, each of 228 color clocks, plus the blanking periods, constitutes one "frame." The television draws sixty of these frames every second, because your home power line is 60 Hz (cycles). The name given to this display method is "raster scan." The fact that your Atari follows a broadcast standard referred to as "NTSC" makes it one of the few home computers that can be video-taped without special equipment.

Just because the scanning beam generates all those scan lines and color clocks doesn't mean that the computer is generating that much display data. Even if the computer did, you wouldn't see the whole image since most television sets display a little less than 200 scan lines of about 170 color clocks. The part where the true picture exists is called the playfield, and now it's time for another definition.

Playfields And Mode Lines

The playfield is the portion of each scan line for which data read from memory can produce colors and luminances. The background exists at the ends of each scan line; the playfield is in the middle. From the viewpoint of one frame, the playfield appears as a rectangular region which extends to the sides of the screen.

Two things control the size of this playfield area. The height in scan line is controlled by the display list as you will see in a moment. Recall that the width in color clocks is set by the DMA control register of the ANTIC.

SDMCTL	**$022F**	**559**	**shadow**
DMACTL	**$D400**	**54272**	**hardware**

D5	**1**	**display list DMA enable**
	0	**display list DMA disable**
D1,D0	**00**	**playfield DMA disable (no playfield)**
	01	**narrow playfield (128 color clocks)**
	10	**standard playfield (160 color clocks)**
	11	**wide playfield (192 color clocks)**

The OS screen handler always uses a standard width playfield. The advantage of the narrow playfield is that less DMA is required, so programs execute faster. Unfortunately, the screen handler routines do not work properly when the playfield width is other than the standard. The wide playfield generates more data than the television can display; its uses are rather limited. It's even possible to turn off playfield completely, in which case ANTIC fills the screen with scan lines of the background color. As will be shown in a moment, the playfield also requires a "display list" so bit five must be set for any playfield type to be generated.

Remember that a byte is made up of eight

binary "bits." If playfield and display list DMA is enabled, bits may be read from the computer memory during the course of one scan line. The bit pattern determines the frequency and intensity changes of the scanning beam, with the result being different color/luminances. The same bit pattern may be repeated for several scan lines. And the bit pattern can be interpreted in different ways. This leads us to yet another definition.

A mode line is a contiguous group of scan lines for which display memory is read only once.

There are two main types of mode lines. In direct memory map modes, the bit pattern produces the same image on each scan line. Text modes are a more complicated mode type which use a character set.

The ANTIC knows how to handle fourteen different kinds of mode lines. Each mode line corresponds to a different method for interpreting a bit pattern. A full screen graphics mode is actually just a series of identical mode lines.

The display list is merely a sequence of bytes in memory that, among other things, tells ANTIC the proper sequence of mode lines for one screen.

Whenever the screen is opened (accomplished in Atari BASIC with the GRAPHICS statement), the screen handler establishes a display list of many mode lines to produce a screen of the desired mode. Modes can be mixed by manually changing the display list. Display lists produced by the screen handler always contain the proper number of mode lines for exactly 192 scan lines of playfield. Altering the display list can affect the total number of scan lines, which is how the vertical size of the playfield is controlled.

The display list also has other functions, such as control of fine scrolling, horizontal blank interrupts, and loading the memory scan counter of the ANTIC so it knows where to start reading memory.

A mode line divided into several parts forms pixels, which are single plotting points somewhere within the playfield area. A pixel's vertical resolution is the same as the mode line in which it is displayed, so there can be just as many pixels vertically as mode lines in the display list. The number of color clocks over which one pixel is spread is also determined by the mode line. Here is a little chart to show you the pixel size for the primary mapping modes:

MODE	COLOR CLOCKS	SCAN LINES	RESOLUTION (full/split screens)
3	4	8	40 by 24/20
4,5	2	4	80 by 48/40
6,7	1	2	160 by 96/80

Note that each time the width of a pixel is reduced, its height also decreases, so a single pixel appears to be square in shape regardless of the graphics mode.

Some Observations About Memory

Now to talk about memory. In the one-color modes, one pixel is represented in memory by one bit. If the bit is on, playfield zero shows. If the bit is off, the background shows. Modes 4 and 6 are the one-color modes. For more color, modes 3, 5 and 7 allow three colors. The tradeoff is that a single bit is no longer sufficient. Two bits, a pair, are required. The total value of the two bits selects either one of three playfields or the background:

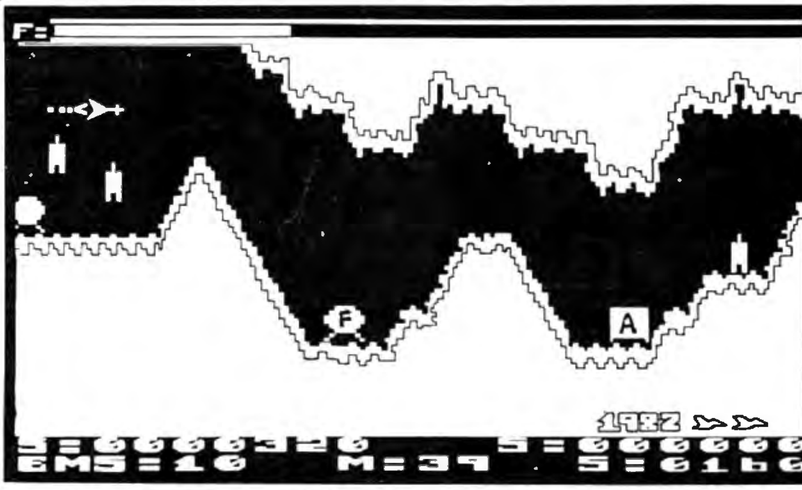
BIT PATTERN	COLOR	PLAYFIELD TYPE
00	0	background
01	1	playfield zero
10	2	playfield one
11	3	playfield two

Playfield zero is the same thing as COLOR 1 in Atari BASIC. Playfield one is really COLOR 2, and so on, with COLOR 0 being the background.

Although modes 4 and 5 both have the same resolution, or pixel size on the screen, mode 5 will require twice as much memory. In the lower resolution modes which require little memory in the first place, the additional memory needed is rather insignificant. You might have noticed that mode 3 had no single color counterpart. Consider that in a 48K system it is possible to have about 150 different mode 3 screens in memory simultaneously. The chip designers probably decided it wasn't worth the effort or memory savings to provide a one color mode with such low resolution.

Therefore, the size of a pixel on the screen is determined by two things: how many scan lines high, and how many color clocks wide. The amount of memory required for a mode is also determined by two things: how many separate pixels to one mode line, and how many color possibilities per pixel. The only real connection between pixel size on the screen and size in memory is that bigger pixels fill up a screen faster, so there are fewer of them, and less memory is needed.

Now, three colors means two bits must be used. Does that mean we are always stuck with only three colors which can't be changed? No. The CTIA is capable of generating 128 color/luminance variations. It can produce sixteen different colors, each in eight different degrees of luminance. But 128 possibilities means seven bits would be required, and, in most cases, seven bits per pixel is simply not feasible. There is a limit to how much memory can be devoted to a screen. The solution to this problem is a sort of compromise, but it also presents some powerful and flexible advantages, too. The solution is to use *color indirection*. ©



All About PET/CBM Character Sets

Louis F. Sander
Pittsburgh, PA

Commodore's unique assortment of graphics characters, combined with their numerous ROM sets and keyboard configurations, make the various PET and CBM character sets maddeningly hard to comprehend. Occasional inaccuracies in published documentation have confused the situation even more. But as of today, the mystery is over – this article describes all the PET/CBM character sets in specific detail, and shows how they relate to each other and to the standard ASCII character set used by many other manufacturers. Such information will be useful to any PET/CBM owner wanting to get past the beginning stages of programming, and will be invaluable to anyone using the IEEE bus or the user port to communicate with a non-Commodore device.

First, some definitions. Many computer devices can display a group of symbols, or *characters,* on paper or on a CRT. The symbols so displayed are called *printing characters,* and they consist of letters, numbers, punctuation marks, special characters, etc.

Within a given piece of hardware, each character is represented by a pattern of bits, which can be stored, manipulated, and transmitted electrically. The binary numbers corresponding to these bit patterns are called *character codes.* In the PET and CBM, all character codes are 8-bit binary numbers, and they are usually referred to by their decimal equivalents.

For example, the code for a PRINTed asterisk (*) is 0010 1010 binary, or 42 decimal. Eight bits allow 256 different codes, which can be represented as decimal numbers in the range 0-255 inclusive. A given code can represent different characters in different machines, or even within one machine, depending on context. In the PET/CBM, for example, different codes are used to put a given character on the screen by PRINTing or by POKEing.

Some character codes do not represent a *printed* character at all. Instead, they instruct the hardware to take a certain non-printing action. These codes are called *control codes* or *control characters.* RETURN, CURSOR DOWN, and RVS are some familiar PET/CBM control actions. If you have ever made your machine do a RETURN by executing the statement PRINT CHR$(13), you have used a control code (the 13) to generate a control action (the RETURN).

A device's *character set* is its complete set of printing and control characters, along with their associated codes. Many computer devices use a standard character set called *ASCII,* pronounced *ask-ee,* which stands for American Standard Code for Information Interchange.

ASCII and the PET/CBM character sets have quite a bit in common, but there are large differences between them which have to be resolved whenever a PET/CBM is to communicate with an ASCII device. The information in this article will allow you to resolve these differences quickly and accurately in your own programs.

The Printed Set

Now let's look in depth at the PET/CBM character sets. To keep things simple, we'll first investigate the *printed character set,* or the complete set of symbols that PET/CBM can display on its screen. The Character Set Demo program will allow us to do just that. Type it in and RUN it right now, being sure to include the semicolon at the end of line 210. If you have an 80-column machine, you need to substitute line 310 for line 200.

If everything has been entered properly, you'll see 256 evenly-spaced characters on the screen. You'll also see the notation "59468 = 12" (or 14), indicating the current contents of memory location 59468. Press any key several times, and observe that the notation alternates between 12 and 14, and, as it does, some of the displayed characters alternate as well. As you press a key, the demo program is changing the contents of 59468, and PET/CBM is changing certain printed characters as that happens. No character codes are being altered at all.

We are demonstrating that every PET/CBM has *two* sets of printing characters. A given character code will produce characters from one set or the other, depending on a number POKEd into 59468. A 12 in that location produces what is often called the "standard" set of printing characters. It is the same in all PET/CBMs, and we will call it Character Set S, for "standard." POKEing 59468 with a 14 produces what is often called the "alternate" character set. This nomenclature is ambiguous, because there are two *different* alternate character sets. Which one you have depends on the ROM set installed in your machine. In this article, we'll call the alternate character set installed in the Original

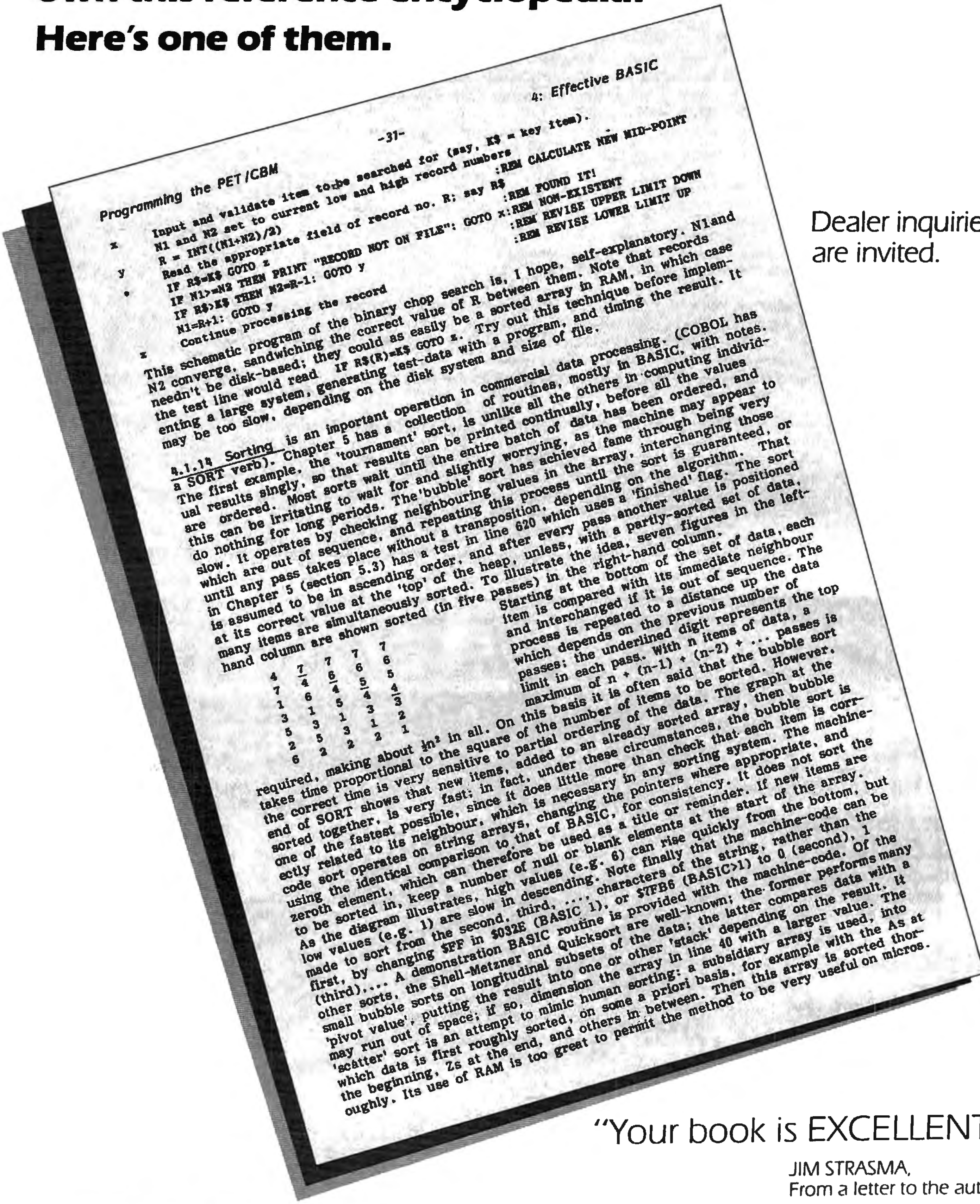
TABLE 1 – SCREEN POKE CHARACTER SETS FOR PET/CBM

POKEs shown are made to locations 32768-33767 for 40-column screens, and 32768-34767 for 80-column screens. Character set selection: S - POKE 59468,12 in all machines. AO - POKE 59468,14 in PETs with Original ROMs. A - POKE 59468,14 in all other machines.

POKE	S	A	AO
0	@	@	@
1	A	a	A
2	B	b	B
3	C	c	C
4	D	d	D
5	E	e	E
6	F	f	F
7	G	g	G
8	H	h	H
9	I	i	I
10	J	j	J
11	K	k	K
12	L	l	L
13	M	m	M
14	N	n	N
15	O	o	O
16	P	p	P
17	Q	q	Q
18	R	r	R
19	S	s	S
20	T	t	T
21	U	u	U
22	V	v	V
23	W	w	W
24	X	x	X
25	Y	y	Y
26	Z	z	Z
27	[	[	[
28	\	\	\
29	]	]	]
30	↑	↑	↑
31	←	←	←
32	Space		
33	!	!	!
34	"	"	"
35	#	#	#
36	$	$	$
37	%	%	%
38	&	&	&
39	'	'	'
40	(	(	(
41	)	)	)
42	*	*	*
43	+	+	+
44	,	,	,
45	-	-	-
46	.	.	.
47	/	/	/
48	0	0	0
49	1	1	1
50	2	2	2
51	3	3	3
52	4	4	4
53	5	5	5
54	6	6	6
55	7	7	7
56	8	8	8
57	9	9	9
58	:	:	:
59	;	;	;
60	<	<	<
61	=	=	=
62	>	>	>
63	?	?	?
64	─	─	─
65	♠	A	a
66	│	B	b
67	─	C	c
68	─	D	d
69	─	E	e
70	─	F	f
71	│	G	g
72	│	H	h
73	╮	I	i
74	╰	J	j
75	╯	K	k
76	└	L	l
77	╲	M	m
78	╱	N	n
79	┌	O	o
80	┐	P	p
81	●	Q	q
82	─	R	r
83	♥	S	s
84	│	T	t
85	╭	U	u
86	╳	V	v
87	○	W	w
88	♣	X	x
89	│	Y	y
90	♦	Z	z
91	┼	┼	┼
92	[illegible]	[illegible]	[illegible]
93	│	│	│
94	π	▒	▒
95	◥	▨	▨
96	Sh Space		
97	▌	▌	▌
98	▄	▄	▄
99	▔	▔	▔
100	▁	▁	▁
101	▏	▏	▏
102	▒	▒	▒
103	▕	▕	▕
104	[illegible]	[illegible]	[illegible]
105	◤	▨	▨
106	▕	▕	▕
107	├	├	├
108	▗	▗	▗
109	└	└	└
110	┐	┐	┐
111	▁	▁	▁
112	┌	┌	┌
113	┴	┴	┴
114	┬	┬	┬
115	┤	┤	┤
116	▎	▎	▎
117	▍	▍	▍
118	[illegible]	[illegible]	[illegible]
119	▔	▔	▔
120	▀	▀	▀
121	▃	▃	▃
122	┘	✓	✓
123	▖	▖	▖
124	▝	▝	▝
125	┘	┘	┘
126	▘	▘	▘
127	▚	▚	▚
128	[illegible]	[illegible]	[illegible]
129	[illegible]	[illegible]	[illegible]
130	[illegible]	[illegible]	[illegible]
131	[illegible]	[illegible]	[illegible]
132	[illegible]	[illegible]	[illegible]
133	[illegible]	[illegible]	[illegible]
134	[illegible]	[illegible]	[illegible]
135	[illegible]	[illegible]	[illegible]
136	[illegible]	[illegible]	[illegible]
137	[illegible]	[illegible]	[illegible]
138	[illegible]	[illegible]	[illegible]
139	[illegible]	[illegible]	[illegible]
140	[illegible]	[illegible]	[illegible]
141	[illegible]	[illegible]	[illegible]
142	[illegible]	[illegible]	[illegible]
143	[illegible]	[illegible]	[illegible]
144	[illegible]	[illegible]	[illegible]
145	[illegible]	[illegible]	[illegible]
146	[illegible]	[illegible]	[illegible]
147	[illegible]	[illegible]	[illegible]
148	[illegible]	[illegible]	[illegible]
149	[illegible]	[illegible]	[illegible]
150	[illegible]	[illegible]	[illegible]
151	[illegible]	[illegible]	[illegible]
152	[illegible]	[illegible]	[illegible]
153	[illegible]	[illegible]	[illegible]
154	[illegible]	[illegible]	[illegible]
155	[illegible]	[illegible]	[illegible]
156	[illegible]	[illegible]	[illegible]
157	[illegible]	[illegible]	[illegible]
158	[illegible]	[illegible]	[illegible]
159	[illegible]	[illegible]	[illegible]
160	█	█	█
161	[illegible]	[illegible]	[illegible]
162	[illegible]	[illegible]	[illegible]
163	[illegible]	[illegible]	[illegible]
164	[illegible]	[illegible]	[illegible]
165	[illegible]	[illegible]	[illegible]
166	[illegible]	[illegible]	[illegible]
167	[illegible]	[illegible]	[illegible]
168	[illegible]	[illegible]	[illegible]
169	[illegible]	[illegible]	[illegible]
170	[illegible]	[illegible]	[illegible]
171	[illegible]	[illegible]	[illegible]
172	[illegible]	[illegible]	[illegible]
173	[illegible]	[illegible]	[illegible]
174	[illegible]	[illegible]	[illegible]
175	[illegible]	[illegible]	[illegible]
176	[illegible]	[illegible]	[illegible]
177	[illegible]	[illegible]	[illegible]
178	[illegible]	[illegible]	[illegible]
179	[illegible]	[illegible]	[illegible]
180	[illegible]	[illegible]	[illegible]
181	[illegible]	[illegible]	[illegible]
182	[illegible]	[illegible]	[illegible]
183	[illegible]	[illegible]	[illegible]
184	[illegible]	[illegible]	[illegible]
185	[illegible]	[illegible]	[illegible]
186	[illegible]	[illegible]	[illegible]
187	[illegible]	[illegible]	[illegible]
188	[illegible]	[illegible]	[illegible]
189	[illegible]	[illegible]	[illegible]
190	[illegible]	[illegible]	[illegible]
191	[illegible]	[illegible]	[illegible]
192	[illegible]	[illegible]	[illegible]
193	[illegible]	[illegible]	[illegible]
194	[illegible]	[illegible]	[illegible]
195	[illegible]	[illegible]	[illegible]
196	[illegible]	[illegible]	[illegible]
197	[illegible]	[illegible]	[illegible]
198	[illegible]	[illegible]	[illegible]
199	[illegible]	[illegible]	[illegible]
200	[illegible]	[illegible]	[illegible]
201	[illegible]	[illegible]	[illegible]
202	[illegible]	[illegible]	[illegible]
203	[illegible]	[illegible]	[illegible]
204	[illegible]	[illegible]	[illegible]
205	[illegible]	[illegible]	[illegible]
206	[illegible]	[illegible]	[illegible]
207	[illegible]	[illegible]	[illegible]
208	[illegible]	[illegible]	[illegible]
209	[illegible]	[illegible]	[illegible]
210	[illegible]	[illegible]	[illegible]
211	[illegible]	[illegible]	[illegible]
212	[illegible]	[illegible]	[illegible]
213	[illegible]	[illegible]	[illegible]
214	[illegible]	[illegible]	[illegible]
215	[illegible]	[illegible]	[illegible]
216	[illegible]	[illegible]	[illegible]
217	[illegible]	[illegible]	[illegible]
218	[illegible]	[illegible]	[illegible]
219	[illegible]	[illegible]	[illegible]
220	[illegible]	[illegible]	[illegible]
221	[illegible]	[illegible]	[illegible]
222	[illegible]	[illegible]	[illegible]
223	[illegible]	[illegible]	[illegible]
224	█	█	█
225	[illegible]	[illegible]	[illegible]
226	[illegible]	[illegible]	[illegible]
227	[illegible]	[illegible]	[illegible]
228	[illegible]	[illegible]	[illegible]
229	[illegible]	[illegible]	[illegible]
230	[illegible]	[illegible]	[illegible]
231	[illegible]	[illegible]	[illegible]
232	[illegible]	[illegible]	[illegible]
233	[illegible]	[illegible]	[illegible]
234	[illegible]	[illegible]	[illegible]
235	[illegible]	[illegible]	[illegible]
236	[illegible]	[illegible]	[illegible]
237	[illegible]	[illegible]	[illegible]
238	[illegible]	[illegible]	[illegible]
239	[illegible]	[illegible]	[illegible]
240	[illegible]	[illegible]	[illegible]
241	[illegible]	[illegible]	[illegible]
242	[illegible]	[illegible]	[illegible]
243	[illegible]	[illegible]	[illegible]
244	[illegible]	[illegible]	[illegible]
245	[illegible]	[illegible]	[illegible]
246	[illegible]	[illegible]	[illegible]
247	[illegible]	[illegible]	[illegible]
248	[illegible]	[illegible]	[illegible]
249	[illegible]	[illegible]	[illegible]
250	[illegible]	[illegible]	[illegible]
251	[illegible]	[illegible]	[illegible]
252	[illegible]	[illegible]	[illegible]
253	[illegible]	[illegible]	[illegible]
254	[illegible]	[illegible]	[illegible]
255	[illegible]	[illegible]	[illegible]

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).
x     N1 and N2 set to current low and high record numbers
      R = INT((N1+N2)/2)                          :REM CALCULATE NEW MID-POINT
      Read the appropriate field of record no. R; say R$
y     IF R$=K$ GOTO z                             :REM FOUND IT!
•     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 implem-
enting 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 individ-
ual 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.
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

				7
		7	7	6
4	7	6	6	5
7	4	4	5	4
1	6	5	4	3
3	1	1	3	2
5	3	3	1	1
2	5	2	2	
6	2			

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
one of the fastest possible, since it does little more than check that each item is corr-
ectly 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 thor-
oughly. 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.

TABLE 2 – CHR$ CHARACTER SETS FOR PET/CBM

The CHR$ function prints all characters shown. Business keyboards do not print codes 161-191. No keyboard prints codes 96-127 or 224-255. *Control Codes in ITALICS apply only to 80-column machines and the newest 40-column machines.* Character set selection is as follows: S - POKE 59468,12 in all machines. AO - POKE 59468,14 in PETs with Original ROMs. A - POKE 59468,14 in all other machines.

CHR$	ACTION	CHR$	S	A	AO	CHR$	S	A	AO	CHR$	S	A	AO
0		32	Space			64	@	@	@	96	Space		
1		33	!	!	!	65	A	a	A	97	!	!	!
2		34	"	"	"	66	B	b	B	98	"	"	"
3	Stop	35	#	#	#	67	C	c	C	99	#	#	#
4		36	$	$	$	68	D	d	D	100	$	$	$
5		37	%	%	%	69	E	e	E	101	%	%	%
6		38	&	&	&	70	F	f	F	102	&	&	&
7	*Bell*	39	'	'	'	71	G	g	G	103	'	'	'
8		40	(	(	(	72	H	h	H	104	(	(	(
9	*Tab*	41	)	)	)	73	I	i	I	105	)	)	)
10		42	*	*	*	74	J	j	J	106	*	*	*
11		43	+	+	+	75	K	k	K	107	+	+	+
12		44	,	,	,	76	L	l	L	108	,	,	,
13	Return	45	-	-	-	77	M	m	M	109	-	-	-
14	*Text*	46	.	.	.	78	N	n	N	110	.	.	.
15	*Set Top*	47	/	/	/	79	O	o	O	111	/	/	/
16		48	0	0	0	80	P	p	P	112	0	0	0
17	Crsr Down	49	1	1	1	81	Q	q	Q	113	1	1	1
18	Reverse	50	2	2	2	82	R	r	R	114	2	2	2
19	Home	51	3	3	3	83	S	s	S	115	3	3	3
20	Delete	52	4	4	4	84	T	t	T	116	4	4	4
21	*Delete Line*	53	5	5	5	85	U	u	U	117	5	5	5
22	*Erase End*	54	6	6	6	86	V	v	V	118	6	6	6
23		55	7	7	7	87	W	w	W	119	7	7	7
24		56	8	8	8	88	X	x	X	120	8	8	8
25	*Scroll Up*	57	9	9	9	89	Y	y	Y	121	9	9	9
26		58	:	:	:	90	Z	z	Z	122	:	:	:
27	*Escape*	59	;	;	;	91	[	[	[	123	;	;	;
28		60	<	<	<	92	\	\	\	124	<	<	<
29	Crsr Right	61	=	=	=	93	]	]	]	125	=	=	=
30		62	>	>	>	94	↑	↑	↑	126	>	>	>
31		63	?	?	?	95	←	←	←	127	?	?	?

CHR$	ACTION	CHR$	S	A	AO	CHR$	S	A	AO	CHR$	S	A	AO
128		160	Sh	Space		192	─	─	─	224	Sh	Space	
129		161	▌	▌	▌	193	♠	A	a	225	▌	▌	▌
130		162	▄	▄	▄	194	│	B	b	226	▄	▄	▄
131	Run	163	▔	▔	▔	195	─	C	c	227	▔	▔	▔
132		164	▁	▁	▁	196	─	D	d	228	▁	▁	▁
133		165	▏	▏	▏	197	─	E	e	229	▏	▏	▏
134		166	▒	▒	▒	198	─	F	f	230	▒	▒	▒
135		167	▕	▕	▕	199	│	G	g	231	▕	▕	▕
136		168	▒	▒	▒	200	│	H	h	232	▒	▒	▒
137	*Toggle Tab*	169	◤	▨	▨	201	╮	I	i	233	◤	▨	▨
138		170	▕	▕	▕	202	╰	J	j	234	▕	▕	▕
139		171	├	├	├	203	╯	K	k	235	├	├	├
140		172	▗	▗	▗	204	└	L	l	236	▗	▗	▗
141	Sh Return	173	└	└	└	205	╲	M	m	237	└	└	└
142	*Graphic*	174	┐	┐	┐	206	╱	N	n	238	┐	┐	┐
143	*Set Bottom*	175	▂	▂	▂	207	┌	O	o	239	▂	▂	▂
144		176	┌	┌	┌	208	┐	P	p	240	┌	┌	┌
145	Crsr Up	177	┴	┴	┴	209	●	Q	q	241	┴	┴	┴
146	Rvs Off	178	┬	┬	┬	210	_	R	r	242	┬	┬	┬
147	Clear Scrn	179	┤	┤	┤	211	♥	S	s	243	┤	┤	┤
148	Insert	180	▎	▎	▎	212	│	T	t	244	▎	▎	▎
149	*Insert Line*	181	▍	▍	▍	213	╭	U	u	245	▍	▍	▍
150	*Erase Begin*	182	▐	▐	▐	214	╳	V	v	246	▐	▐	▐
151		183	▀	▀	▀	215	○	W	w	247	▀	▀	▀
152		184	▀	▀	▀	216	♣	X	x	248	▀	▀	▀
153	*Scroll Down*	185	▂	▂	▂	217	│	Y	y	249	▂	▂	▂
154		186	┘	✓	✓	218	♦	Z	z	250	┘	✓	✓
155		187	▖	▖	▖	219	┼	┼	┼	251	▖	▖	▖
156		188	▝	▝	▝	220	▒	▒	▒	252	▝	▝	▝
157	Crsr Left	189	┘	┘	┘	221	│	│	│	253	┘	┘	┘
158		190	▘	▘	▘	222	π	▒	▒	254	▘	▘	▘
159		191	▚	▚	▚	223	◥	▧	▧	255	π	▒	▒

versacalc T.M.

YES! We said "SORT VISICALC"!

A UNIQUE VISICALC(tm) ENHANCEMENT

NOW AVAILABLE FOR PET

If you use Visicalc(tm) but are bumping into its limitations, you need Versacalc(tm)! Versacalc runs within Visicalc but uses no extra memory; in fact, it effectively increases memory by letting you call in modules from disk as needed.

A Tutorial section makes clear such features as @LOOKUP, DIF, @NA, @ERROR, which are not well explained in the Visicalc manual.

A Utilities section makes it easy to create your own menu-driven modules which condense hundreds of commands into four keystrokes. You can build in sophisticated error checking (e.g. Is the input value between certain limits?). Now it is possible for people untrained in Visicalc to perform the weekly updating without constant instruction.

for Commodore Pet & CBM 2040 - DOS 1.0
available for 8050 on request

Retail $125

SORT Visicalc, and Other Useful Wonders!

NOW YOU CAN:

- SORT a Visicalc screen on any column, ascending or descending, and all related formulas and labels are sorted too.
- put the entire disk CATALOG on the screen at once!
- easily do Year-To-Date accumulations!
- "pound" formulas to expose the full formulas in place on the screen!
- append two Visicalc files!
- print the contents of a /SS file!
- print the contents of a /PF file!
- AND our EASEL BINDER is so nice that you will put your other manual in it!

Distributed by:

Anthro-Digital Software
P.O. Box 1385
Pittsfield, MA 01202
413-448-8278

Versacalc is a trademark of Versacalc Enterprises, Inc
Visicalc is a trademark of Personal Software, Inc

SM-KIT

for Commodore Computers

A Programming Productivity Tool

The SM-KIT is a collection of machine language firmware programming and test aids for BASIC programmers. SM-KIT is a 4K ROM (twice the normal capacity) which you simply insert in a single ROM socket on any BASIC 4 CBM/PET–either 80 column or 40 column. Includes both programming aids and disk handling commands.

ERROR DETECTION: the SM-KIT automatically indicates the erroneous line and statement for any BASIC program error.
LINE NUMBERING: the SM-KIT automatically numbers BASIC statements until you turn the function off.
SCREEN OUTPUT: the commands FIND, DUMP, TRACE and DIRECTORY display on the CRT while you hold the RETURN key (display pauses when the key is released). Continuous output is selected with shift-lock.
OUTPUT CONTROL to DISK or PRINTER: in addition to displaying on the CRT, you can direct output to either disk or printer.
HARDCOPY: allows screen displays to be either printed or stored on disk.
FIND: searches all or any part of a program for text or command strings or variable names. Either exact search or wild card search supported.
RENUMBER: the SM-KIT can renumber all or any part of a program. The selective renumbering allows you to move blocks of code within your program.
VARIABLE DUMP: displays the contents of floating point, integer, and string variables (both simple and array). Can display all variables or any selected variables.
TRACE: SM-KIT can trace program execution either continuously or step by step starting with any line number. Selected program variables can be displayed while tracing.
DISK COMMANDS: as in DOS Support (Universal Wedge), the "shorthand" versions of disk commands may be used for displaying disk directory, initializing, copying, scratching files, load and run, etc.
LOAD: SM-KIT can load all or part of BASIC or machine language programs. It can append to a program in memory, overwrite any part of a program, load starting with any absolute memory location, and load without changing variable pointers.
MERGE: allows merging all or any part of a program on disk with a program in memory.
SAVE and VERIFY: SM-KIT provides one step program save and verification. It also allows you to save any part of a program, or any address range.
DELETE: allows any program segment to be deleted.
REPEAT KEYS: allows repeating functions if not already available on specific PET/CBM model.

ONLY $40

A 4K ROM with both programming and disk handling aids.

Developed by (and available in Europe from) SM Softwareverbund-Microcomputer GmbH, Scherbaumstrasse 29, 8000 Munchen 83, Germany

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.

PET ROMs Character Set A0, for "alternate, original," and the alternate character set in all other ROMs Character Set A. These two alternate sets contain the same characters, but in a different order, as we will see later on.

About 75% of the characters in all three sets are identical. Sets S, A, and AO differ only in the characters produced by the alphabetic keys A through Z, and in four other characters, all graphics. At "power on," graphics keyboard PET/CBM's have Character Set S enabled, while machines with business keyboards have Character Set A.

Character sets can be switched by POKEing 59468 with 12 or 14, or with other numbers as well. Numbers having a binary representation of the form XXXX 110X will produce character set S, while any other number will produce your machine's alternate character set. In machines having the GRAPHIC and TEXT commands, these can also be used to switch character sets.

Now back to the demo program. Without touching the keyboard, study the characters displayed on your screen. Notice that there are 256 characters, all different, and that the first 128 of them are repeated in reverse field to make up the second 128. (There may seem to be two identical SPACE characters, but there aren't – the second one is SHIFTED SPACE, and your computer treats it as a separate character altogether.) This is the complete set of printing characters from the currently activated set. In other words, you are looking at every character your machine can display at this moment.

Now press any key and study the characters in the other set. Again, there are 256 unique symbols, 128 regular and 128 in reverse field. Press a key several times, and notice which characters change as the character sets are toggled. If you count them, you'll find 60 characters that change – 30 regular and 30 reverse field. Note which ones they are, and notice that certain combinations of characters can never be on the screen at the same time, the HEART and the lowercase "s," for example.

You have now seen every character that your machine is able to display. All other PET/CBM's have the same printing characters, but in some machines they are gotten at in a slightly different way. Altogether, there are 316 different characters, 256 of them available at any one time.

Since we've now looked at the complete repertoire of printing characters, let's look further into character codes, the other part of the character set. A character can be displayed on PET/CBM's screen in one of three ways: by POKEing a code into a screen memory location, by pressing a key, or by executing a PRINT statement. Additionally, your machine can send characters to, or receive them from, devices connected to the IEEE, user, recorder, and memory expansion ports. In every case, character codes are used to specify which character is to be displayed, recorded, or transmitted.

The Screen And CHR$ Sets

Our demonstration program POKEd characters to the screen, using the 256 character codes from 0 to 255 inclusive, which produced 256 different printed symbols. POKEing a 1 gave an A, a 2 gave a B, and so on through all the printed characters. This particular combination of codes and characters is valid *only* for screen POKEing, and is summarized in Table 1. We'll call it the *Screen POKE Character Set.*

All other character manipulation in the PET/CBM uses a completely different group of codes to print these same characters, and it is summarized in Table 2. Many of the printing characters and control functions in this set can be activated directly from the keyboard, and all of them can be activated by using the CHR$ function. We will call this the *CHR$ Character Set.*

Some people call it PET ASCII, but that terminology is misleading – PET/CBM's CHR$ character set has twice as many codes as ASCII, and only about half of the 128 ASCII codes have the same meaning in the ASCII and CHR$ character sets!

All PET/CBM keyboard and PRINT operations use the CHR$ character set; it is also used whenever characters are sent to or received from external devices such as printers, files, or modems. If you tell PET to send an asterisk to your printer, it will, in fact, send 0010 1010, or 42 in decimal notation. And whenever PET receives a 42, whatever the 42 may have represented in the sending device's character set, PET interprets it as an asterisk.

There are 256 CHR$ codes, numbered from 0 to 255 inclusive, and the CHR$ character set differs substantially from the POKE set, although both can be used to display the same symbols. Here are the essential differences:

- Very few characters have identical POKE and CHR$ codes.
- There are no CHR$ codes for reverse field characters. Instead, the RVS ON/OFF key or its corresponding CHR$ codes are used to produce them.
- The CHR$ set includes 14 control characters (28 in 80-column machines and newer 40-column machines) in addition to its 128 printing characters.
- Since there are 256 CHR$ codes, and only

PET/CBM OWNERS

TABLE 3 – AMERICAN STANDARD CODE FOR INFORMATION INTERCHANGE (ASCII)

The ASCII Codes in Decimal Form:

0	NUL	32	SPC	64	@	96	`
1	SOH	33	!	65	A	97	a
2	STX	34	"	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EOT	36	$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	'	71	G	103	g
8	BS	40	(	72	H	104	h
9	HT	41	)	73	I	105	i
10	LF	42	*	74	J	106	j
11	VT	43	+	75	K	107	k
12	FF	44	,	76	L	108	l
13	CR	45	-	77	M	109	m
14	SO	46	.	78	N	110	n
15	SI	47	/	79	O	111	o
16	DLE	48	0	80	P	112	p
17	DC1	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	S	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v
23	ETB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	:	90	Z	122	z
27	ESC	59	;	91	[	123	{
28	FS	60	<	92	\	124	¦
29	GS	61	=	93	]	125	}
30	RS	62	>	94	^	126	~
31	US	63	?	95	_	127	DEL

English Names of the Special Characters:

33 – Exclamation point
34 – Quotation mark
35 – Number sign
36 – Dollar sign
37 – Percent
38 – Ampersand
39 – Apostrophe
40 – Opening parenthesis
41 – Closing parenthesis
42 – Asterisk
43 – Plus
44 – Comma
45 – Hyphen (Minus)
46 – Period (Decimal point)
47 – Slant
58 – Colon
59 – Semicolon
60 – Less than
61 – Equals
62 – Greater than
63 – Question mark
64 – Commercial at
91 – Opening bracket
92 – Reverse slant
93 – Closing bracket
94 – Circumflex
95 – Underline
96 – Grave accent
123 – Opening brace
124 – Vertical line
125 – Closing brace
126 – Tilde

Key to Control Code Abbreviations:

ACK – Acknowledgement
BEL – Bell
BS – Backspace
CAN – Cancel
CR – Carriage return
DC1 – Device control #1
DC2 – Device control #2
DC3 – Device control #3
DC4 – Device control #4
DEL – Delete
DLE – Data link escape
EM – End of medium
ENQ – Enquiry
EOT – End of transmission
ESC – Escape
ETB – End of transmission block
ETX – End of text
FF – Form feed
FS – File separator
GS – Group separator
HT – Horizontal tab
LF – Line feed
NAK – Negative acknowledgement
NUL – Null
RS – Record separator
SI – Shift in
SO – Shift out
SOH – Start of heading
SPC – Space
STX – Start of text
SUB – Substitute
SYN – Synchronous idle
US – Unit separator
VT – Vertical tab

128 + 14 = 142 CHR$ characters (156 in some machines), many of the CHR$ codes have no meaning at all in the PET/CBM, and in many cases one printed character has two different CHR$ codes!

Table 3 shows the standard ASCII character set. It is presented in a similar format to Table 2, to facilitate comparison of the ASCII and PET/CBM character sets. Study the two tables carefully, and you'll see that PET/CBM has all but seven of the ASCII printed characters, (94-96 and 128-126), but often with different character codes.

You'll also notice that ASCII, being a seven-bit code, has no character codes above 127, and lacks many of PET/CBM's printing characters.

Because there are so many ASCII control codes, most ASCII keyboards use a special CONTROL key, similar to the SHIFT key, to generate them. CTRL A often sends a 1 code (SOH), CTRL B a 2 code (STX), CTRL C a 3 code (ETX), etc. Also, the meanings of the ASCII control codes, established with commercial message traffic in mind, are almost completely foreign to PET/CBM.

No wonder it's sometimes hard to use non-Commodore devices with your machine! But now that you have Tables 2 and 3, you can write programs for perfect conversions between ASCII and PET/CBM codes. Table 2 shows you exactly what code PET/CBM sends when a given character is transmitted, and Table 3 shows you exactly how an ASCII device will interpret that code. Conversely, Table 3 shows you the intended character representation of every ASCII code your machine receives from outside, while Table 2 shows which code it has to be converted to to have the same representation inside your PET/CBM.

Some Example Conversions

A few examples will illustrate the conversions. Suppose that your PET, with Character Set A enabled, is connected through a modem to an ASCII terminal, and that you are sending messages back and forth. The ASCII terminal sends the lowercase letter "a." Table 3 shows that the code actually transmitted will be 97 decimal, or 0110 0001. If your PET receives that code and displays it on the screen as a PRINTed character, Table 2 shows that it will be displayed as an exclamation point!

So you'll need some software in your PET that converts received ASCII input to CHR$ format before displaying it. In this case, whenever PET receives a 97, the program should convert it to a 65 before PRINTing it. Of course, the program should also be smart enough to convert (or not convert) any of the other ASCII codes between 0 and 127 so that they give the proper display on your PET.

Going the other way, suppose that you press the unshifted "b" key on your PET, and want the distant ASCII terminal to see it as a lowercase "b." Table 2 tells us that your PET will send a 66, which Table 3 tells us the ASCII terminal will interpret as an *uppercase* "B," which is not at all what you want. So your program has to convert the 66 to a 98 before transmitting it, and to make conversions on any other transmitted characters where it's appropriate.

If you study Tables 2 and 3, you'll be able to determine every sending and receiving conversion, and to write your programs accordingly. If the remote device has a character set different from standard ASCII (many of them do), all you need to do is compare it to Table 2, and you'll be able to program the conversions.

```
100 REM    *** CHARACTER SET DEMO ***
120 REM
130 REM SHOWS EVERY PET/CBM CHARACTER
140 REM (KEY PRESS CHANGES CHAR. SET)
150 REM
160 PRINT"{CLEAR}"
170 FORCH=0TO255
180 POKE(32768+2*CH+40*INT(CH/20)),CH
190 NEXTCH
200 FORI=1TO23:PRINT:NEXT
210 PRINTTAB(32)"59468=1";
220 IFPEEK(59468)=14THEN250
230 POKE59468,12:POKE33767,50
240 GETA$:IFA$=""THEN240
250 POKE59468,14:POKE33767,52
260 GETA$:IFA$=""THEN260
270 GOTO230
280 REM
290 REM ** LINE 200 FOR 80 COL. CMB'S:
300 REM
310 FORI=1TO11:PRINT:NEXT
```

In the May issue of **COMPUTE!**, *Steve Steinberg's article "Language Lab" was inadvertently published with the wrong program accompanying it. It is reprinted here in its entirety.*

An ATARI Learning Program

Language Lab

Steve Steinberg
Washington, DC

Language Lab is a program to use your ATARI to help you build vocabulary in a foreign language. It is basically a computerized version of that old standby of language education, the flash card set, and I have found it extremely simple and effective to use. It is structured so that you can drill and score yourself on as many words as you like, but I find it most useful if you display a fifteen or twenty word vocabulary drill on the screen, spot the errors and review them, then try again. If you have enough memory, you can also use it to create a fair sized foreign language dictionary.

I have used a handful of French words in the program example but you can easily change this to any language you want. Simply change line 55 LANG$="FRENCH" to LANG$="GERMAN", "SPANISH", "NAVAJO" or whatever you like and enter the appropriate word pairs in DATA.

The DATA, beginning on line 1000 is easy to expand as your language skill increases and can be used in conjunction with either a self teaching or school language course. Just enter the DATA in word pairs, the first in English, the second in whatever language you are working with.

The key to the vocabulary drills is the random word subroutine, lines 500 through 550. In line 510 X=(1+INT(RND(1)*25)) the 25 is equal to the number of word pairs entered as DATA. As you increase the number of word pairs by adding new DATA this number should also be appropriately increased. You can also alter this line to drill yourself on only part of your total foreign language vocabulary.

Let's assume, for example, that you have 600 word pairs in DATA but only want to drill yourself on the last 100 words you have entered. In that case, change line 510 to X=(500+INT(RND(1)*100)).

Lines 160,180,260,280,330 and 430 use the ATARI cursor advance and line "up" arrow keys to provide a format that will display as much of your language drill or translations on the screen at one time as possible, but you can replace these with just "PRINT" statements if you prefer. This would be useful if you want to use the program for drill and translation of whole phrases and sentences instead of just single words. Don't forget, however, to increase the size of the appropriate string dimensions (ENGLISH$,WORD$,TRANSLATE$) in line 50.

One final note; if you happen to own IRIDIS 2 (and if you have an ATARI computer I don't think you can find a better bargain in software) you can easily add the appropriate subroutine to use Language Lab for Russian, Greek, Hebrew or whatever you wish by adding the foreign alphabet in lower case. I am currently using the program to teach myself Classical Greek. I hope this program will be useful for budding language students and in any case good luck with it, bon chance, and auf wedersehen.

```
40 REM LANGUAGE LAB
41 REM
45 REM BY STEVE STEINBERG
50 DIM LANG$(15),ENGLISH$(20),WORD$(20),
TRANSLATE$(20),Q$(2)
55 LANG$="FRENCH"
60 PRINT "{CLEAR}":PRINT "        ";LANG
$;" LANGUAGE LAB"
65 PRINT :PRINT
70 PRINT "1) ";LANG$;" TO ENGLISH VOCABU
LARY DRILL"
75 PRINT :PRINT "2) ENGLISH TO ";LANG$;"
 VOCABULARY DRILL"
80 PRINT :PRINT "3) ";LANG$;" TO ENGLISH
 TRANSLATOR"
85 PRINT :PRINT "4) ENGLISH TO ";LANG$;"
 TRANSLATOR"
90 PRINT :PRINT "ENTER NUMBER OF DESIRED
 PROGRAM";:INPUT CHOICE:IF CHOICE>4 THEN
 GOTO 90
95 GOTO CHOICE*100
100 REM LANGUAGE TO ENGLISH DRILL
110 PRINT :PRINT :PRINT "HOW MANY WORDS"
;:INPUT N:COUNT=0:SCORE=0
120 ? "{CLEAR}":? "TRANSLATE THE FOLLOWI
NG WORDS INTO     ENGLISH":PRINT
125 GOSUB 500
130 IF COUNT=N THEN GOSUB 600
135 IF COUNT=N THEN GOTO 100
140 PRINT WORD$,
```

```
150 INPUT TRANSLATE$,
160 IF TRANSLATE$=ENGLISH$ THEN SCORE=SC
ORE+1:COUNT=COUNT+1:PRINT "{UP}{30 RIGHT
}CORRECT"
170 IF TRANSLATE$=ENGLISH$ THEN GOTO 125

180 IF TRANSLATE$<>ENGLISH$ THEN COUNT=C
OUNT+1:PRINT "{UP}{11 RIGHT}WRONG! IT'S
";ENGLISH$:GOTO 125
200 REM ENGLISH TO LANGUAGE DRILL
210 PRINT :PRINT :PRINT "HOW MANY WORDS"
;:INPUT N:COUNT=0:SCORE=0
220 ? "{CLEAR}":? "TRANSLATE THE FOLLOWI
NG WORDS INTO     ";LANG$:PRINT
225 GOSUB 500
230 IF COUNT=N THEN GOSUB 600
235 IF COUNT=N THEN GOTO 200
240 PRINT ENGLISH$,
250 INPUT TRANSLATE$,
260 IF TRANSLATE$=WORD$ THEN SCORE=SCORE
+1:COUNT=COUNT+1:PRINT "{UP}{30 RIGHT}CO
RRECT"
270 IF TRANSLATE$=WORD$ THEN GOTO 225
280 IF TRANSLATE$<>WORD$ THEN COUNT=COUN
T+1:PRINT "{UP}{11 RIGHT}WRONG! IT'S ";W
ORD$:GOTO 225
300 ? "{CLEAR}":PRINT LANG$;" TO ENGLISH
 TRANSLATOR":PRINT :PRINT "ENTER ";LANG$
;" WORD"
310 ? :INPUT TRANSLATE$
320 READ ENGLISH$,WORD$
330 IF WORD$=TRANSLATE$ THEN PRINT "{UP}
{15 RIGHT}";:PRINT ENGLISH$:RESTORE :GOT
O 410
340 IF ENGLISH$<>TRANSLATE$ THEN GOTO 32
0
400 ? "{CLEAR}":PRINT "ENGLISH TO ";LANG
$;" TRANSLATOR":PRINT :PRINT "ENTER ENGL
ISH WORD"
410 ? :INPUT TRANSLATE$
420 READ ENGLISH$,WORD$
430 IF ENGLISH$=TRANSLATE$ THEN PRINT "{
UP}{15 RIGHT}";:PRINT WORD$:RESTORE :GOT
O 410

440 IF WORD$<>TRANSLATE$ THEN GOTO 420
500 REM RANDOM WORD SUBROUTINE
510 X=(1+INT(RND(1)*25))
520 RESTORE
530 FOR M=1 TO X:READ ENGLISH$,WORD$
540 NEXT M
550 RETURN
600 REM SCORE SUBROUTINE
610 PRINT :PRINT "OUT OF ";N;" VOCABULAR
Y WORDS YOU HAVE    CORRECTLY TRANSLATED
 ";SCORE;"."
620 PRINT :PRINT "YOUR SCORE IS ";INT(SC
ORE*(100/N));" PER CENT"
630 PRINT :PRINT "GO AGAIN (Y OR N)";:IN
PUT Q$:IF Q$="Y" THEN RETURN
640 IF Q$="N" THEN END
1000 DATA ONE,UN,TWO,DEUX,THREE,TROIS,FO
UR,QUATRE,RED,ROUGE,BLUE,BLEU,GREEN,VERT
,MAN,HOMME,CHILD,ENFANT
1010 DATA HAT,CHAPEAU,PENCIL,CRAYON,HAM,
JAMBON,EGG,OEUF,CITY,VILLE,COUNTRY,PAYS,
OF,DE,UNDER,SOUS,MONDAY,LUNDI
1020 DATA TUESDAY,MARDI,WEDNESDAY,MERCRE
DI,SATURDAY,SAMEDI,SUNDAY,DIMANCHE,YES,O
UI,NO,NON,OLD,ANCIEN
```

©

A map of the significant machine language routines in the VIC Super Expander. You can translate these hexadecimal numbers into decimal, then SYS to them and watch the effects.

VIC Super Expander Memory Map

Chuan Chee
St. Catharines, Canada

General Input/Output Routines

A000-A001 **Vector:** RESET ($A044)
A022-A003 **Vector:** NMI ($A077)
A004-A008 ROM identification ('a0CBM')
A009-A010 **Table:** function key numbers
A011-A043 **Table:** initial function key definitions
A044-A076 RESET routine
A077-A08A NMI routine
A08B-A0BE Parse KEY (get parameters and check syntax)
A0BF-A131 Display all function key definitions
  A110- A11C Print '"+chr$(34)' and an optional '+'
  A11D- A131 Print '"+chr$(13)' and an optional '+'
A132-A135 **Table:** ASCII string for output ('key' backwards)
A136-A13F **Table:** ASCII string for output ('"+chr$(13)' backwards)
A140-A149 **Table:** ASCII string for output ('"+chr$(34)' backwards)
A14A-A17A Delete current function key string (key number in .X)
A17B-A1B0 Insert string into function key definition area
A1B1-A1BE Locate function key definition (key number in .X, return index in .Y)
A1BF-A213 **Table:** new BASIC keywords in ASCII form
A214-A237 **Table:** vectors corresponding to new BASIC tokens ($CC to $DD)
A238-A2A1 Initialize kernal vectors, I/O, RAM
A2A2-A2C1 **Table:** kernal vectors (L,H)
A2C2-A2C7 Warm start routine
A2C8-A317 Output a character to device 3 (char in .A)
A318-A336 End music mode
A337-A365 Interpret keyboard matrix input
A366-A369 **Table:** keyboard matrix code for function keys
A36A-A371 **Table:** conversion pattern for function keys
A372-A394 IRQ routine
A395-A3A5 Input a char from any device (device number in $99)
A3A6-A3B3 Output a char to any device (char in .A, device num in $9A)
A3B4-A3F1 Input each char from keyboard buffer
  A3B4- A3E7 Handle 'RUN' key
  A3E8- A3F1 Handle 'RETURN' key
A3F2-A3FC Input from device 0
A3FD-A406 Print an error message in GRAPHIC 0 mode (error token in .A)
A407-A4B9 Lexically analyse BASIC source line (translate to tokens)
A4BA-A503 Print BASIC tokens in ASCII form
A504-A529 Start new BASIC statement
  A515- A523 Handle new tokens ($CC to $D6)
A52A-A58A Get and evaluate an expression
  A558- A58A Handle new function tokens ($D7 to $DD)
A58B-A596 **Table:** BASIC vectors for RAM
A597-A5A4 Change BASIC vectors during RESET

Music Routines

A5A5-A5D0 Save current sound table (address of table in .X,.Y)
A5D1-A601 IRQ music driver
A602-A625 **Table:** conversion for note index to frequency
A626-A6E5 Interpret music mode characters (char in .A)
  A629-A643 Execute 'O' command, default 3
  A644-A65D Execute 'T' command, default 0
  A65E-A674 Execute 'S' command, default 4
  A675-A686 Execute 'V' command, default 7
  A687-A693 Execute 'R' command
  A694-A69B Execute 'P' command
  A69C-A6A7 Execute 'Q' command
  A6A8- Play new note (note index in .Y)
  A6AB-A6B3 Save new sound table when previous note finishes
  A6B4-A6B9 Common return routine
  A6BA-A6CD Play notes 'A' to 'G'
  A6CE-A6DA Execute '#' command
  A6DB-A6E5 Execute '$' command
A6E6-A6EC **Table:** conversion for notes to note index
A6ED-A6EF **Table:** conversion for octave to base note index
A6F0-A6F9 **Table:** conversion for tempo to duration (jiffies)

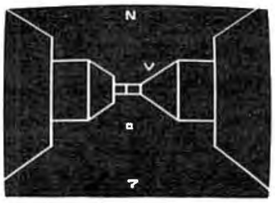
Parsing New Command Routines

A6FA- Look for and evaluate first 1-byte and two 2-byte parameters
A6FD- Look for and evaluate two 2-byte parameters
A700- Look for and evaluate one 2-byte parameter
A714-A71B Save one 1-byte parameter (parameter in .A, index in .Y)
A71C- Look for and evaluate two 1-byte parameters
A71F-A72B Look for and evaluate one 1-byte parameter
A72C-A73F Parse GRAPHIC (get parameters and check syntax)
A740-A762 Parse CIRCLE
A763-A7A4 Parse DRAW
A7A5-A7BC Parse POINT
A7BD- Parse COLOR
A7C8-A7CE Go to execute commands after parsing
A7CF-A7D8 Parse REGION
A7D9-A7DC Parse SCNCLR
A7DD-A7E9 Parse SOUND
A7EA-A809 Parse CHAR
A80A-A810 Parse PAINT

VIC-20* VIC-20*

VICTORY SOFTWARE INC.*

WE SELL FUN!™



TREASURES OF THE BAT CAVE **$19.95**
Battle the vampire bats as you search their cave for gold bullion. Fast, real time action will keep you playing for hours. Of course, you are in a different cave every time you play.

COSMIC DEBRIS **$14.95**
This highly addictive arcade type game will keep you battling the aliens for days.

GRAVE ROBBERS **$13.95**
Introducing the first GRAPHIC ADVENTURE ever available on the VIC-20! With realistic audio-visual effects, you explore an old deserted graveyard and actually see the perils that lie beyond.

NIGHT RIDER **$12.95**
High speed night time driving simulator.

STREET SWEEPERS **$14.95**
Gobble up all of the dots in the maze before the ensuing nemesis gets you. The maze is different every time, and if you succeed in getting all the dots, you get progressively harder mazes to complete as your skills increase. Does this sound like Pac Man? It isn't! Highly recommended and extremely addicting.

Send for free catalog.
All programs fit in the standard VIC memory, and can be controlled from the keyboard.
All programs on cassette tape.
Ordering—Please add $1.50 postage and handling per order.
PA residents please add 6% sales tax.

VICTORY SOFTWARE INC.
2027-A S. J. Russell Circle, Elkins Park, PA 19117
(215) 576-5625

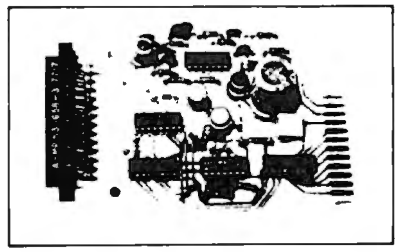
VIC-20® VIC-20®

NEW VIC-20 AND PET PRODUCTS

*****COMMUNICATION CARTRIDGE*****
Makes VIC into dumb terminal with selectable parameters.
Model MW-300 49.95

*****V-DATA BASE CASSETTE*****
Allows user to input mail lists, and mini inventories.
Model MW-301 29.95

*****8 BIT DIGITAL TO ANALOG CONVERTER AND 8 CHANNEL ANALOG TO DIGITAL CONVERTER*****
Allows your VIC or PET to output analog signals. Plugs into your parallel users port. Has capability for 8 channel analog to digital conversion.



Model MW-304p PET 139.95
Model MW-304v VIC 129.95

MICRO WORLD ELECTRONIX INC.
6340 W. Mississippi Ave.
Lakewood, Colorado 80226 (303) 934-1973

WUNDERWARE® PRESENTS GAMES FOR YOUR VIC-20

These games and many more! Send 50¢ (refundable with order) for catalog and free program listing

THE MAD PAINTER
This game is a little unique and a lot of fun. You control a paint brush, moving it around a colorful maze. Your job is to paint the entire maze. This is not as easy as it sounds becose in the maze with you are two voracious Bristle Biters (they love paint brushes). Occasionally you will receive a visit from an Invisible Stomper who leaves footprints in you fresh paint. Requires joystick
$9.95

SPACE RESCUE!
Evil forces have trapped three of your men deep in the perilous depths of the dreaded Dromedary Caverns on the planet of Kam-El. You can rescue them but the tunnel to the lower chamber is narrow and full of hazards. A single player game against time. Space Rescue keeps the top score of the day on the screen. Requires joystick
$9.95

SNAKE!
A fast and fun action game for one player. You're a big snake roaming around the screen. Mice, rabbits, eggs and feet appear at random. Your mission in life is to bite these targets. You have to be quick—the targets don't stay for long. The main problem is, you always seem to be running into the wall or into yourself (the longer you play, the longer, and harder to avoid your tail)! Snake! Keeps high score and requires a joystick.
$9.95

- Price includes Postage & Handling.
- Foreign orders and COD's; Please add $3.00.

Catalog is included with order.
Prices are subject to change without notice
Send check or money order to:

wunderware
P.O. BOX 1287, JACKSONVILLE, OR 97530

VIC-20 is a registered trademark of Commodore Business Machines.

VIC-20 Assembler

EDITOR Used to create and update the assembly language source-program library. It syntax-checks each statement as it is entered, allowing only valid mnemonics and addressing modes. Menu options are Create, Print, Load, Delete, Insert, Change, and Save.

ASSEMBLER Inputs source-program segments created with The Editor. It lists the original source as well as the generated machine language in hexadecimal or decimal, and optionally produces machine language load segments for The Loader. Printer output is supported.

LOADER The Loader inputs the load segments from the library created by The Assembler. The machine language program is POKEd into memory and BASIC's top-of-memory pointer may be automatically reset to a value specified in the source program. The loader may be easily incorporated into another BASIC program or it may be run stand-alone and will SYS to the entry address specified in the source program.

SIGNIFICANT FEATURES

The Assembler can assemble program segments of up to 99 statements on the 5K VIC-20 and several hundred statements on larger systems. It can assemble an unlimited number of such segments in sequence, creating a load segment for each. It is necessary only to include an EQU for every referenced label which occurs outside the segment. There is no limit to the number of labels used or referenced, and labels may be of any length.

The BYT pseudo-op may define literal and hex strings of any length as well as decimal data and address constants in high-low format.

Operand fields may include complex algebraic expressions utilizing the + - * / ↑ (integer and fractional powers supported), & (logical AND), ! (logical OR) operators, and any number of terms in combinations of decimal, hexadecimal, symbolic address and literal formats. Negative numbers generate twos complement values.

Everything runs fast, is easy to use, pleasant to look at, and does just what you expect it to.

$24.95 + $1.00 postage & handling. VISA/MC accepted with $1.00 service charge and inclusion of acct. # and expiration date. Send payment to French Silk, P.O.Box 207, Cannon Falls MN, 55009.

VIC-20 is a registered trademark of Commodore Business Machines.

French Silk smooth ware

A811-A817	Parse RPOT
A818-A81B	Parse RPEN
A81C-A81F	Parse RSND
A820-A823	Parse RCOL
A824-A827	Parse RGR
A828-A842	Parse RDOT
A843-A846	Parse RJOY
A847-A84E	Look for first 1-byte parameter
A84F-A866	Indirect jump to execute new commands (pointer to parameter save area in .X,.Y, command index in .A)
A867-A878	**Table:** vector to execute new commands (H)
A879-A88A	**Table:** vector to execute new commands (L)

Execute New Command Routines

A88B-AA22	Execute GRAPHIC
A8AB- A94E	Handle GRAPHIC 1,2,3 if previous was 0.
A8D4-A942	Transfer BASIC program to above $2000 and execute CLR
A943-A94E	Make screen at $1E00 and character set at $1000
A94F-A9AB	Handle GRAPHIC 4
A967-A9AB	Transfer BASIC program down to old location and execute CLR
A9AC- A9B7	Handle GRAPHIC 0 if previous was 1,2,3
A9B8- AA22	Set up proper GRAPHIC screen
AA23-AA28	Execute RGR
AA29-AA6A	Execute COLOR
AA6B-AA84	Execute REGION
AA85-AA8B	Execute RCOL
AA8C-AAE6	Execute RDOT
AAE7-AAF1	Execute POINT
AAF2-AB12	Execute SCNCLR
AB13-AB22	Execute DRAW (c TO x,y ...)
AB23-AB34	Execute DRAW (c,x,y TO x,y ...)
AB35-AB54	Execute SOUND
AB55-AB69	Execute RSND
AB6A-AB76	Execute RPOT
AB77-AB7D	Execute RPEN
AB7E-	Plot a single point from parameter save area
AB86-ABE4	Plot a single point from beginning scaled X,Y coordinates
ABE5-	Set up pointers to character and colour memory
ABFA-AC0A	Set up pointer to colour memory
AC0B-	Draw a line with a new starting coordinate
AC11-AC92	Draw a line starting from previous coordinate (using a version of Bresenham's DDA algorithm)
AC93-AD12	Execute CIRCLE (using principal of digital differential analyser (DDA))
AD13-	Convert starting angle to radians
AD19-AD22	Divide FAC#1 by 16
AD23-	Calculate new scaled X and Y coordinate on locus
AD39-AD6B	Calculate unit offset * scaled radius
AD6C-ADDE	Execute PAINT
ADDF-	Check for possible new lower bound pivot coordinate
ADE8-AE01	Save pivot coordinate
AE02-AE0B	Check for possible new upper bound pivot coordinate
AE0C-	Check if able to PAINT a coordinate
AE0F-AE1E	Check if able to PAINT a coordinate (X,Y in .A,.Y)
AE1F-AE23	Move beginning scaled X,Y coordinate to .A,.Y
AE24-AE3B	Check if coordinate has been already plotted on
AE3C-AE44	Move beginning scaled X coordinate to the right
AE45-AE51	Move beginning scaled X coordinate 2 to the left
AE52-AE56	Flag 'FORMULA TOO COMPLEX' error message
AE57-AED9	Execute CHAR
AEDA-AF13	Execute RJOY
AF14-AF33	Set up correct VIC chip screen registers
AF34-	Save number of coordinates and colour register
AF39-AF3E	Save colour register
AF3F-AF47	Copy beginning from ending scaled X,Y coordinate
AF48-AF75	Scale X and Y coordinates
AF76-AFB0	Scale X or Y coordinate to the range 0 to 159 (.X = .A*coordinate*2/256) (number of columns or rows in .A)
AFB1-AFBA	**Table:** vector to map Y coordinate to colour memory (L)
AFBB-AFCE	**Table:** vector to map X coordinate to character memory (L)
AFCF-AFE2	**Table:** vector to map X coordinate to character memory (H)
AFE3-AFE5	**Table:** bit set for colour memory
AFE6-AFE6	(not used – contains $00)
AFE7-AFEE	**Table:** bit mask for highres mode
AFEF-AFF6	**Table:** bit mask for multicolour mode
AFF7-AFFA	**Table:** bytes to plot in multicolour mode
AFFB-AFFE	**Table:** conversion factor for VIC chip screen registers
AFFF-AFFF	(not used – contains $AA)

Note:

(H):	high byte of a two-byte address
(L):	low byte of a two-byte address
Vector:	two-byte address used for indirection of execution
Pointer:	two-byte address for data
Index:	one-byte offset for a table

General RAM Area

0024	Number of coordinates
0024	**Flag:** colour register mode ($FF = multicolour, $00 = highres)
0024-0025	**Pointer:** New start of variables / start of BASIC memory
0026	Temp area for building VIC chip registers / for building character byte / for saving start of BASIC (L)

Current Coordinates

0062	Ending scaled X coordinate (0 to 159)
0063	Beginning scaled X coordinate (0 to 159)
0064	Scaled X difference (absolute value)
0065	Ending scaled Y coordinate (0 to 159)
0066	Beginning scaled Y coordinate (0 to 159)
0067	Scaled Y difference (absolute value - 1)

World's Most Cost Effective Development System

- Type your programs directly from your Commodore VIC-20 keyboard into the built-in 4 kilobyte ROM emulator
- Jumper direct to target ROM socket
- Test programs in circuit
- Burn them direct to EPROM with built-in EPROM programmer and power supply
- Burns & runs EPROMS for the Commodore VIC-20, too
- Software on EPROM included
- Comprehensive 25 page manual
- Fits EXP PORT or SUPER EXP Leaves user PORT free

PROMQUEEN CARTRIDGE COMPLETE ONLY $169.50

manual available separately for $20.00 credited on cartridge purchase
Jumper Cable, 2 ft: $18.50 with cartridge; 2732A EPROMS $12.50 each with cartridge

Gloucester Computer Bus Co. 6 Brooks Rd., Gloucester, MA 01930
617-283-7719

Shipping charges for all items extra, Mass. residents add 5% tax.
VISA AND MASTERCARD ACCEPTED

TAYLORMADE SOFTWARE • TAYLORMADE • TAYLORMADE SOFTWARE • TAYLORMADE

for the COMMODORE VIC-20

AEROBICS POINTS CALCULATOR

Now you can monitor your fitness program. Enter your time and (distance) to learn your exact aerobics points earned and calories used. For distance activities learn your yards/min., M.P.H., and min./mile. Save your weekly, monthly, and year aerobics points totals on the tape datassette. See a bar graph on the screen of your average weekly aerobics points for each month. Easy to use, all selections are made from a menu by a single keystroke.

Two programs come on one tape cassette: One for the basic VIC-20 (3583 bytes) will calculate aerobics points for running, walking, cycling and swimming. The second program requires 3K memory expansion, and will also calculate aerobics points for 25 additional activities.

Cassette and manual...U.S. $24.95
(U S and Canada shipping/handling $1.50)
(Foreign shipping/handling $3.00)

TAYLORMADE SOFTWARE
P.O. Box 5574
Lincoln, NE 68505
(402) 464-9051

MasterCard VISA

VIC is a trademark of Commodore Business Machines Inc

TAYLORMADE SOFTWARE • TAYLORMADE

COMMODORE DEALER

VIC-mendous Software

Clowns	Tank Invader
Air Targets	Break-Away
Depth Charge	Dungeon X
Easy Hi-Res.	Easy Multi-Color
Guided Missile	Gunner
Nuclear Attack	Replace
Robots	Tank Battle

Each For Only $9.95

Chomper $12.95
Brain Warp.......... $12.95
Music Machine $12.95
CW Trainer.......... $15.95

RANDOM ACCESS COMPUTERS
120 Benning Drive
P.O. Box 1453
Destin, Florida 32541
Ph: 904-837-7201

COMMODORE DEALER

MIGHTY "V" TO THE RESCUE

VIC 20®
BUSINESS AND HOME SOFTWARE
That'll KNOCK your Mainframe OUT!!
BY TSASA

Now available on cassettes –

☐	1. THE MORTGAGE CALCULATOR C	$ 8.95
☐	2. THE GASOLINE UN-GUZZLER C	$ 8.95
☐	3. THE BILLING SOLVER	$19.95
☐	4. THE CLIENT TICKLER C P	$19.95
☐	5. THE TAPE GENI C P	$ 9.95
☐	6. THE ACCOUNTANT C P	$29.95
☐	7. THE BAR CHART C P	$ 8.95
☐	8. CASH FLOW C	$13.95
☐	9. THE DEPRECIATOR C	$10.95
☐	10 BUSINESS INVENTORY C P	$18.95
☐	11 THE BIDDER C	$14.95
☐	12. THE TEMPERATURE CONVERTER C	$ 8.95
☐	13. FIND THE WORDS C P	$10.95
☐	14. DEAR DIARY C	$12.95
☐	15. PROFIT SHARING PLAN C	$18.95
☐	16. LEASE/BUY? C	$14.95
☐	17 RATIOS C	$ 8.95
☐	18. PRESENT VALUE C	$12.95
☐	19. THE SYNDICATOR C	$15.95
☐	20. MOTHER'S RECIPES C P	$12.95
☐	21 THE TAPE WORM C	$12.95
☐	22 THE CLUB LISTER C	$13.95
☐	23 AT HOME INVENTORY C	$12.95
☐	24. UTILITY BILL SOLVER C	$12.95
☐	25. VICKY-CALC C	$12.95
☐	26. BIRTHDAY GUESSER C	$ 8.95
☐	27 THE METRIC BRAIN C	$10.95
☐	28 ROLL DA' DICE C	$12.95
☐	29 THE TICKER TAPE C	$16.95
☐	30 THE SUPER BROKER C	$12.95
☐	31 THE MORTGAGE COMPARER C	$ 8.95
☐	32. THE RULE OF '78 C	$ 9.95
☐	33. MATH-A-MAGICIAN C	$14.95
☐	34. HOME BUDGET C	$12.95
☐	35 IT'S TRAVEL TIME C	$10.95
☐	36. YOUR MEDICAL RECORDS C P	$14.95
☐	37 THE INVESTMENT ANALYST C	$12.95
☐	38. THE PHONE DIRECTORY C P	$ 9.95
☐	39 THE AMORTIZER P	$24.95
☐	40 THE CHECKBOOK BOOKY C P	$12.95
☐	41 THE INCOME & EXPENSER C P	$14.95
☐	42 THE LOAN REPAYER C	$10.95
☐	43. MY CREDIT CARDS C P	$14.95
☐	44 GRADE MY KIDS C P	$14.95
☐	45. NET WORTH STATEMENT C P	$14.95
☐	46 THE SAVINGS ACCOUNT MIZER C	$13.95
☐	47 CALENDAR MY APPOINTMENTS P	$16.95
☐	48 MIXED UP WORDS C	$ 8.95
☐	49 THE BUDGETER C	$15.95
☐	50 PRINT MY PROGRAM P	$ 8.95

JUST RELEASED

☐	51 JOGGER'S LOGGER C	$11.95
☐	52 NUISMATIC PHANATIC C	$12.95
☐	53 SUPER SHOPPER C P	$12.95
☐	54 P E R T MY VIC C	$16.95
☐	55 LINEAR REGRESS C	$16.95
☐	56 STATISTIC SADISTICS C	$16.95
☐	57 TRACK MY WEIGHT C	$12.95
☐	58 LIGHTNING SORT C	$12.95
☐	59 GOLF SCORECARD C	$11.95
☐	60 ORDER TRACKER C P	$18.95

© Copyright TSASA 1981

FREE CATALOG

Specify CRT TV (C) or (P) Printer
Programs available direct from TSASA or at fine Computer Dealers Nationwide

DEALER INQUIRIES WELCOME

Send check or MO with $1.50 Postage & Handling
COD add $1.50
TSASA

NAME ____________
ADDRESS ____________
CITY ____________
STATE ________ ZIP ________

☐ SEND FREE CATALOG
Mail to: 2 Chipley Run
West Berlin, NJ 08091
Telephone (609)346-3063

For Scaling Coordinates

0069	Multiplicand - 1
006A	16-bit product
006B-006C	16-bit multiplier

For DRAW

0069	Scaled X unit direction - 1
006A	Scaled Y unit direction
006B-006C	Number of scaled Y units left to plot before next scaled X unit (count up)
006D-006E	Number of points left to plot (count up)

For PAINT

0069	**Index:** pivot coordinates save area

For CHAR

0069	Current row (0 to 19)
006A	Current column (0 to 19)
006B	Length of string
006C-006D	**Pointer:** string location

General RAM Area

009B	**Index:** begining of current function key definition
009B-009C	**Pointer:** current char set address / byte in char set / position in screen memory / destination of byte of BASIC program to transfer
009D	**Index:** end of function key definition area
009E	Current function key number / length of current function key string
009F	Length of current function key string (count down)
009E-009F	**Pointer:** byte in colour memory
00AC-00AD	**Pointer:** current byte (function key definition, tape, scrolling)
00C3	**Flag:** 0 = have transferred BASIC program to a new location
00C3-00C4	**Pointer:** kernal set up / current music table / parameter save area ($033C)
00FB-00FC	**Pointer:** top of BASIC memory (usually same as $0284-$0285)

For Key

028F-0290	**Vector:** interpret keyboard input ($A337)
02A1	Number of bytes taken by Super Expander in high memory ($88)
02A2	Number of characters in function key definition
02A3	**Index:** current byte of function key string
02A4	Length of function key string (amount left to output)

For Music

02A5	Previous character in music mode
02A6	Music mode ($80 = on)
02A7	Screen echo ($50 = on, $00 = off)
02A8	Current voice (sound register - 1)
02A9	Current note index
02AA	Current duration (jiffies)
02AB	Current sound amplitude (volume * 2)
02AC	Current octave (base note index)
02AD	Voice 1 note index (+$80)
02AE	Voice 1 duration count down (jiffies)
02AF	Voice 2 note index (+$80)
02B0	Voice 2 duration count down (jiffies)
02B1	Voice 3 note index (+$80)
02B2	Voice 3 duration count down (jiffies)
02B3	Voice 4 note index (+$80)
02B4	Voice 4 duration count down (jiffies)
02B5-02BF	(for expansion)

For Execution Of New Commands

02C0-02C2	**Jump table:** execute new commands (JMP $A84F)
02C3	Current VIC chip left margin register
02C4	Current VIC chip top margin register
02C5	Current VIC chip number of columns register
02C6	Current VIC chip number of rows register
02C7	Current row of cursor
02C8	Current GRAPHIC mode
02C9	(for expansion)
02CA	Current colour register parameter (while plotting)
02CB	Current screen colour
02CC	Current border colour
02CD	Current character colour
02CE	Current auxiliary colour
02CF	**Index:** parameter save area (while plotting)
02D0	Current character set address page
02D1	Usual character set address page ($80)
02D2-02D3	**Pointer:** old limit of BASIC memory
02D4	Old screen memory page
02D5	Last scaled X coordinate (0 to 159)
02D6	Last scaled Y coordinate (0 to 159)
02D7	**Flag:** $00 = DRAW c,x,y TO, $01 = DRAW c TO / current number of out of range coordinates ($00 = within range)
02D8	Old number of out of range coordinates ($00 = within range)
02D9	**Index:** parameter save area (while getting parameters)
02DA-02FF	(for expansion)

Operating System Vectors

0300-0301	**Vector:** error message	($A3FD)
0302-0303	**Vector:** BASIC warm start	($C483)
0304-0305	**Vector:** lexically analyse BASIC source line	($A407)
0306-0307	**Vector:** print BASIC tokens in ASCII form	($A4BA)
0308-0309	**Vector:** start new BASIC statement	($A504)
030A-030B	**Vector:** get and evaluate an expression	($A52A)
0314-0315	**Vector:** IRQ	($A372)
0316-0317	**Vector:** BRK instruction	($A2C2)
0318-0319	**Vector:** NMI	($FEAD)
031A-031B	**Vector:** BASIC OPEN statement	($F40A)
031C-031D	**Vector:** BASIC CLOSE statement	($F34A)
031E-031F	**Vector:** set input	($F2C7)
0320-0321	**Vector:** set output	($F309)
0322-0323	**Vector:** restore I/O	($F3F3)

0324-0325	**Vector:** input a character	($A395)
0326-0327	**Vector:** output a character	($A3A6)
0328-0329	**Vector:** test STOP key	($F770)
032A-032B	**Vector:** BASIC GET statement	($F1F5)
032C-032D	**Vector:** abort I/O	($F3EF)
032E-032F	**Vector:** user BRK instruction	($A2C2)
0330-0331	**Vector:** BASIC LOAD statement	($F549)
0332-0333	**Vector:** BASIC SAVE statement	($F685)
0334-033B	(for expansion)	

Save Area

033C-03F8	**Save area:** parameter passing / pivot coordinates (PAINT)	

For Circle

033C	**Index:** X or Y
0347-0348	Old scaled X coordinate on locus
0349-034A	Old scaled Y coordinate on locus
034B-034C	New scaled X coordinate on locus
034D-034E	New scaled Y coordinate on locus
034F-0353	Floating point unit offset X coordinate
0355-0359	Floating point unit offset Y coordinate

©

This concludes a three-part series on getting started in machine language.

Part III

Machine Language: First Steps

Jim Butterfield
Associate Editor

In the two previous episodes, our intrepid hero, F. R. Vescent, has coded the following bar graph program in machine language:

```
027A  A2  30              LDX   #$30
027C  A0  00              LDY   #$00
027E  E8          YLOOP   INX
027F  E0  3A              CPX   #$3A
0281  D0  02              BNE   SKIP
0283  A2  30              LDX   #$30
0285  8A          SKIP    TXA
0286  20  D2  FF          JSR   $FFD2
0289  C8                  INY
028A  CC  00  03          CPY   $0300
028D  90  EF              BCC   YLOOP
028F  A9  0D              LDA   #$0D
0291  20  D2  FF          JSR   $FFD2
0294  60                  RTS
```

After finishing the coding job, he enters it into memory using the machine language monitor like this:

```
.:  027A  A2  30  A0  00  E8  E0  3A  D0
.:  0282  02  A2  30  8A  20  D2  FF  C8
.:  028A  CC  00  03  90  EF  A9  0D  20
.:  0292  D2  FF  60  ..  ..  ..  ..  ..
```

With the machine language program in place, F. R. returns to BASIC and writes:

```
200 DATA 15,10,30,35,28,28,15,0
210 READ V:IF V=0 GOTO 300
220 POKE 768,V
230 SYS 634
270 GOTO210
300 END
```

The program runs properly, but the machine language is in the first cassette buffer, and this makes it hard to SAVE. F. R. uses the following piece of trickery to make the ML coding more tractable: He types:

```
FOR J=634 TO 660:PRINT PEEK(J);:NEXT J
```

634 is the address of the start of the ML program ($027A) and 660 is the address of the last byte ($0294). So what F. R. is doing is writing a line to cause the program to be dumped, byte by byte, to the screen. He will get something like:

```
162  48  160  0  232  224  58...
```

There will be two spaces between each pair of numbers. Now F. R. does some clever screen editing. He places the cursor just before the 4 of 48, and presses the DEL key and the comma. Now the two numbers read 162,48 ... and F. R. moves the cursor ahead to just before the 160 and repeats the sequence. Eventually, he gets a long line of numbers with commas between. He does *not* press RETURN, but backs up to the beginning of the line and types several Insert keys; now he has room to enter the extra information "110 DATA". Now he presses RETURN and the bytes of his machine language program are entered as part of a DATA statement.

There's another line left over and he must repeat the editing sequence to create a 120 DATA ... line and complete the DATA recording of his program. Now he types in line 100:

```
100 FOR J=634 TO 660:READ M:POKE J,M:NEXT J
```

The whole program should now read:

```
100 FOR J=634 TO 660:READ M:POKE J,M:NEXT J
110 DATA 162,48,160,0,232,224,58,208,2,162,48,138,
    32,210,255,200,204,0
120 DATA 3,144,239,169,13,32,210,255
200 DATA 15,10,30,35,28,28,15,0
210 READ V:IF V=0 GOTO 300
220 POKE 768,V
230 SYS 634
270 GOTO 210
300 END
```

Now: whenever we say RUN, the machine language program is POKEd into place (line 100) before the main program uses it. We may now safely SAVE the BASIC program to tape or disk without worrying about how to save the machine language part. The BASIC program makes its own machine language where it needs it.

Serious programmers will think of more efficient ways to convert the program into easily SAVEable code. More advanced machine language programmers will find better ways than DATA statements. But this is a start.

We've traced a machine language program through its conception, assembly, and implementation. It's not too hard a job, and there are aids to help you along the way.

This example was picked because it presented a fairly easy challenge. Even so, it took us three installments to see it through. The bigger jobs are not much harder; most of the work was in the housekeeping.

For beginners who have never followed the whole process through, it's a worthwhile exercise. Once you've seen the trick done, you can see how to do your own. ©

This is the safety net you need to avoid losing hours of programming time when a power failure or a keyboard lockout strikes your Atari.

Atari Variable Table Refresh And Program Backup

Jon Harding
Rochester, NY

Writing new programs, especially long ones, usually requires many hours of editing and typing which can lead to two problems: (1) the variable name table grows too long and (2) the editing session may be catastrophically terminated by a keyboard lockup or power failure. Even commercial software may have variable name tables which are larger than necessary. And power failures occur only at the worst possible time: after typing in the 252nd line of that new game you're so anxious to try! The following routine eliminates the first problem and minimizes the losses of the second.

Type in these two programs (but delete REMs to conserve memory), SAVEing the first as "WRKLOAD.SAV" and LISTing the second as "BACKUP.LST." The meaning of the extensions should be clear. These routines will become part of your program, but can be deleted when you've finished typing. Initiate a new program or add to an existing program, already LOADed, by ENTERing "D:BAKUP.LST". After typing and/or editing five to ten lines, type G.32000 <RETURN>. The program in RAM is LISTed to the disk as "WRK," the RAM area cleared, and "WRK" re-ENTERed. Edit or add more code and type G.32000 again.

This time the old "WRK" is RENAMEd "WRK.BAK," the program in RAM becomes "WRK," and is re-ENTERed. The next time, and thereafter, "WRK.BAK" is deleted, "WRK" becomes "WRK.BAK," and the program in RAM becomes "WRK" (and is re-ENTERed).

This procedure provides *father-son (mother-*

daughter) backup and keeps the variable name table as small as possible. When you finish a session, type G.32000 once more to save the latest changes in "WRK." When you start a new session just type ENTER"D:WRK" and you pick up where you left off. If you use this routine to back up other programs on your disk, be sure to save the latest version by its proper name, e.g. "D:GASBILL.UTL." You could also delete the "WRK" files before using the routine on another program, but it's not necessary since they will be replaced anyway.

The beauty of this routine is that it only takes a short command to invoke and can't be started by the program in RAM because of the STOP at 31998. The XIO... code is the special I/O command, explained on page 29 of the *Atari BASIC Reference Manual* which allows you to access DOS utilities without calling DOS, among other things.

Other Possibilities

More generally, this scheme suggests the possibility of a disk with many short utility routines which can be SAVEd for use by themselves or LISTed for use during program writing/editing. Examples might include: listing the disk directory to the screen (without calling DOS), printing the disk directory, a screen viewing of a dumped machine language program, and hex to decimal conversion.

A note of caution: if you write a program which will reside in RAM with the program you're working on, it *must* be complete within itself. In other words, when execution is completed, everything defined must be undefined (variables, dimensions, arrays), all devices OPENed must be CLOSEd, and there must be an END. (The CLR command takes care of undefining anything DIMensioned and clears all variables.) The closing statement of such a routine should look like this: lineno CLOSE #1:CLR:END.

Program 1.

```
32039 REM ***************************
32040 ENTER ":WRK"
32041 REM ***************************
```

Program 2.

```
31997 REM ***************************
31998 ? "SHOULDN'T BE HERE!":STOP
31999 REM *IF 'WRK.BAK' EXISTS, DELETE I
T*
32000 TRAP 32010:XIO 33,#1,0,0,"WRK.BAK"
:TRAP 40000
32005 REM *IF 'WRK' EXISTS, RENAME IT WW
RK.BAK'*
32010 TRAP 32020:XIO 32,#1,0,0,"D:WRK,WR
K.BAK":TRAP 40000
32015 REM *LIST PROGRAM TO DISK AS 'WRK'
*
32016 REM *'RUN WRKLOAD.SAV' ERASES RAM
AND RE-ENTERS 'WRK'*
32020 LIST "D:WRK":RUN "D:WRKLOAD.SAV"
32050 REM ***************************
```

©

How To Use The 6560 Video Interface Chip

Dale Gilbert
Henegar, AL

The 6560 Video Interface Chip, the VIC chip, provides low cost, high resolution, color video to a color monitor or a color television, and it also incorporates a sound generator, A/D converters [*analog/digital*], and even a light pen feature.

The 6560, some RAM, a crystal, a few bus drivers, and a little decode logic is all the hardware that is required to add color and sound to a microprocessor that has an expansion bus.

The VIC capabilities include on-chip sync generation, screen grid size of up to 192 horizontal dots by 200 vertical dots, two character sizes, three independent programmable tone generators, a white noise generator, an amplitude modulator, screen centering, on-chip DMA address generation, and two modes of color operation.

The 6560 VIC is manufactured by MOS Technology, Inc.. Commodore Business Machines incorporates the 6560 in their VIC-20 Computer. I purchased my chip from Falk-Baker Associates, 382 Franklin Avenue, Nutley, New Jersey 07110, for $14.95.

6560 Software

To produce colored characters, VIC addresses two blocks of memory at the same time. This address method produces twelve bits of data. The eight bit block of memory is called the Character Pointer Block (called the *screen memory* on the VIC-20).

The second block of memory is called the Character Color Block (called the *color nibble area* on the VIC-20). This block contains four-bit character color data.

VIC takes the character pointer data, left shifts it three times, and adds the result to the character cell base address contained in bits zero through three of register five.

VIC then puts the result on the address bus which addresses another block of memory called the Character Cell Block (called character bit maps on the VIC-20). This block of memory is eight bits wide. The data obtained from this address is video information on an 8 x 8 character matrix. The matrix is eight bytes high and eight bits wide.

VIC takes the four-bit character color data and, if the MSB is 0, the character matrix will be displayed in high resolution mode. If the MSB is one, the character matrix will be displayed in the multicolor mode.

When the high resolution mode is selected and when bit-3 of register-F is a zero, all one bits of the character cell data will be displayed in the background color and all zero bits will be in the foreground color. The three remaining bits of the character color data specify the color of the foreground. The color of the background is specified by bits four through seven of register-F. If bit-3 of register-F is 1, the one bits of the character cell data will be displayed in foreground color and the zero bits will be displayed in background color.

If bit-3 of register-F is one, all the character cell matrix will have the same color background. If bit-3 is a zero, all the character cell matrix will have common character colors.

When the multicolor mode is selected (MSB of the character color data is one), there is a pairing of bits of the character cell data. The character matrix now is a 4 x 8 dot matrix with each dot's color determined by the code of each pair. The code has four possibilities: 00,01,10,11. If a dot code is 00, its color is the background color specified by bits four through seven of register-F. If the code is 01, the dot color is the same as the external border color specified by bits zero through two of register-F. If the code is ten, the dot color is the foreground color specified by the three bits of the character color data. If the code is 11, the color of the dot is specified by bits four through seven of register-E.

VIC produces a TV raster of up to twenty-two columns by up to twenty-three rows of character matrix surrounded by a border. The base address of the character pointer block contains the first upper left pointer for that character matrix. The base address plus one of the character pointer blocks contains the pointer for the next right character matrix. It is the responsibility of the MPU to manipulate the pointers in the character pointer block of memory. A whole raster of repeated characters (character matrix) can be obtained by just repeating the pointers in the character pointer block.

6560 Hardware

The 6560 VIC has fourteen address pins (A0-A13) and twelve data pins (D0 - D11). When the 02 clock is high, the MPU can place an address on the address pins and read or write data into any of the sixteen eight-bit registers via data pins D0 - D7. VIC decodes the address pins and selects registers zero through F when address 1000 through 100F

(the VIC-20's VIC chip ignores the A 15 line) is placed on the address pins.

The address pins are input pins when 02 is one; if 02 is zero, then these pins are output address pins.

When the 02 clock is one, the MPU can also write or read data to the character pointer RAM, the character color RAM, and the character cell RAM. The character cell memory may be RAM, ROM, or both. The base address of the character cell block and/or the character pointer can be changed by modifying a register in the VIC.

When the 02 clock is low, the VIC addresses memory in such a way that the character pointer RAM and the character color RAM is selected at the same time. VIC must receive character pointer data on D0 through D7 and character color data on D8 through D11 at the same time.

The R/W pin four is an input only pin and must be driven by the MPU when 02 is one and held high when 01 is one.

Pins 38 and 39 are the master clock inputs. The 6560 VIC requires a 14.31818 MHz, two phase, five volt, non-overlapping signals. The master clock uses a standard 14.31818 MHz crystal (4x color) and the delay of 74LS gates to give a non-overlapping signal. Resistors R1 and R2 are used to give extra pull up to CMOS levels. CMOS gates don't seem fast enough for this clock.

Pins 35 and 36 are the system output clock used for system timing and driving the clock of the 6512 MPU (if used). A 6502 MPU can be used by feeding pin 36 to the 00 IN pin on the 6502. Removing the original 00 signal and wiring this new 00 is the only alteration needed on the mother MPU.

Because the 6502 address lines are active when 01 is 1, the expansion address lines to the VIC and its associated memory must be isolated from the MPU bus during this time.

The data bus should also be buffered and gated for this same reason.

The system clocks are five volt, non-overlapping, 1.02 MHz signals.

Pin 19 provides the sound output which must be fed to an amplifier to a drive speaker. The output impedence is approximately 1000 ohms.

Pin 3 is the output pin for the composite sync and the luminance signal. This pin is an open drain which makes it easy to shift to the needed voltage level for a RF modulator, TV first video amplifier, or a black and white CRT monitor.

In the following wiring diagram, diodes K3, K4, K5, and K6; resistors R5, R6, and R7; and C1 make up the level shifter for a TV output or video monitor output.

The VIC is a superior CRT controller for a B/W monitor due to the varying levels of luminance required for a color picture. VIC can produce varying shades of gray.

Pin 2 provides the composite color signal. This signal contains the color phase and amplitude information plus the 3.58MHz burst signal. Pin 2 is a high impedence output buffer which can be applied to the first chroma amplifier of a TV, color monitor, or RF modulator.

Pins 17 and 18 are the input pins for the Pot-X and Pot-Y analog to digital converters. A pot is used to charge an external capacitor tied to the pot wiper and fed to pin 17 or 18. These pins are systematically pulled to ground after each charge voltage reading. The voltage is digitized and deposited in register 8 or 9.

Pin 37 is the light gun/pen pin. The voltage of triggering is approximately 2.5 volts on the falling edge. Holding this pin low clears registers 6 and 7. The values of registers 6 and 7 represent the horizontal and vertical positions of the current dot being scanned. The light gun/pen option is only available on a 6560-101 which is sometimes identified by a white dot on the case of the IC.

Address decoding must be provided so that the character color nibble RAM will be selected, when the character pointer RAM is addressed by the VIC, but not selected when the MPU addresses the character pointer RAM. A data bus transceiver must be provided to isolate the nibble bus from the byte bus along with the logic for this transceiver. The logic must enable the MPU to read or write to the nibble bus when the 02 clock is one.

The expansion 02 may be used as the E(02) if the expansion 02 has no more than one LS gate delay and it is the true MPU 02. If these provisions can't be met, connect E(02) to the V02 (pin 36).

This author used addresses 1000 through 100F for the VIC; 2000 through 23FF for the character pointer block; 2400 through 27FF for the character color block; and 2800 through 2BFF for the character cell block.

The addition of a 6560, a few RAM chips, and a few gates will free up and complement many a MPU.

Color Code

Auxiliary/Background/Border/Foreground

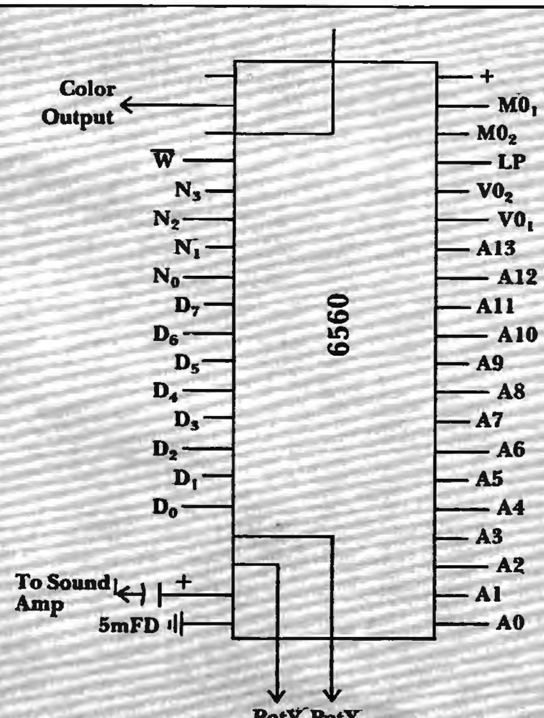
0 BLACK	8 ORANGE
1 WHITE	9 LIGHT ORANGE
2 RED	A PINK
3 CYAN	B LIGHT CYAN
4 MAGENTA	C LIGHT MAGENTA
5 GREEN	D LIGHT GREEN
6 BLUE	E LIGHT BLUE
7 YELLOW	F LIGHT YELLOW

VIC Control Registers

Register	Data	Address
R0	1=I O=N ; Horiz. Center Vx4 Dots 6 5 4 3 2 1 0	1000
R1	Vertical Center Vx2 Dots 7 6 5 4 3 2 1 0	1001
R2	Cp A9 ; No. of Character Matrix Columns 6 5 4 3 2 1 0	1002
R3	RV 0 ; No. of Character Matrix Rows 5 4 3 2 1 0 ; 1=D 0=S	1003
R4	Raster Value 8 7 6 5 4 3 2 1	1004
R5	Base Cp A13 A12 A11 A10 ; Base Char. Cell A13 A12 A11 A10	1005
R6	LP Horizontal Position 7 6 5 4 3 2 1 0	1006
R7	LP Vertical Position 7 6 5 4 3 2 1 0	1007
R8	Pot-X 7 6 5 4 3 2 1 0	1008
R9	Pot-Y 7 6 5 4 3 2 1 0	1009
RA	Sw 1=On ; Frequency Osc. -1 6 5 4 3 2 1 0	100A
RB	Sw 1=On ; Frequency Osc. -2 6 5 4 3 2 1 0	100B
RC	Sw 1=On ; Frequency Osc. -3 6 5 4 3 2 1 0	100C
RD	SW 1=On ; Frequency Noise Gen. 6 5 4 3 2 1 0	100D
RE	Auxiliary Code 3 2 1 0 ; Audio Amplitude 3 2 1 0	100E
RF	Background Code 3 2 1 0 ; 1=B 0=F ; Border Code 2 1 0	100F

Abbreviations:

I = Interlace	S = 8x8 Matrix
N = None Interlace	D = 16x8 Matrix
CP = Character Pointer Base Address	B = Common Background
R = Raster	F = Common Foreground
V = Value	LP = Light Pen
	Sw = Switch



©

This compactor analyzes and compresses BASIC at lightning speed. For PET/CBM, Upgrade and 4.0 BASIC, with disk drive.

Machine Language Compactor

David L. Evans
Caldwell, ID

When the programs "Compactor" and "Un-Compactor" by Robert Baker appeared in the Sept./Oct., 1980 and May, 1981, issues, respectively, of **COMPUTE!** I immediately typed them in and tried them out on several of my programs.

To my dismay, compacting a 16K program required approximately 48-52 minutes. I am a college student and spend a limited amount of time on my computer (a 16K 4.0 PET) due to study time and my job. That compacting time ate heavily into the time I spend on my PET and I decided to do something about it. This program is the result.

I tried several different BASIC versions of the program, but could come up with no noticeable increase in speed. I then decided to give machine language a try. The result? A 947% increase in speed! Even this phenomenal increase in speed can be improved a bit further by eliminating the error checking in my input subroutine.

The program consists of five different sections. The input section prints the directions and inputs the input and output file names. The scanning section scans the BASIC program for target line numbers and prints them for you to see, while also storing them for use by the compacting section. The compacting section compacts the program (where possible) and prints "DONE" when finished.

A following subroutine section contains an input subroutine, a line number checking routine, and a disk error routine. Following these four sections is the data and work storage area. This program was designed to run on both Upgrade and 4.0 ROMs. It will automatically adjust itself to work on either ROM set, so all the user has to do is to LOAD the PROGRAM and type RUN.

Running the program is fairly simple. Load the program into memory with the LOAD or DLOAD command, depending upon which computer you have. If you do not have DOS 2.0 or higher, make sure that both drives are initialized. Type RUN, and the computer will print a short greeting along with some brief directions.

Typing In Compactor

Below are the step-by-step instructions for entering and SAVEing Compactor. **COMPUTE!** generally provides BASIC loaders so that readers who don't know machine language can easily type in and use machine language subroutines and programs. Though it could be done, we felt that creating a BASIC loader for Compactor would be too complicated to be worthwhile: Compactor sits in memory starting at address 1024 (like a BASIC program). A loader would involve an awkward process of writing over itself and checksums would be equally difficult to use.

Instead, Compactor is presented as *hex dump*. To enter it into the computer:

1. Type: SYS 1024 (to enter the machine language monitor).
2. When prompted by the period (.), type M 0400 0460 and the screen will reveal (in hexadecimal numbers) the contents of the memory cells between address 0400 and 0460. You can type over these numbers and hit RETURN on each line, thereby entering the new numbers into memory. So, for the first line, you type: 00 0B 04 FF FF 9E 31 30 (RETURN). Continue on until you've typed and entered line 0460 as it appears in the magazine listing.
3. Then do the next block of memory. Type: M 0468 0500 and continue on, replacing the values on screen with Compactor's numbers as printed. When you've typed in the last line, (0B70), you are done.
4. Now Compactor is in your computer's memory and it must be SAVEd onto a disk. Type: S "0:COMPACTOR", 08,0400,0B78 (RETURN) and it will now be available for use as a program called "COMPACTOR" on the disk in Drive zero.
5. If the power went off just before you finished (and you lost everything, including your temper), you might prefer to take up the author's offer (at the end of the article) to make a copy for you.

After reading the directions, type in your input file name preceded by the drive number the file is on. Example : 0:COMPACTOR. You will then be asked for the output file name; do the same with it as you did with the input file name. The computer will then scan and compact your program. If any disk error is detected, the program will report it and will return to BASIC. After your program has been compacted, LOAD it into memory and type the CLR command; this will fix all the line links. Be sure to reSAVE the corrected copy.

Clearly, machine language lends itself to input/output programs more readily than BASIC does. Once the basics of how to open files from machine language are learned (see **COMPUTE!**, April, 1981, #11), it is easy to generate input/output programs in machine language which will run at a phenomenal speed.

For those of you who do not want to type this in, I will make you a copy of it if you will send me $3.00 and a self-addressed, stamped mailer and a blank tape or disk. For those who send a disk, I have DOS 2.0, so all disks will be written in DOS 2.0.

David Evans
2202 Ellis Ave.
Caldwell, ID 83605

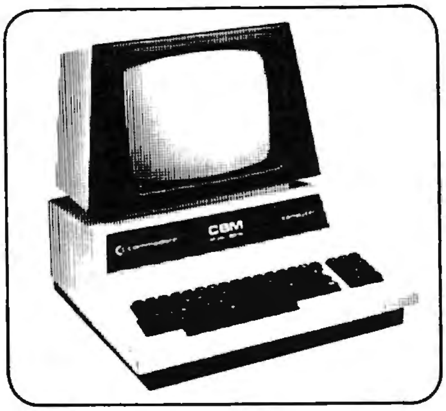
```
Ø4ØØ ØØ ØB Ø4 FF FF 9E 31 3Ø
Ø4Ø8 33 37 ØØ ØØ ØØ AD 5Ø C3
Ø41Ø C9 2E FØ 38 A9 24 8D FØ
Ø418 Ø4 8D 3Ø Ø5 8D EC Ø6 8D
Ø42Ø Ø9 Ø7 A9 AE 8D 69 Ø6 8D
Ø428 6E Ø6 8D A7 Ø6 8D 89 Ø9
Ø43Ø 8D 91 Ø9 8D 99 Ø9 A9 D9
Ø438 8D 88 Ø6 8D 5E Ø7 8D DA
Ø44Ø Ø8 A9 DC 8D 89 Ø6 8D 5F
Ø448 Ø7 8D DB Ø8 A9 78 85 Ø1
Ø45Ø A9 ØB 85 Ø2 AØ ØØ A9 ØØ
Ø458 91 Ø1 C8 DØ F9 E6 Ø2 A5
Ø46Ø Ø2 C9 16 DØ EF A9 ØD 8D
Ø468 88 ØB A9 ØØ 8D 87 ØB A2
Ø47Ø ØØ BD 48 ØA C9 ØØ FØ Ø7
Ø478 2Ø D2 FF E8 4C 71 Ø4 A2
Ø48Ø ØØ 2Ø CF FF 9D 5Ø ØC C9
Ø488 ØD FØ Ø8 E8 EØ 12 FØ Ø3
Ø49Ø 4C 81 Ø4 E8 A9 ØD 9D 5Ø
Ø498 ØC AD 5Ø ØC C9 3Ø FØ Ø4
Ø4AØ C9 31 DØ CB AD 51 ØC C9
Ø4A8 3A DØ C4 A2 ØØ BD Ø9 ØB
Ø4BØ C9 ØØ FØ Ø7 2Ø D2 FF E8
Ø4B8 4C AD Ø4 A2 ØØ 2Ø CF FF
Ø4CØ 9D 7Ø ØC C9 ØD FØ Ø8 E8
Ø4C8 EØ 12 FØ Ø3 4C BD Ø4 AD
Ø4DØ 7Ø ØC C9 3Ø FØ Ø4 C9 31
Ø4D8 DØ 95 AD 71 ØC C9 3A DØ
Ø4EØ 8E A9 ØF 85 D2 85 D3 A9
Ø4E8 Ø8 85 D4 A9 ØØ 85 D1 2Ø
Ø4FØ 63 F5 2Ø F7 Ø9 A2 ØØ BD
Ø4F8 5Ø ØC C9 ØD FØ Ø4 E8 4C
Ø5ØØ F7 Ø4 A9 2C 9D 5Ø ØC 9D
```

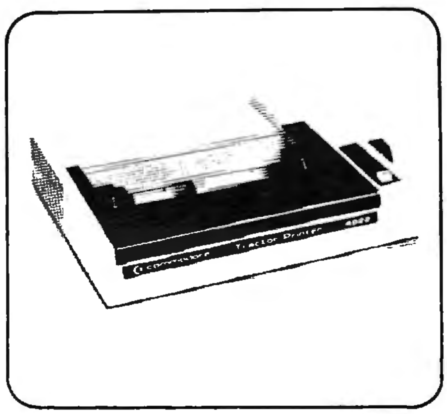
```
Ø51Ø 52 9D 53 ØC E8 E8 E8 E8
Ø518 8E 86 ØB 86 D1 A9 Ø5 85
Ø52Ø D2 85 D3 A9 Ø8 85 D4 A9
Ø528 5Ø 85 DA A9 ØC 85 DB 2Ø
Ø53Ø 63 F5 2Ø F7 Ø9 2Ø CE Ø9
Ø538 2Ø CE Ø9 18 AD 78 ØB 6D
Ø54Ø 79 ØB BØ Ø7 C9 ØØ DØ Ø3
Ø548 4C A1 Ø6 2Ø CE Ø9 AD 78
Ø55Ø ØB 8D 7A ØB AD 79 ØB 8D
Ø558 7B ØB 2Ø D7 Ø9 AD 79 ØB
Ø56Ø C9 ØØ FØ D4 C9 89 FØ Ø8
Ø568 C9 8D FØ Ø4 C9 A7 DØ EA
Ø57Ø A9 ØØ 8D 7C ØB 8D 7D ØB
Ø578 2Ø D7 Ø9 AD 79 ØB C9 3A
Ø58Ø BØ DB 9Ø Ø3 2Ø D7 Ø9 AD
Ø588 79 ØB C9 2Ø FØ EA C9 3A
Ø59Ø BØ 58 C9 3Ø 9Ø 54 A9 ØA
Ø598 85 Ø1 A9 ØØ 8D 7E ØB A2
Ø5AØ Ø8 ØA 2E 7E ØB Ø6 Ø1 9Ø
Ø5A8 Ø9 18 6D 7C ØB 9Ø Ø3 EE
Ø5BØ 7E ØB CA DØ EC 8D 7C ØB
Ø5B8 AD 7D ØB ØA 85 Ø1 ØA ØA
Ø5CØ 18 65 Ø1 6D 7E ØB 8D 7D
Ø5C8 ØB 38 AD 79 ØB E9 3Ø 8D
Ø5DØ 79 ØB 18 AD 79 ØB 6D 7C
Ø5D8 ØB 8D 7C ØB 9Ø Ø3 EE 7D
Ø5EØ ØB 2Ø D7 Ø9 AD 79 ØB 4C
Ø5E8 8E Ø5 AD 7C ØB 8D 83 ØB
Ø5FØ AD 7D ØB 8D 84 ØB 2Ø 9C
Ø5F8 Ø9 AD 85 ØB C9 FF FØ 71
Ø6ØØ AD 87 ØB C9 ØØ DØ Ø7 AD
```

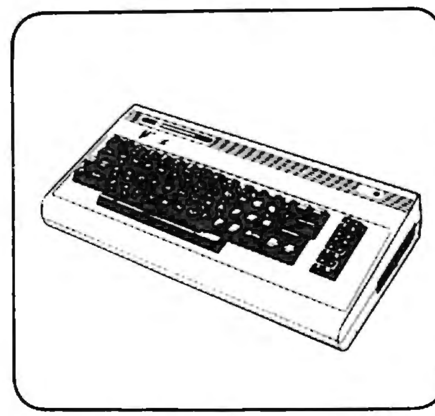
```
Ø61Ø ØB 8D 1E Ø6 AD 88 ØB 8D
Ø618 1F Ø6 AD 7C ØB 8D FF FF
Ø62Ø EE 87 ØB AD 87 ØB C9 ØØ
Ø628 DØ ØA EE 88 ØB AD 88 ØB
Ø63Ø C9 16 BØ 22 AD 87 ØB 8D
Ø638 44 Ø6 AD 88 ØB 8D 45 Ø6
Ø64Ø AD 7D ØB 8D FF FF EE 87
Ø648 ØB AD 87 ØB C9 ØØ DØ Ø3
Ø65Ø EE 88 ØB 4C 71 Ø6 A2 ØØ
Ø658 BD 39 ØB C9 ØØ FØ Ø7 2Ø
Ø66Ø D2 FF E8 4C 58 Ø6 A9 Ø5
Ø668 2Ø E2 F2 A9 ØF 2Ø E2 F2
Ø67Ø 6Ø A2 ØØ BD 1F ØB C9 ØØ
Ø678 FØ Ø7 2Ø D2 FF E8 4C 73
Ø68Ø Ø6 AE 7C ØB AD 7D ØB 2Ø
```

commodore
SHOW SPECTACULAR

8032-32K 80 COL CRT	
REG $1495	**$1065**
64K ADD-ON MEMORY	
REG $500	**$395**
9000 134K SUPER PET	
REG $1995	**$1795**
4032 32K 40 COL CRT	
REG $1295	**$965**
4016 16K 40 COL CRT	
REG $995	**$765**
8050-DUAL DISK 950K	
REG $1795	**$1295**
4040-DUAL DISK 343K	
REG $1295	**$995**
2031-SINGLE DISK 170K	
REG $695	**$525**
C2N-CASSETTE DRIVE	
REG $75	**$65**
4022-80 COL PRINTER	
REG $795	**$595**
8023P-136 COL PRINTER	
REG $995	**$849**
8300P-40CPS LTR QLTY	
REG $2250	**$1995**
8024-MANNESMAN TALLEY	
REG $1995	**$1595**
8024L-LETTER TALLEY	
REG $2495	**$1995**
25CPS-STARWRITER	
REG $1895	**$1445**
CBM-IEEE MODEM	
REG $279	**$229**
VOICE SYNTHESIZER	
REG $395	**$329**







PET TO IEEE CABLE	
REG $39.95	**$34**
IEEE TO IEEE CABLE	
REG $49.95	**$39**
VIC 20	
REG $299	**$265**
VIC 1540 DISK 170K	
REG $599	**$499**
VIC 1515 30CPS PRINTER	
REG $395	**$349**
VIC 1011 RS 232 INTER	
REG $49.95	**$39**
VIC 1112 IEEE INTER	
REG $99.95	**$79**

MATOR SHARK HARD DISK DRIVE

*Winchester Disk
*Commodore DOS 2.5
*24 Megabyte
$6475

Strobe 100 Plotter
*PET Interface
*Software Pack
$975

SCHOOL SPECIALS
3 for 2 Pricing
CALL US!

20% Off on Software

Special pricing on Atari, Apple, Digilog, Epson, Hayes Modem, Printwheels

PHILADELPHIA COMPUTER DISCOUNT ©

P.O. Box 170 St. Davids, PA 19087 (215)687-8540

MAIL ORDER PHONE 1-800-345-1289

PREPAID ORDERS SHIPPED FREE
VISA & MASTERCARD ADD 3%
COD - UPS
PA RESIDENTS ADD 6% SALES TAX

H1181C

```
Ø688 83 CF AD 79 ØB C9 2C DØ
Ø69Ø Ø3 4C 7Ø Ø5 C9 2Ø FØ Ø3
Ø698 4C 5D Ø5 2Ø D7 Ø9 4C 8A
Ø6AØ Ø6 2Ø CC FF A9 Ø5 2Ø E2
Ø6A8 F2 A2 ØØ BD 49 ØB C9 ØØ
Ø6BØ FØ Ø7 2Ø D2 FF E8 4C AB
Ø6B8 Ø6 A2 ØØ BD 7Ø ØC C9 ØD
Ø6CØ FØ Ø4 E8 4C BB Ø6 AØ ØØ
Ø6C8 B9 7Ø ØB C9 ØØ FØ Ø8 9D
Ø6DØ 7Ø ØC C8 E8 4C C8 Ø6 86
Ø6D8 D1 A9 Ø6 85 D2 85 D3 A9
Ø6EØ Ø8 85 D4 A9 7Ø 85 DA A9
Ø6E8 ØC 85 DB 2Ø 63 F5 2Ø F7
Ø6FØ Ø9 AE 86 ØB 86 D1 A9 Ø5
Ø6F8 85 D2 85 D3 A9 Ø8 85 D4
Ø7ØØ A9 5Ø 85 DA A9 ØC 85 DB

Ø71Ø Ø9 A2 Ø6 2Ø C9 FF AD 78
Ø718 ØB 2Ø D2 FF AD 79 ØB 2Ø
Ø72Ø D2 FF 2Ø CC FF A9 ØØ 8D
Ø728 7F ØB 2Ø CE Ø9 AD 78 ØB
Ø73Ø 8D 8Ø ØB AD 79 ØB 8D 81
Ø738 ØB A9 ØØ 8D 82 ØB 18 AD
Ø74Ø 78 ØB 6D 79 ØB BØ Ø7 C9
Ø748 ØØ DØ Ø3 4C 63 Ø9 2Ø CE
Ø75Ø Ø9 AE 78 ØB 8E 83 ØB AD
Ø758 79 ØB 8D 84 ØB 2Ø 83 CF
Ø76Ø 2Ø D7 Ø9 AD 79 ØB C9 2Ø
Ø768 FØ F6 C9 3A FØ F2 C9 ØØ
Ø77Ø FØ ØE C9 8F DØ 19 2Ø D7
Ø778 Ø9 AD 79 ØB C9 ØØ DØ F6
Ø78Ø A9 Ø1 8D 82 ØB 2Ø 9C Ø9
Ø788 AD 85 ØB C9 FF DØ 9B A2
Ø79Ø Ø6 2Ø C9 FF AD 8Ø ØB 2Ø
Ø798 D2 FF AD 81 ØB 2Ø D2 FF
Ø7AØ AD 83 ØB 2Ø D2 FF AD 84
Ø7A8 ØB 2Ø D2 FF 2Ø CC FF A9
Ø7BØ Ø4 8D 7F ØB AD 82 ØB C9
Ø7B8 ØØ FØ 12 A2 Ø6 2Ø C9 FF
Ø7CØ A9 3A 2Ø D2 FF 2Ø CC FF
Ø7C8 A9 Ø5 8D 7F ØB A9 ØØ 8D
Ø7DØ 82 ØB 4C E9 Ø7 A2 Ø6 2Ø
Ø7D8 C9 FF AD 79 ØB 2Ø D2 FF
Ø7EØ 2Ø CC FF EE 7F ØB 2Ø D7
Ø7E8 Ø9 AD 79 ØB C9 89 FØ Ø8
Ø7FØ C9 8B FØ Ø4 C9 A7 DØ Ø5
Ø7F8 A9 Ø1 8D 82 ØB AD 79 ØB
Ø8ØØ C9 ØØ DØ Ø3 4C 9C Ø8 C9

Ø81Ø D7 Ø9 AD 79 ØB C9 ØØ DØ
Ø818 F6 4C 9C Ø8 AD 79 ØB C9
Ø82Ø 22 DØ 46 A2 Ø6 2Ø C9 FF
Ø828 AD 79 ØB 2Ø D2 FF 2Ø CC
Ø83Ø FF EE 7F ØB 2Ø D7 Ø9 AD
Ø838 79 ØB C9 22 FØ 97 C9 ØØ
Ø84Ø DØ E1 AD 82 ØB C9 ØØ FØ
Ø848 ØD A9 ØØ 8D 79 ØB A2 Ø6
Ø85Ø 2Ø C9 FF 4C 1C Ø7 A2 Ø6
Ø858 2Ø C9 FF A9 22 2Ø D2 FF
Ø86Ø 2Ø CC FF EE 7F ØB 4C 9C
Ø868 Ø8 AD 79 ØB C9 3A FØ Ø3
Ø87Ø 4C D5 Ø7 2Ø D7 Ø9 AD 79
Ø878 ØB C9 2Ø FØ F6 C9 3A FØ
Ø88Ø F2 C9 8F FØ 8A C9 ØØ FØ
Ø888 13 A2 Ø6 2Ø C9 FF A9 3A
Ø89Ø 2Ø D2 FF 2Ø CC FF EE 7F
Ø898 ØB 4C E9 Ø7 AD 82 ØB C9
Ø8AØ ØØ DØ Ø7 AD 7F ØB C9 AB
Ø8A8 9Ø ØD A9 ØØ 8D 79 ØB A2
Ø8BØ Ø6 2Ø C9 FF 4C 1C Ø7 2Ø
Ø8B8 CE Ø9 18 AD 78 ØB 6D 79
Ø8CØ ØB BØ Ø7 C9 ØØ DØ Ø3 4C
Ø8C8 63 Ø9 2Ø CE Ø9 AE 78 ØB
Ø8DØ 8E 83 ØB AD 79 ØB 8D 84
Ø8D8 ØB 2Ø 83 CF 2Ø 9C Ø9 AD
Ø8EØ 85 ØB C9 FF FØ 2C 2Ø D7
Ø8E8 Ø9 AD 79 ØB C9 8F DØ Ø3
Ø8FØ 4C ØF Ø8 C9 2Ø FØ EF C9
Ø8F8 3A FØ EB C9 ØØ FØ B8 A2
Ø9ØØ Ø6 2Ø C9 FF A9 3A 2Ø D2

Ø91Ø E9 Ø7 A2 Ø6 2Ø C9 FF A9
Ø918 ØØ 2Ø D2 FF A9 Ø1 2Ø D2
Ø92Ø FF 2Ø D2 FF AD 83 ØB 2Ø
Ø928 D2 FF AD 84 ØB 2Ø D2 FF
Ø93Ø 2Ø CC FF A9 Ø4 8D 7F ØB
Ø938 2Ø D7 Ø9 AD 79 ØB C9 2Ø
Ø94Ø FØ F6 C9 3A FØ F2 C9 ØØ
Ø948 FØ Ø4 C9 8F DØ ØD A2 Ø6
Ø95Ø 2Ø C9 FF A9 3A 2Ø D2 FF
Ø958 2Ø CC FF A9 ØØ 8D 82 ØB
Ø96Ø 4C E9 Ø7 A2 Ø6 2Ø C9 FF
Ø968 A9 ØØ 2Ø D2 FF 2Ø D2 FF
Ø97Ø 2Ø D2 FF 2Ø CC FF A2 ØØ
Ø978 BD 65 ØB C9 ØØ FØ Ø7 2Ø
Ø98Ø D2 FF E8 4C 78 Ø9 A9 Ø6
Ø988 2Ø E2 F2 2Ø F7 Ø9 A9 Ø5
Ø99Ø 2Ø E2 F2 2Ø F7 Ø9 A9 ØF
Ø998 2Ø E2 F2 6Ø A9 ØC 85 Ø2
Ø9AØ A9 ØØ 85 Ø1 A8 AA AD 83
Ø9A8 ØB D1 Ø1 DØ Ø9 C8 AD 84
Ø9BØ ØB D1 Ø1 FØ 13 88 C8 C8
Ø9B8 DØ EC E6 Ø2 A5 Ø2 C9 16
Ø9CØ DØ DE A9 ØØ 8D 85 ØB 6Ø
Ø9C8 A9 FF 8D 85 ØB 6Ø 2Ø D7
Ø9DØ Ø9 AD 79 ØB 8D 78 ØB A2
Ø9D8 Ø5 2Ø C6 FF 2Ø E4 FF 85
Ø9EØ Ø1 A9 Ø1 8D 89 ØB 2Ø F7
Ø9E8 Ø9 A5 Ø1 8D 79 ØB A9 ØØ
```

COMMODORE VIC

PERSONAL COMPUTER

60% more powerful than VIC-20

$299.00

Special Sale Price

- **We Stock More VIC-20 Programs and Accessories than anyone in the USA!**
- **We Stock "United Micro's" VIC-20 Super Programs! Arcade Quality up to 256 Levels-Super Graphics Sound-Fast Action Unrelenting Challenge.**
- **VIC-20 Owners Write for Free Catalog.**

FOR THE SPECIAL SALE PRICE OF $299.00 you get the COMMODORE VIC-20 computer *plus* WE ADD 3000, BYTES OF MEMORY to give you 60% MORE PROGRAMMING POWER! This powerful fullsized extra featured computer includes the 6502 microprocessor (LIKE APPLE) 20,000 bytes ROM with a 16K extended LEVEL II Microsoft BASIC, 8000 bytes RAM plug in expandable to 32,000 bytes RAM, 66 key typewriter professional expanded keyboard with graphic symbols on keys, color command keys, high resolution graphics, 512 displayable characters, text display is 22 lines 23 characters, sound and music, real time, upper lower case, full screen editing cursor, floating point decimal and trig functions, string arrays, scrolling, multi statement lines, file managment, PEEK AND POKE. Assembly machine language is available. We have easy to use self teaching books and programs. Accept TAPE-DISK-PLUG IN CARTRIDGES, connects to any TV, includes AC adaptor, R.F. modulator, switch box, self teaching instruction book, comes in a beautiful console case for ony $299.00.

LOW COST PLUG IN EXPANSION

Expansion accessories plug directly into this computer, extra RAM memory, Controllers, a Cassette, A Telephone Modem for only $109.00, an 80 Column Printer for $375.00, even the 170K Disk Drive plugs in direct. You do not have to buy an expensive expansion interface.

WHY SUCH A LOW PRICE

WE GIVE YOU 60% to 400% MORE PROGRAMMING POWER THAN VIC-20! You can't beat our prices for the VIC-20 with increased programming power added! We sell direct to customers. We save you the profit margin normally made by computer stores, department stores, and distributors. We are willing to take a smaller margin to develop volume!

INVEST IN YOUR CHILDREN

Educate your children while they play. Every kid wants to play electronic games. (We have some of the best). The next natural step for their curiosity is to try simple programming. They can do this in 20 minutes with our simple self teaching instruction book. High schools are teaching computer math, science and programming - some start in grammar school.

If you provide this computer as a Teacher and Tutor at home, before you know it your child will be writing computer programs. You can use your T.V. to EDUCATE not frustrate your family and eliminate T.V. boredom with programs that challenge, stimulate and entertain the whole family. We have a wide variety of games, recreational, home finance and educational programs to choose from. Why pay $140.00 to $295.00 for an electronic game when you can buy this powerful computer for only $299.00.

COMMUNICATE WITH THE WORLD

Plug in your VIC telephone modem. Now you can get a world of information through your telephone, plus electronic mail. Just dial up the information you want. UPI wire service, stock market, historical information by topic from over 60 magazines, including New York times. Airline information, order tickets, get weather information anywhere in the world, restaurant and hotel information, thousands of categories are on line for you, business, finance, education, entertainment, games etc. YOU'LL BE THE TALK OF YOUR NEIGHBORHOOD. Our telephone modem price is by far the lowest available.

IMMEDIATE REPLACEMENT WARRANTY

If your computer fails because of warranty defect within 90 days from date of purchase, you simply send your computer to us via United Parcel Service prepaid. We will "immediately" send you a replacement computer at no charge via United Parcel Service prepaid. No one we know gives you this kind of warranty service. Most computer warranty service takes 30 to 90 days to handle - this fantastic "immediate replacement warranty" is backed by COMMODORE COMPUTER, a MAJOR national brand electronics manufacturer.

SPECIAL SALE PRICE $299

FOR ONLY $299 you get the POWERFUL 28K COMMODORE VIC with 60% MORE PROGRAMMING POWER THAN VIC-20! 28,000 bytes total memory (20,000 bytes ROM, 8000 bytes RAM and extended LEVEL II BASIC), the professional 66 keyboard, color, sound, music self teaching instruction book, A.C. adaptor, R.F. modulator, T.V. switch box, owners manual plus all the other features listed, in a beautiful console.

SPECIAL SALE PRICE $379

FOR ONLY $379 you get the 41K COMMODORE VIC with 400% MORE PROGRAMMING POWER THAN VIC-20! We add 16,000 bytes memory to the VIC-20. You get a total of 41,000 bytes memory (20,000 bytes ROM, 21,000 bytes RAM and extended LEVEL II BASIC) plus all the extra features shown for the 28K COMMODORE VIC.

SPECIAL DATA CASSETTE SALE

THIS SPECIAL DATA CASSETTE has special electronics that eliminates loading problems and loss of programs recorded on tape! Includes tape counter and your selection of any $14.95 "GAME PACK" program FREE!!! Reg. Price $90.00 Sale Price $69.00 (A standard tape recorder will not work).

15 DAY FREE TRIAL

DON'T MISS THIS SALE -ORDER NOW

- ☐ Please send me the 28K Commodore VIC Computer for $299.00
- ☐ Please send me the 41K Commodore VIC Computer for $379.00
- ☐ Special Data Cassette $69.00

We ship C.O.D. and honor Visa and Master Card.

Name ______________________

Address ______________________

City ______________________

State __________ Zip Code __________

☐ VISA ☐ MASTER CARD ☐ C.O.D.

Credit Card No. ______________________

Expiration Date ______________________

Add $10.00 for shipping, handling and Insurance. Illinois residents please add 6% tax. Add $20.00 for CANADA, PUERTO RICO, HAWAII orders. WE DO NO EXPORT TO OTHER COUNTRIES.

Enclose Cashiers Check, Money Order or Personal Check. Allow 14 days for delivery (21 days for Personal Check orders,) 2 to 7 days for phone orders. Canada orders must be in U.S. dollars.

we are commodore VIC experts!!

PROTECTO
ENTERPRIZES

BOX 550, BARRINGTON, ILLINOIS 60010
Phone 312/382-5244 to order

```
Ø9FØ 8D 89 ØB 2Ø CC FF 6Ø A2
Ø9F8 ØF 2Ø C6 FF A2 ØØ 2Ø E4
ØAØØ FF C9 ØD FØ Ø7 9D AØ ØC

ØA1Ø 2Ø CC FF AD AØ ØC C9 3Ø
ØA18 DØ Ø8 AD A1 ØC C9 3Ø DØ
ØA2Ø Ø1 6Ø A9 ØD 2Ø D2 FF A2
ØA28 ØØ BD AØ ØC 2Ø D2 FF EØ
ØA3Ø ØØ FØ Ø4 E8 4C 29 ØA 68
ØA38 68 AD 89 ØB C9 ØØ FØ Ø7
ØA4Ø A9 ØØ 8D 89 ØB 68 68 6Ø
ØA48 93 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
ØA5Ø 4D 41 43 48 49 4E 45 2Ø
ØA58 4C 41 4E 47 55 41 47 45
ØA6Ø 2Ø 43 4F 4D 5Ø 41 43 54
ØA68 4F 52 ØD ØD ØD 42 59 3A
ØA7Ø 2Ø 44 41 56 49 44 2Ø 45
ØA78 56 41 4E 53 ØD ØD ØD 45
ØA8Ø 4E 54 45 52 2Ø 46 49 4C
ØA88 45 2Ø 4E 41 4D 45 2Ø 57
ØA9Ø 49 54 48 2Ø 54 48 45 2Ø
ØA98 44 52 49 56 45 2Ø 4E 55
ØAAØ 4D 42 45 52 ØD ØD 5Ø 52
ØAA8 45 43 45 44 49 4E 47 2Ø
ØABØ 49 54 2E ØD ØD 45 58 41
ØAB8 4D 5Ø 4C 45 3A 2Ø 2Ø 2Ø
ØACØ 3Ø 3A 46 49 4C 45 4E 41
ØAC8 4D 45 ØD ØD 44 4F 2Ø 54
ØADØ 48 45 2Ø 53 41 4D 45 2Ø
ØAD8 57 49 54 48 2Ø 54 48 45
ØAEØ 2Ø 4F 55 54 5Ø 55 54 2Ø
ØAE8 46 49 4C 45 2Ø 4E 41 4D
ØAFØ 45 2E ØD ØD ØD ØD 49 4E
ØAF8 5Ø 55 54 2Ø 46 49 4C 45
ØBØØ 2Ø 4E 41 4D 45 2Ø 3F 2Ø

ØB1Ø 54 2Ø 46 49 4C 45 2Ø 4E
ØB18 41 4D 45 2Ø 3F 2Ø ØØ 93
ØB2Ø 46 4F 55 4E 44 2Ø 54 41
ØB28 52 47 45 54 2Ø 4C 49 4E
ØB3Ø 45 2Ø 4E 55 4D 42 45 52
ØB38 ØØ 93 54 4F 4F 2Ø 4D 41
ØB4Ø 4E 59 2Ø 4C 49 4E 45 53
ØB48 ØØ 93 43 4F 4D 5Ø 41 43
ØB5Ø 54 49 4E 47 2Ø 4C 49 4E
ØB58 45 2Ø 4E 55 4D 42 45 52
ØB6Ø 2E 2E 2E 2E ØØ 93 12 44
ØB68 4F 4E 45 ØD ØD ØD ØD ØØ
ØB7Ø 2C 5Ø 2C 57 ØØ ØØ ØØ ØØ
```

©

Insight: Atari

Bill Wilkinson
Optimized Systems Software
Cupertino, CA

This month I will first respond to some of that unanswered mail; then part six of Inside Atari BASIC [*a continuing series within this column*] will delve further into string and array magic; and finally we will do a preliminary exploration of the depths of Atari's FMS.

Graphics Revisited

Actually, the title of this section might better be "machine language revisited." Probably none of my columns has generated as much response as part four of my Atari I/O series, subtitled "Graphics," in the February, 1982, issue of **COMPUTE!** Unfortunately, most of the response has been of the "I can't make it work" variety. Of course, my first response is "but I *know* it works!" Yet still the letters ask, "How?"

I do not intend to turn this column into a tutorial on machine language. There are several good books available on 6502 machine language (including the Inmans' book specifically for *The Atari Assembler*), and any struggling beginner who is trying to make do without at least one of them is simply a masochist. However, my ego says that it will be better fed if more readers understand my articles.

For the most part, it seemed that those who had trouble with my February article assumed that what was published was some neat program to be used as is. *Not so!* I had simply given you a set of *subroutines* to use with your own programs. For an example, let us take a simple BASIC routine and its machine language equivalent. First, the BASIC:

```
30000 POKE 20,0 : POKE 19,0
30010 IF PEEK(19)=0 THEN 30010
30020 RETURN
```

Now, you would not mistake that for a complete BASIC program. But, if I told you that entering this routine and then executing a GOSUB 30000 from your program would produce a 4.2667-second pause, you would know when and how to use it. So let's do the same thing in machine language:

```
PAUSE
        LDA #0
        STA 20 ; "poke 20,0"
        STA 19 ; "poke 19,0"
LOOP
        LDA 19
        BEQ LOOP ; "if peek(19)=0 then loop"
        RTS      ; "return"
```

Again, this is *not* a complete program! *But* if you enter it (say at the end of your own machine language program) and then execute a JSR PAUSE, it will produce a 4.2667 second pause. Note, then, that JSR in machine language is the equivalent of GOSUB in BASIC.

The graphics routines (Program 5) in my February article are just subroutines, to be placed in your own machine language program and then JSRed to perform their actions. Perhaps the biggest mistake I made was in presenting these as an assembled listing (complete with "*=$660"). I certainly never used them as such. In point of fact, I tested them by .INCLUDEing them in my test programs, which were written with the OSS EASMD Assembler/Editor. And one of the test programs I used was, indeed, the example given on page 77 of that same article.

So how do *you* get these subroutines in and working for you? First and foremost, you obviously must type in all that code. Perhaps the best thing to do would be to type it in exactly as shown, including even the "*= $660" and the ".END". Then assemble it and carefully compare the object code generated with that in the magazine. When all appears correct, remove the "*=" line and the ".END" line, renumber the whole thing (I would suggest REN 29000,5 or something similar), and LIST it to diskette or cassette. Now use NEW and write your mainline code. When you are reasonably satisfied with it, LIST it to disk or cassette also.

Now what? Obviously, if you have OS/A+, I suggest you use .INCLUDE (an assembler pseudo-op which allows you to include one file while assembling another). In fact, I tend to write assembly code structured as follows:

```
.INCLUDE #D:SYSEQU.ASM
<my mainline code>
.INCLUDE #D:library-routine-number-1
.INCLUDE #D:library-routine-number-2
...
.END
```

If my "mainline" code is big enough, I may even break it into two or three pieces and .INCLUDE each of them separately.

But what if you don't have .INCLUDE capability? Well, several assemblers have "FILE" or "CHAIN," which are not quite as flexible (since you don't return to where you left off after you have assembled a chained-to file...thus making the procedure next to useless for zero page equate files, etc.); but the principle is generally the same: put your mainline code first and then CHAIN to the subroutine files.

And what if you have the Assembler/Editor cartridge? (For all of its faults, it is still a remarkably flexible tool, especially considering that it is usable with cassette-based systems.) Again, the principle holds. The only real difference is that you must do the INCLUDEs yourself. How? Via the ENTER command. If you haven't noticed it up until now, get your manual and read up on the ",M" option of ENTER. You can merge two or more machine language program files (including cassette files) via the ",M" option! Just as you can with BASIC, except that BASIC always presumes you want to merge.

Are there things to watch out for? Of course. Would I ever give you a method without a handful of caveats? (1) If you ENTER/merge a file with line numbers which match some (or all) of those in memory, you will overwrite the in-memory lines. (2) If you EVER forget the ",M" option, you will wipe out everything in memory so far. (3) You won't find out about duplicate labels until you assemble the whole thing.

But even with all these cautions, I *strongly* recommend that you store each of your hard-earned routines on its own file/cassette. It then becomes almost easy to write the next program that needs some of those same routines.

By the way, caution number 1 in the previous paragraph is the reason I suggested RENumbering the graphics routines to 29000, or some such out of the way place. If you make notes of what each file (or cassette) does, as well as what line numbers it occupies, you can build a powerful library. And a P.S.: generally, .INCLUDE, FILE, and CHAIN commands do not require unique line numbers, so you need not worry about RENumbering subroutines for use in such environments.

Gozinta and Gozouta

As long as we are on the subject of machine language techniques, I would like to point out the absolute necessity of establishing entry and exit conventions for each and every subroutine. Again, if you will refer to Program 5 from the February issue, you will note that each routine (GRAPHICS, COLOR, POSITION, PLOT, LOCATE, DRAWTO, and SETCOLOR) specified ENTER and EXIT conditions. For example, GRAPHICS requires that the desired graphics mode number be placed in the A-register before the JSR GRAPHICS. Upon return (RTS), the Y-register is guaranteed to contain a completion status.

On machines with more registers, it is good practice to write subroutines in a way that any registers not specifically designated in the ENTER and EXIT conditions are returned to the caller unchanged. On the 6502 microprocessor, though, it is generally hard to write any significant routine that does not affect all three registers. Therefore, I have adopted the opposite convention for this CPU: If the ENTER/EXIT comments don't say otherwise, I presume that all registers are garbage when the routine returns. What convention you adopt doesn't really matter; just be sure to stick to one, and only one, method and you won't go wrong.

FILL From Machine Language

For those of you who are experienced machine language programmers and have not been kept entertained up to this point, take heart. The other question most asked about my February article was something like "so how do you call FILL from assembler?" I guess my comment that FILL from assembly language was exactly the same as from BASIC didn't make a very good impression. So, okay, I know when I'm licked. Herewith is a FILL subroutine, which I would hope you would include with the rest of the graphics routines and keep in your library for future use.

This time, I won't make the mistake of putting in line numbers and using "*=" and ".END" This is a straight subroutine; type it in and JSR to it *only* after you have satisfied its ENTER conditions.

```
FILL H,V

;  ENTER: Must have previously drawn the right hand edge
;         of the area to be FILLed via JSR's to PLOT and
;         DRAWTO. Just prior to JSR FILL, it must have
;         performed a JSR PLOT to establish the top (or
;         bottom of the line which will define the left edge of
;         the area to be FILLed. FILL presumes that the
;         color to fill with is that which was most recently
;         chosen via JSR COLOR. Finally, on entry, FILL
;         expects the registers to specify the ending position
;         of the line which will define the left edge of the
;         filled area, as follows:
;            h (horizontal) position in X,A registers
;              (X has LSB of position, A has MSB)
;            v (vertical) position in Y register
;
;  EXIT:  Y-register has completion status from OS fill
;         routine
;

FILDAT  =  765   ;where XIO wants the fill color
CFILL   =  18    ;fill is XIO 18
; rest of equates are from February article and program;
;
FILL
        JSR  POSITION    ; subroutine from Feb. 1982
                         article
        LDA  SAVECOLOR   ; value established via JSR
                         COLOR
        STA  FILDAT      ; see BASIC manual: color used
                         for FILL
        LDX  #6*$10      ; file 6...where S: normally is
```

```
LDA  #CFILL      ; the fill command (XIO 18)
STA  ICCOM,X     ; ... is specified
LDA  #0
STA  ICAUX1,X    ; remember, XIO 18, #6, 0
JSR  CIO         ; and let the OS do the work
RTS              ; ...and give us status in the Y-reg
```

By the way, did you notice that we didn't actually specify "S:" for the XIO, as specified in the BASIC manual? That's because the BASIC manual doesn't tell the whole truth. If you perform XIO on an already open file, the operating system ignores any filename you give it! Want to save a little space in your BASIC programs? Use 'XIO 18,#6, 0,0,junk$' where 'junk' is *any* string variable you happen to be using for any other purpose in your program.

Inside Atari BASIC: Part 6

Last month, we delved into the hopefully-no-longer-mysterious details on how string and array space is allocated from Atari BASIC and BASIC A+. We showed how to fool BASIC into believing that a perfectly ordinary string was located smack in the middle of screen space. The advantage of such deceptions is that BASIC can move strings of bytes at extremely high speeds, faster than you could ever hope to accomplish with any BASIC subroutine.

We did not discuss one other significant use of such string moves: Player/Missile Graphics. Obviously, if you can move the screen bytes around, you can move the players around just as well, and just as fast. Again, several games and utilities now available on the market use just this technique.

I also promised in the last column to tell of possible uses for multiple variables in the same address space (that is, having a string and an array occupying the same hunk of memory). If the idea interests you, read on.

One thing which BASICs in general lack is a good means of handling record input/output. How many times have you seen programs doing disk I/O using PRINT# and INPUT#? Yuch. (I have several reasons for that "yuch," but the best one is simply that PRINT#ing an item means that the number of disk bytes occupied depends upon the contents of the item.) But what is the alternative? With many BASICs, there is none. With Atari BASIC there is at least GET# and PUT#, but they are slow. So let us examine a way to make PRINT# and INPUT# work for us, instead of against us.

First, we will examine a small program:

```
100 DIM RECORD$(1),NAME$(20),QUANTITY
   ORDERED(0)

110 OPEN #1,8,0,"D:JUNK"

120 VVTP = PEEK(134) + 256*PEEK(135)

130 POKE VVTP+4,27 : POKE VVTP+6,27

140 GOSUB 900

150 PRINT "GIVE NAME AND QUANTITY:"

160 INPUT NAME$

170 INPUT TEMP : QUANTITYORDERED(0) =
   TEMP

180 PRINT #1;RECORD$

190 CLOSE #1

200 REM --- READ FILE WE JUST CREATED
   ---

210 OPEN #1,4,0,"D:JUNK"

220 GOSUB 900

230 INPUT #1,RECORD$

240 PRINT "WE READ BACK IN:"

250 PRINT ,,NAME$

260 PRINT ,,QUANTITYORDERED(0)

270 CLOSE #1

290 END

900 REM --- CLEAR THE VARIABLES ---

910 NAME$="                    ":REM
   20 BLANKS

920 QUANTITYORDERED(0)=0

930 RETURN
```

Surprised? Even though we cleared the variables in line 220, the input of line 230 re-read them from the file. How? Because line 130 set the dimension and length of RECORD$ to 27, which includes the original single byte of RECORD$, the 20 bytes of NAME$, and the six bytes of the single element of the array QUANTITYORDERED. So PRINT# thought it had to print 27 bytes for RECORD$, and INPUT# allowed RECORD$ to accept up to 27 bytes.

Wow! With one fell swoop we have managed to allow fast disk I/O of any sized record, right? Wrong. Unfortunately, there are several limitations

Royal Software

YOUR MARKETPLACE FOR:

* ATARI

WE DEAL EXCLUSIVELY IN PRODUCTS FOR THE ATARI (THE BEST) COMPUTER

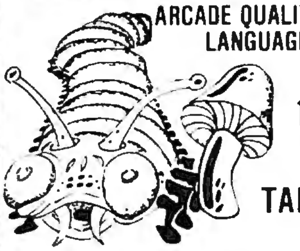
ACTION QUEST

A real time Adventure game played with joystick. You, as the ghost, move from room to room in an attempt to capture valuable prizes while eluding capture by strange creatures. $26.90

16K DISK OR TAPE

MEGALEGS

ARCADE QUALITY MACHINE LANGUAGE GAME



16K DISK OR TAPE $31.40

FAST ACTION . . .
Kill the attacking creatures. It's a spectacular arcade game with lots of excitement! Sure to be one of the TOP games in the months to come.

PRO-FOOTBALL

Play against the computer or another player. Over 25 offense and defence plays. The BEST football game yet.

16K
$26.90 TAPE
$31.40 DISK

HARDWARE

800 Computer 16K	S648
800 Computer 48K	S757
400 Computer 16K	S338
810 Disk Drive	S448
850 Interface	S178
410 Recorder	S 78
825 Printer	S599
16K Ram module	S 69
32K Ram module	S109
Graphics Tablet	S278

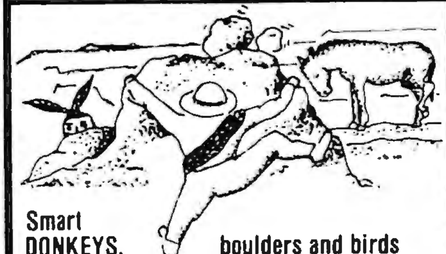
Trade your 400 or older 800 for the new model 800 — Call

SOFTWARE

PAC MAN (cart.)	S39.50
Centipede (cart.)	S39.50
Microsoft Basic (D)	S79.90
Ghost Hunter (T)	S25.50
Ali Baba (D) 32K	S28.90
Protector (T) 32K	S22.45
Apple Panic (D)	S26.90
Threshold (D) 40K	S33.90
GORF (D)	S35.10
Wizard of Wor (D)	S35.10
Text Wizard (D)	S87.90
Dodge Racer (T)	S20.20
Chicken (T)	S20.20
De Re Atari	S17.90
Atari Games Book	S13.90
Personal Finance	S67.50
Zork 1 or II (D)	S35.90
Deluxe Invaders (D)	S31.40
Eastern Front	S25.40
Mousekattack (D)	S31.40
Galactic Chase (T)	S22.40
Bug Attack (T,D)	S26.90
Star Raiders (Cart)	S44.90
Missile Command	S39.50
Midnight Magic (D)	S31.45
Tanktics (T,D)	S24.65
Controller (D)	S27.00

Call or Write for
FREE
ATARI PRODUCTS CATALOG
with nearly 1000 items

Canyon Climber



Smart DONKEYS, boulders and birds dropping rocks try to stop you from reaching the top. A real challenge. Arcade quality.

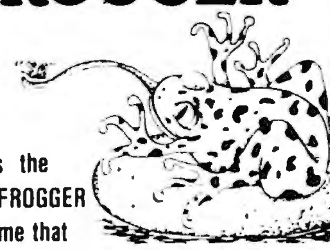
16K
DISK $26.90 TAPE $26.90

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.

32K $26.90
DISK or TAPE

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 800-235-5759

FOR INFORMATION CALL 805-967-3889

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. CA residents add 6% sales tax. 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, 5138 Hollister Avenue, Santa Barbara, CA 93111

* Atari is Trademark of Atari Inc

to this technique. (1) The record cannot be over 255 bytes long or INPUT# won't be able to retrieve it all. And any size over 127 bytes will wipe out routines/data in the lower half of page $600 memory. (2) The record cannot contain a RETURN (155 decimal, 9B hex) character. It will print fine, but the INPUT# will terminate on the first RETURN it sees. (3) The other strings in the record (NAME$ in our example) will *not* have their lengths set properly by the INPUT#, thus necessitating something like the routine at line 900. But if you insert "280 PRINT LEN(NAME$)", you will always get a result of 20.

Well, limitations one and three are easy enough to predict and understand, but how do you insure that your data does not contain a RETURN code? For strings which have been INPUT by a user, that's easy: the RETURN code will never appear in such a string. But what about numbers? Remember that we will be printing the internal form of Atari decimal floating point numbers. Can such numbers contain a byte with a value of 155 ($9B)? Yes, but such a number would be in the range of -1E-74 to -9.E-73, which is unlikely enough to ignore for most purposes.

So, in summary, is this make-a-record technique useful? I'm not sure. Certainly BGET/BPUT or RGET/RPUT from BASIC A+ or their USR equivalents under Atari BASIC are much easier to code and use. And, yet, there is a certain elegance to record-oriented techniques which is not entirely lost to me. I probably will stick with the constructs we invented for BASIC A+, but I would respect a program using the above techniques.

A few last commments: the pokes of line 130 depend on RECORD$ being the first variable defined. Recall my comments from last month about LISTing and reENTERing a program to insure a particular order of definition. Also, if you need to alter a variable other than variable number zero, remember that the formulas are:

VVTP+8 * VNUM+4 for the LSB of the length
VVTP+8 * VNUM+6 for the LSB of the DIMension

(and, again, see last month's article for fuller explanations).

And, finally, I really would be interested in hearing from anyone who uses the techniques I have devised here to produce a unique, real-world program that does things that can't be done otherwise.

Fun With FMS, Canto The First

Remember that fix for burst I/O I gave you in the May, 1982, issue? Did you try it?. Did it prevent burst I/O errors? Yep. Did it slow down every kind of disk read? Yep. Oooooopsy daisy. Well, you can't be completely right all the time. This month, we will try again.

First, I would like to explain, in terms of the FMS listing and the commentary (Chapter 12 – BURST I/O, *Inside Atari DOS*, **COMPUTE! Books**) why the fix I gave you in the May, 1982, issue worked insofar as it fixed the burst I/O problems.

To begin with, examine the code at locations $09F8-$09FD and $0AD2-$0AD7. These are the locations in PUT-BYTE and GET-BYTE, respectively, where the burst I/O routine is called. But lo! In PUT-BYTE, the JSR to burst I/O is directly preceded by a BCS, meaning that burst I/O won't occur unless carry is clear. But, in GET-BYTE, the JSR to burst I/O is directly preceded by a BCC – burst I/O occurs in read mode only if carry is set!

Now, if you examine the label "WTBUR" at $0A1F, you will note that the first thing that occurs is a test of FCBFLG to find out if we are in update mode or not. If we are updating, we don't burst. But note that GET-BYTE called the label "RTBUR", AFTER the test, and so would always burst, whether in update mode or not. What I tried to do was change the "JSR RTBUR" (at $0AD4) to a "JSR WTBUR" and then use the carry flag to distinguish between the type of request (I changed the BMI at $0A24 to a BCC). *Great!* It worked! Except...it worked *too* well. Unfortunately, FCBFLG is zero (and therefore plus) when we have a file open for read only; so, therefore, the burst I/O was suppressed for *all* reads. Nuts.

We try again, using a slightly different approach. We will still count on the carry being set when called from PUT-BYTE and reset when called from GET-BYTE. This time, though, we will examine the actual I/O mode in use. FMS receives the I/O mode from CIO when the file is opened and places it in FCBOTC. Recall that the only legal values are 4, 6, 8, 9, and 12. Well, burst I/O is only illegal in modes 6 (read directory) and 12 (update). But mode 6 is handled separately (see $0AC5-$0ACB), so 12 is all we are really concerned with. Anyway, without further ado, here's the listing of the FMS patch:

```
          *=$0A1F
;
; first, patch the code where WTBUR used to be
;
WTBUR
BURSTIO
          LDA     FCBOTC,X  ; Open Type Code byte
          EOR     #$0C      ; check for mode 12...
                              update
          BEQ     NOBURST   ; it IS update...don't burst
          ROR     A         ; move carry to MSB of A
                              register
          NOP               ; filler only
TBURST
; ... and the STA BURTYP remains ... but now BURTYP is
```

```
;    negative if BURSTIO was called from GET-BYTE and
;    positive if it was called from PUT-BYTE.
;
            *=$0A41
; so we must patch here to account for the sense of being
; inverted from the original.
            BPL        WRBUR      ; called from PUT-BYTE
            *=$0AD4
; finally, we must patch the GET-BYTE call so that it no
; longer JSR's to RTBUR.
            JSR        BURSTIO    ; call the common burst
                                    routine
;
            .END
```

And for those of you who don't want to type all that in, you might simply use BUG to do the following changes:

```
C A20 < 82,13,49,0C,F0,24,6A,EA
C A41 < 10
C AD5 < 1F
```

And, last but not least, from BASIC you may use the following:

```
POKE 2592,130
POKE 2593,19
POKE 2594,73
POKE 2595,12
POKE 2596,240
POKE 2597,36
POKE 2598,106
POKE 2599,234
POKE 2625,16
POKE 2773,13
```

Fun With FMS, Canto The Second

Not long ago, an OSS customer told me that he couldn't use Atari DOS to SAVE (option K on the menu) the contents of ROM. "How sneaky," cried I, "Best to use the SAVE command under OS/A+. We wouldn't do anything that nasty to you!"

But we did. And we do. And it isn't because we or Atari are sneaky or nasty; it is yet another phenomenon of burst I/O. Recall that when the burst I/O test is passed, FMS calls SIO to transfer the sectors of data directly from the user's buffer space. In order to do so, though, it must write the sector link information (last three physical bytes in a sector) into the correct spot in the user's buffer before calling SIO. Then, when SIO returns, it restores those three bytes and tries to write the next sector the same way. Again, if you have *Inside Atari DOS*, you can follow this happening at addresses $0A52-$0A7A, in the "WRBUR" code.

Ah...but what happens when you try to do burst I/O writes from ROM? FMS blindly tries to put its goodies into those three bytes and call SIO. SIO does what it is told, and FMS thinks that all is OK. Except that all is *not* OK! Those three bytes did *not* get changed, so what was written to the disk is garbage. And even ERAsing the file won't work, because the sector links are badly messed up. Crunchy, crunchy goes the disk, under worst-case circumstances.

Now this restriction is fairly easy to get around: one simply writes a program (in BASIC or machine language) which writes the desired bytes to the disk one at a time, thus preventing burst I/O. So I don't feel that I am giving away deep, dark Atari secrets when I give you an easier method to prevent burst I/O. Simply do either of the following:

```
from BUG:      C A2E < 0
from BASIC:    POKE 2606,0
```

Again, for those of you with the FMS listing, note that what we are doing is changing the AND #$02 which checks for text mode (the read and write text line commands are $05 and $09, neither of which have bit $02 turned on) into an AND #$00 instruction, thus fooling the BEQ that follows into thinking that FMS can't do burst I/O because it's doing text mode I/O. Not too terribly tricky, and it works well.

I cannot recommend that you make this patch a permanent part of most system disks, since it completely disables burst I/O and makes the system load and save files considerably slower. Change it, use it, and then forget it. ©

After you've written a FORTH program, you can metacompile, leaving out words which are not a working part of the application.

The FORTH Page

Headless Metacompilation

Richard Mansfield
Assistant Editor

Somehow, the prefix *meta* always sounds both glamorous and forbidding. Attached to a word, *meta* usually adds a special meaning involving "over" or "beyond" or "transformed" – it can seem paradoxical. A *metatruck* would be a truck that carries other trucks. MetaFORTH, likewise, means using FORTH to compile itself.

"FORTH for PET," an adaptation of FORTH by L. C. Cargile and Michael Riley, supports a FORTH Metacompiler (for Upgrade or 4.0 BASIC with 32K RAM) which can be purchased separately for $30. "FORTH for PET" itself does not require the extra memory.

The goal of metacompilation is to reduce memory size or modify FORTH itself by recompiling the entire FORTH dictionary. Obviously, you cannot eliminate words like DUP and SWAP. They are in the definitions of many other words and will always be central in any working FORTH application.

However, there are applications which will not need keyboard entry, disk access, or other such normal FORTH capabilities. If you write a game which does not involve the disk, you could metacompile a special FORTH which does not have such words as FLUSH, LOAD, etc.

Clear, Step-by-step Instructions

The Metacompiler's instruction booklet is written so clearly that a relative novice in FORTH will have little difficulty with it. The process of metacompilation is essentially straightforward and the manual leads you through it, step-by-step, explaining what's happening along the way. You load your ordinary FORTH into the computer, load screen ten from the Metacompiler disk, and it requests a target address for a new, temporary FORTH. Since regular FORTH starts at hex $0400, you can respond with $4000 as the start of the new version. Both will be resident in memory simultaneously, but after metacompilation, control of the computer is passed to the upper one.

After you've indicated your target address, you can sit back and watch the dictionary being recreated at the new location. The words are put on screen as they are recompiled. Or you can leave and come back about six minutes later to find two FORTHs in the computer. You'll be able to communicate with the upper version and it will respond normally.

The final step is to compile back down, giving $0400 as the target this time, and making any changes you want to the Metacompiler's disk screens. These changes will be reflected in the new FORTH you are creating.

The manual takes you carefully through an entire editing process where you create a stand-alone "Calendar" program. This calendar application is truly a FORTH transfigured: when run, it uses DUP and other key words, of course, but it will not recognize them from the keyboard. It can do nothing with the disk. In fact, it understands only the few words that relate to the calendar functions to which it is dedicated. What's more, it's *headless.*

Removing Heads

In FORTH, there are four "fields" to an ordinary word: the name, link, code, and parameter fields. The "head" of a word is the name and link fields, taken together. If a word in your new FORTH application will only be used in further definitions and will never be independently executed, you can type DROP-HEADS before it on the Metacompiler disk. That word and any words after it will be compiled without name or link fields when the new FORTH is created. To resume head compilation, type COMPILE-HEADS.

An Excellent FORTH Adaptation

It is gratifying to find this metacompilation capability, a valuable and sophisticated technique, available on PET/CBM computers. Equally pleasing is the excellent FORTH implementation by the same authors, "FORTH for PET" ($50, 16 or 32K, Upgrade or 4.0 BASIC, with disk) to which this Metacompiler is an optional addition.

"FORTH for PET" is a full FIG FORTH system (allowing all FORTH 79 extensions as an option), a complete assembler, several sample programs, many utilities, and introductory and reference manuals. Perhaps most attractive is the fact that the full, powerful Commodore screen editing system works the same way that it works in BASIC. You move anywhere fast and modify screens with great ease. Other versions of FORTH are sometimes encumbered by awkward editing procedures.

"FORTH for PET," and "Metacompiler for FORTH" are both available from A B Computers, 525 Bethlehem Pike, Colmar, PA, 18915. ©

If you've ever wanted to thoroughly document, explore, and understand your computer's BASIC or Operating Systems – the techniques and programs here are your tools. Written for OSI, these ideas can be modified for other computers. Next month this article concludes with additional programs and examples.

Resource Part I:

Mapping Machine Language Code

T. R. Berger
Coon Rapids, MN

Have you ever tried to document your machine software by annotating disassemblies? Have you ever tried to move these programs by reconstructing assembler source listings from disassemblies? If so, you know what a huge investment of time is needed. This article covers a group of BASIC programs which will facilitate regenerating fully documented assembler source listings starting from machine language programs in much less time than the painful direct route.

When I undertook to write these programs, I did not even dream how powerful they would be. I never really anticipated regenerating a source listing of 8K OSI Microsoft Disk BASIC. When I realized that this task could be done, what was one simple program expanded into the four presented here. A much modified and improved version of the single program which started this all off is also included here. If OS65D would allow six buffers to be open at once, these programs could be vastly speeded up and simplified.

These programs are written in BASIC for disk based OSI computers. However, the programs are carefully documented so that those using other 6502 machines with different disassemblers should have no difficulty in copying the idea. The programs accept as input an ASCII file produced by the OSI version of the Apple disassembler (see *Dr. Dobbs Journal,* September, 1976, p. 22). The output is a collection of ASCII files which include the following:

1. An assembly source listing of code which will reassemble at the same location without further editing.
2. Equate files necessary to run the assembly source through an assembler.
3. Separate cross reference files for each of the following:
   a. Zpage addresses,
   b. Jumps and jumps to subroutines,
   c. Memory calls, and
   d. Branches.

A single pass program RESOURCE S in included for resourcing small programs. On a 48K C8PDF it has no trouble handling OS65D. Since only symbols and cross references are kept in memory, a 32K machine should also have no trouble. Cross reference strings in RESOURCE S are of limited size so that the program will crash in attempting to cover 8K BASIC. The Zpage cross references to $AC overrun about halfway through. Since RESOURCE S is a compressed version of the program package presented, I will comment very little on it. When there are a very large number of cross reference strings, the program slows way down due to garbage collection. In Microsoft BASIC, garbage collection times go up approximately as the square of the number of strings in memory (and not their size).

Run Times Approached 24 Hours

I have written this package so that hobbyists can understand their most commonly used language: BASIC. A source file for 8K BASIC is colossal. Therefore, many shortcuts are necessary to complete the resourcing task. I originally tried to enlarge RESOURCE S to cope with the job. OS65D has only two disk buffers requiring that a large amount of information be kept in memory for a single program. So many strings were generated and garbage collection time became so great that run times approached 24 hours. Clearly this is not the way to go. I broke the task into small pieces, each being completed in a reasonable amount of time.

On 8" floppies the BASIC disassembler source ($03A1 - $2300) takes 28 tracks (84K). Those using minifloppies must tackle BASIC in three or more passes, using the cross reference tables to properly join the final product.

The final product, scattered through several files, takes up about 36 tracks. There is no hope of assembling these files without a linking assembler. (Leroy Erickson has written such an extension for the OSI Assembler.) However, printout of the source and the cross reference tables greatly simplifies the annotation and documentation process. After one pass of RESOURCE S over OS65D, it was possible to reassemble OS65D at the same location. After about two hours of editing, a file was obtained which assembles anywhere.

Using my maps of OS65D and Jim Butterfield's maps of BASIC, you should be able to obtain fully documented source listings of both BASIC and OS65D. I would hope to see more articles using specific parts of OS65D and BASIC. Namely, what are some subroutines, how do they work, how does one use them, and how does one resource them?

The entire program package presented here is written in BASIC. This sped implementation and modification time. It also makes the programs easier to understand. The price paid is runtime, which is considerable over 8K BASIC. Efforts have been made to optimize runtimes, especially on inner loops. This adds steps to the process, but significantly reduces program running times.

Of course, one must edit the files generated by these programs. I use a group of utilities which constitute a useful BASIC text file editor and processor. I will describe these utilities in a future article.

The three most useful utilities are a transfer program to move large text files around, a print program to output large files to a printer, and a fast sorter to sort symbol tables. A further useful addition is a large text file single pass character-oriented line editor.

How It Works

The first program (PASS 1) takes the disassembly listing (which I will call SOURCE) and compresses it into a scratch file (which I call SCRATCH). The main working file is SCRATCH. It is about 25% smaller than SOURCE and serves as input to the other programs. A typical line of SOURCE looks as follows:

```
1A3D BD11B0 LDA $B011,X.
```

In SCRATCH this same line would be:

```
1A3D LDA HHB011,X.
```

The code field has been eliminated and $B011 has been changed to a six letter symbol. All four digit operands $XXXX are changed to six letter symbols HHXXXX, which is the maximum size for symbols in OSI's Assembler. Except for immediate operands, two digit operands $YY are replaced by six letter symbols HHZZYY. Further, the first H in every operand is always aligned as the eleventh letter in a line. BASIC is much too slow to search a line for a symbol. Aligning symbols makes them easy to find when editing. For example,

```
MID$(IN$,11,6)
```

removes a symbol from a line IN$. The 'H' in position eleven distinguishes a symbol. The 'Z' in position thirteen distinguishes a Zpage reference.

A line in SOURCE

```
1A40 FF ???
```

would appear in SCRATCH as

```
1A40 .BYTE $FF.
```

This step makes the resource file assembler ready. Bad disassembly of opcodes must be fixed by editing the final file if a true source file is needed. In particular, tables and text are not resourced correctly, only made assembler-ready.

The first program also builds a table of two byte operands (which I will call SYMBOL). SYMBOL is used in PASS 2 to generate labels and an equate file of two byte operands. Since SYMBOL is searched repeatedly in PASS 2, it must be sorted. Sorting SYMBOL means a fast binary search can be used which is many times faster than a sequential search. (For BASIC, this addition reduced line process times in PASS 2 from about 5 seconds per line to less than 1 second per line.) Since BASIC requires 800 symbols, this search method cuts hours off PASS 2. Accordingly, PASS 1 keeps a sorted symbol table.

PASS 2 generates the resource file (which I call OBJECT). It reads one line of SCRATCH:

```
1A3D LDA HHB011,X.
```

It searches SYMBOL for 1A3D. If 1A3D is found, a numbered line

```
10000 HH1A3D LDA HHB011,X
```

is output to OBJECT. Since 1A3D is now defined by a label, it is marked as 'used' in SYMBOL. If 1A3D is not found, a numbered line

```
10000        LDA HHB011,X
```

is output to OBJECT. After OBJECT is complete, the unmarked symbols in SYMBOL are operands which are not defined by labels in OBJECT. Thus, an equate file (which I call EQUATE) is written using these unmarked terms from SYMBOL. For example, if 1A3D is unmarked, it would be written to EQUATE as a numbered line

```
5000 HH1A3D = $1A3D.
```

Except for Zpage labels, OBJECT and EQUATE are ready for the assembler.

PASS 3 generates the various symbol tables. The symbols are picked out of SCRATCH along with their addresses. A symbol HHXXXX is stored in a string SS$(I) as XXXX. A check is run to see if the symbol already appears in the table. If it does not, the counter SN is incremented and the symbol is added. This list is stored as a sorted table.

Suppose that HHXXXX appears in Line YYYY and that SS$(I) = XXXX. Then UYYYY is appended to the right hand end of SA$(I) where U is chosen to give information about the opcode on line YYYY. Some thought went into the choice of U. In the branch table, the middle letter of a branch instruction comes closest to distinguishing all branches. Thus U is the middle letter of the opcode. Again in the JMP and JSR table, the middle letter distinguishes JMP from JSR. Thus U is M or S in this case. The first letter of the opcode is chosen for the memory table.

In decoding programs, I have found that the most important fact to know about Zpage opcodes is their addressing mode. That is, is an opcode indexed or not? Thus, U is the extreme right hand symbol of the disassembly line. This includes), X, and Y. It is not possible from this to tell whether the Y means indexed or indirect indexed. However, given the simplicity of this approach, it is adequate.

Example 1.

```
10000          .BYTE $17
10010          LDA #$16
10020          STA  HHZZC7
10030 HH18DD JSR  HH18ED
10040          JSR  HH19D1
10050          STA  HHZZC5
10060          STY  HHZZC6
10070          DEC  HHZZC7
10080          BMI  HH1922
10090          BNE  HH18DD
10100 HH18ED JSR  HH19BC
10110          LDA  (HHZZC5,X)
10120          TAY
10130          LSR A
10140          BCC  HH1901
10150          LSR A
10160          BCS  HH1910
10170          CMP #$22
10180          BEQ  HH1910
10190          AND #$07
10200          ORA #$80
10210 HH1901 LSR A
10220          TAX
10230          LDA  HH17A5,X
10240          BCS  HH190C
10250          LSR A
10260          LSR A
10270          LSR A
10280          LSR A
10290 HH190C AND #$0F
10300          BNE  HH1914
10310 HH1910 LDY #$80
10320          LDA #$00
10330 HH1914 TAX
10340          LDA  HH17E9,X
10350          STA  HHZZC1
10360          AND #$03
10370          STA  HHZZC2
10380          LDA  HHZZC8
10390          BNE  HH1923
10400 HH1922 RTS
10410 HH1923 TYA
10420          AND #$8F
10430          TAX
10440          TYA
10450          LDY #$03
10460          CPX #$8A
10470          BEQ  HH1939
10480          LSR A
10490          BCC  HH1939
10500          LSR A
10510          LSR A
```

If SA$(I) becomes too long, it is written to a cross reference file and SA$(I) is emptied. (In RESOURCE S this step is not performed, the program bombs when SA$(I) becomes too long.) These "long strings" will appear out of order in the file. (The first few cross references may be out of order.) The symbol table can be resorted by most any sorting program. As it stands, the table is "almost in order."

PASS 4 generates the Zpage equate file which I call ZEQUATE. This is done using the Zpage cross reference file generated in PASS 3. The file resembles the EQUATE file.

In resourcing a large program, there will not be enough room on one disk for all the files generated. SCRATCH, and various other files may be moved using a transfer utility. Symbol and cross reference files may be sorted using a sort utility. Final files may be printed using an output utility.

Example 1 shows the OBJECT file (resourced assembly language) for the beginning of the disassembler in the Extended Monitor. Example 2 gives the two equate files. Example 3 gives the output from the Assembler using these three files. Example 4 gives the four cross reference tables. The first address in each row is the symbol. The other addresses following are the cross references, with some indication as to opcode.

How To Use It

STEP 1) Creating a SOURCE file.

If you plan to resource BASIC, you must move the Extended Monitor since it overlays part of BASIC. In another article, I will give explicit instructions on how to do this. I find it handy to have the Extended Monitor available while BASIC is resident.

After trying several methods, I've decided that the following is the easiest way to generate a SOURCE file. It uses the disk output capability of OS65D. The code you are resourcing should not overlay the disk buffer used. (Video with polled keyboard is assumed; otherwise, recheck the I/O flags.)

a) Initialize a fresh disk.

b) Copy the directory Track D onto this disk using OS65D's copy utility (D is Track 8 on 8" floppies).

c) Create files for all empty tracks except Tracks 0 and D. Delete all directory entries on the new disk.

d) Load the machine language program to be resourced.

e) Load and run the Extended Monitor.

We must now set all the various pointers for a disk buffer. To resource BASIC you need a very large file. Let the first available track be N where the directory is on Track N-6 (N=9 on 8" floppies).

f) Choose a first track number N for your SOURCE file. Let M be the last track number on the disk (M=76 on 8" floppies). Do not choose N so that either N=0 or the directory track is included in the range of N to M.

g) Using the "at" (@) sign command, set the following buffer values. (These are valid for OS65D V3.2, i.e. 8" floppies. The correct values for minifloppies are given in the *OS65D User's Guide.*)

ADDRESS ($)	ADDRESS (D)	VALUE	
2326	8998	7E	BUFFER START ADDRESS
2327	8999	31	
2328	9000	7E	BUFFER END ADDRESS
2329	9001	3D	
232A	9002	N	FIRST TRACK OF FILE
232B	9003	M	LAST TRACK OF FILE
232C	9004	N	CURRENT BUFFER TRACK
232D	9005	0	DIRTY BUFFER FLAG
23C3	9155	7E	ADDRESS DISK OUTPUT
23C4	9156	31	(CURRENT BUFFER ADDR.)

h) Mount the fresh disk.

I) From EM type (i.e. turn on disk output)

!IO ,22 <return>.

The next few steps write directly to the disk without error correction. If you make an error, perform step l) and restart at step g. Presumably you know the start ($XXXX) annnd the finish ($YYYY) addresses of the code to be resourced. For BASIC these are $XXXX=$03A1 and $YYYY=$2300. For OS65D these are $XXXX=$2336 and $YYYY =$2E1E. For the ROMs these are $XXXX=$FD00 and $YYYY=$FFFA.

j) Commence disassembly with

QXXXX <return>.

k) Put your finger on LINEFEED. Hold it there until $YYYY has been disassembled. Then hit RETURN.

All but the last track of the SOURCE file is on the disk. The last track is still in the buffer. This last part is also missing an "end of file" marking. The next few steps turn off the disk output long enough to make corrections, then turn it back on to write the final track to the disk.

The next step turns off disk output by creating a syntax error, and puts a mark to help find the end of the SOURCE file.

l) Type

!XIT <return>.

m) Search for the end of the file

W!XIT>317E,3D7F.

If all has gone well, you will receive a message

(*) VVVV/21

where VVVV is the address of ! in the expression !XIT. If you do not receive such a message, it is possible (but unlikely) that a "disk write" occurred in the middle of the word !XIT. Go back to step g) and start again. When you reach step k), hit RETURN five times instead of just once, then proceed. If you do not receive the message (*), something is definitely wrong somewhere. Start a careful search (*Beware:* some values given only work for 8" floppies).

n) Using quotes and "at" check the following

```
VVVV/21   "!
VVVV+1/58 "X
VVVV+2/49 "I
VVVV+3/54 "T
```

o) Make the following change.

VVVV/21 0D

Now the "end of file" marker is properly installed. Next we write the buffer to the disk.

p) Make the following pointer change.

```
23C3/YY 7E
23C4/WW 3D
```

q) Write down the value TK

232C/TK

r) Type

!IO ,22 <Return>

The entire SOURCE file is on the disk. It starts on Track N and ends on Track TK. We must now create a directory entry for this file.

s) Load BASIC and CREATE, but do not run CREATE. (You may need a different disk to do this.)

t) Delete line 20290 (which would erase all the work you have done). It reads:

20290 DISK!"IN "+T$:DISK!"SA "+T$+",1=317/"+P$

u) Run CREATE and name the SOURCE file on your new disk.

STEP 2) Create a SCRATCH and SYMBOL file entry.

These files must be on the same disk as SOURCE. Be sure to reload CREATE so that line 20290 is not missing. SCRATCH can be about 25% smaller than SOURCE. SYMBOL can be 1-2 tracks. On 8" floppies, 8K BASIC needs two tracks for

SYMBOL (4K file size).You may put these files anywhere as long as they do not overlap the directory track, Track 0, or the tracks used by SOURCE.

STEP 3) PASS 1.

Run the first resource program. Prompting will tell you what to do. The new disk must be in the drive throughout the run. The screen will display the current status. On large programs, be prepared for several-minute waits for garbage collection. A five minute wait between screen data lines probably means there has been a system crash. This program will not work with ROM BASIC since the garbage collector is defunct. (See *PEEK(65)*, March 1980, p. 3 for a fix.)

The SOURCE, SYMBOL, and SCRATCH file may fill a disk, so you may have to move some files to other disks. SOURCE is no longer needed, but should be saved in case of trouble. Symbol is needed for PASS two and SCRATCH is needed for PASSes two and three. Using a transfer utility you may move SCRATCH and SYMBOL to a new disk.

STEP 4) PASS 2.

The second resource program generates an EQUATE file and the resourced assembly listing OBJECT. Create such files on a disk containing SCRATCH and SYMBOL. EQUATE need not be large, usually much less than a track. OBJECT should be slightly larger than SOURCE.

The next step creates all the cross reference tables. Each table needs its own file. SCRATCH is the input file. The branch table will probably be the largest file.

STEP 5) PASS 3.

Repeat this step until all cross reference tables are complete. Only Zpage cross references are essential. However, I find the Zpage and JSR tables the most useful. You may wish to sort these tables, even though they are "almost sorted."

STEP 6) PASS 4.

Create the Zpage equate file: ZEQUATE. Input to this program is the Zpage cross reference file. This step is the final one which creates the list of Assembler Zpage equates.

Any of the files generated may be dumped to a printer using a printer utility. The process is much simpler than it sounds. The single pass resource program eliminates most steps if only small programs are being resourced.

Moving ASCII Text Files To The Assembler

For small programs, the resource can actually be assembled by the OSI Assembler. The three files (OBJECT, EQUATE, and ZEQUATE) must be merged and the program counter location given (10 * = $XXXX).

The resourced files are ASCII text files with an end of file (EOF) marker:

XIT <return>.

Since OSI's Assembler does not keep an ASCII file, more is needed. We must transfer the disk text files into the Assembler/Editor. In OS65D it is easy to reset output flags with:

IO ,02.

However, only one input is recognized and, if this is not the keyboard, then keyboard input is dead. During disk input, the keyboard is disabled. In particular, OS65D has no way of recognizing the end of a file except by an operating system error. This is a definite deficiency in OS65D. When an operating system error does occur, the IO flags are properly reset to default values.

If a file is on Tracks 2 and 3, inputing these tracks will result in a system error as soon as Track 3 is finished. The trouble is that the actual file may end halfway through Track 3. The rest of Track 3 may contain absolutely destructive information, such as Assembler commands or operating system commands. My favorite is the following. The ASCII character "left bracket" occurs as input opening the Indirect File. This file fills up memory, wiping out everything in the way. It eventually reaches the disk addresses. You hear a thunk and the disk goes dead. If input continues, it next reaches the screen memory filling the screen with jazzy characters. It goes on to the color memory, tone generator, etc. You've probably had this occur and wondered what happened. It's just the Indirect File, filing all the garbage away.

One solution is to remove the destructive information on the track. Another simpler one is to create an operating system error at the end of the file, in this case, midway through Track 3. Input errors to the OSI Assembler do not cause the IO flags to be reset. We must be more subtle than just having an input error. If E<return> is sent to the Assembler, it exits to the operating system. In the operating system command mode, any line which is not a legal command creates a syntax error. For example, another E<return>, will do the job. The following changes to PASS two and PASS four will prepare files for entry into the Assembler. Add the following lines:

PASS Two

```
642 PRINT #7,"E"
644 PRINT #7,"E"
842 PRINT #7,"E"
844 PRINT #7,"E"
```

PASS 4

```
472 PRINT #7,"E"
474 PRINT #7,"E"
```

There is yet another problem. In their normal positions, the disk buffers occupy the same space as program memory. This problem can be solved by moving the buffers. Use the following steps to load first the file ZEQUATE, second EQUATE, and third OBJECT into the Assembler.

a) Load and run the Extended Monitor.

b) Suppose the file we wish to load starts on Track N and ends on Track M. Perform STEP l) g) from "HOW TO USE IT." Be sure to use the values given below (or larger values where you have RAM).

ADDRESS ($)	ADDRESS (D)	VALUE	
2326	8998	00	
2327	8999	50	
2328	9000	00	
2329	9001	5C	
232A	9002	N	
232B	9003	M	
232C	9004	N-1	
232D	9005	FF	
23AC	9132	00	ADDRESS MEMORY
23AD	9133	5C	BUFFERED INPUT

Note that 232C, 232D, 23AC, and 23AD have strange values. These values track the disk into loading the first track of your file into memory. Otherwise you would have to do that job separately.

c) If you have already loaded the first file, skip this step. Initialize the Assembler.

d) Re-enter the Assembler.

e) Get input by

!IO 20 <return>.

f) Repeat a) - e) until all files are loaded.

Your files are now merged in the Assembler. Be sure to inspect them carefully before assembling.

Remarks, Refinements, Additions

Resource will execute on 8K BASIC in a reasonable amount of time. The longest pass (PASS One) will run slightly less than an hour on a 1 MHZ machine.

This package of programs is, in a sense, incomplete. Using the cross reference tables, one could give mnemonic names to all of the various labels and equates. These could be entered into a file. Then one extra pass over OBJECT could exchange address labels with mnemonic labels.

A big file line editor utility could be added to edit any one of the files created. If tables are known at disassembly time, they can be edited into SCRATCH. Incorrectly disassembled code could be corrected. These steps could be performed also on SCRATCH or OBJECT.

If table locations are known in advance, disassembly can be cleaned up considerably by replacing all table bytes with $FF (or any other value not equal to a 6502 opcode). Then, all tables will appear in the resource as a sequence of lines: 10000 .BYTE $FF. Using an editor, it then would be a simple task to replace each $FF by its correct value. I used this procedure on BASIC.

OS65D cannot be changed in this way since it will crash. But there is a simple solution. Move OS65D from addresses 2XXX to addresses, say 5XXX. When SCRATCH and SYMBOL are complete, go through them, changing the leading 5's back to 2's. A program to do this is simple to write. SYMBOL must be resorted and repetitions deleted. This way, I was able to use the trick with $FF in tables to obtain an accurate resource of the code in OS65D.

A simple, but useful, utility would be a commenter. Such a utility would allow the user to add comments to the end of each line of the resourced file or to insert lines into the file. I have used this technique to produce the various listings in this article. I hope to present a future article on this editor.

Even though I am careful to fully document the machine software I write, I still find it useful to run the resource program over my machine programs. The cross reference files often reveal infelicities and logical inaccuracies.

I am still improving these programs. If you think of a nice enhancement, I'd be glad to hear about it.

Example 2.

```
1000  ;EQUATE FILE
1010 ;
1020  ;ZPAGE
1030 ;
1040  HHZZC1 = $C1
1050  HHZZC2 = $C2
1060  HHZZC5 = $C5
1070  HHZZC6 = $C6
1080  HHZZC7 = $C7
1090  HHZZC8 = $C8
1100 ;
1110 ;
1120  ;TWO BYTE
1130 ;
1140  HH17A5 = $17A5
1150  HH17E9 = $17E9
1160  HH1939 = $1939
1170  HH19BC = $19BC
1180  HH19D1 = $19D1
```

Example 3.

```
10 18D8                  * = $18D8
1000                   ;EQUATE FILE
1010                  ;
1020                   ;ZPAGE
1030                  ;
1040 00C1=              HHZZC1 = $C1
1050 00C2=              HHZZC2 = $C2
1060 00C5=              HHZZC5 = $C5
1070 00C6=              HHZZC6 = $C6
1080 00C7=              HHZZC7 = $C7
1090 00C8=              HHZZC8 = $C8
```

```
1100                   ;
1110                   ;
1120                   ;TWO BYTE
1130                   ;
1140 17A5=             HH17A5 = $17A5
1150 17E9=             HH17E9 = $17E9
1160 1939=             HH1939 = $1939
1170 19BC=             HH19BC = $19BC
1180 19D1=             HH19D1 = $19D1
10000 18D8 17                 .BYTE $17
10010 18D9 A916               LDA #$16
10020 18DB 85C7               STA  HHZZC7
10030 18DD 20ED18  HH18DD JSR  HH18ED
10040 18E0 20D119             JSR  HH19D1
10050 18E3 85C5               STA  HHZZC5
10060 18E5 84C6               STY  HHZZC6
10070 18E7 C6C7               DEC  HHZZC7
10080 18E9 3037               BMI  HH1922
10090 18EB D0F0               BNE  HH18DD
10100 18ED 20BC19  HH18ED JSR  HH19BC
10110 18F0 A1C5               LDA (HHZZC5,X)
10120 18F2 A8                 TAY
10130 18F3 4A                 LSR A
10140 18F4 900B               BCC  HH1901
10150 18F6 4A                 LSR A
10160 18F7 B017               BCS  HH1910
10170 18F9 C922               CMP #$22
10180 18FB F013               BEQ  HH1910
10190 18FD 2907               AND #$07
10200 18FF 0980               ORA #$80
10210 1901 4A      HH1901 LSR A
10220 1902 AA                 TAX
10230 1903 BDA517             LDA  HH17A5,X
10240 1906 B004               BCS  HH190C
10250 1908 4A                 LSR A
10260 1909 4A                 LSR A
10270 190A 4A                 LSR A
10280 190B 4A                 LSR A
10290 190C 290F    HH190C AND #$0F
10300 190E D004               BNE  HH1914
10310 1910 A080    HH1910 LDY #$80
10320 1912 A900               LDA #$00
10330 1914 AA      HH1914 TAX
10340 1915 BDE917             LDA  HH17E9,X
10350 1918 85C1               STA  HHZZC1
10360 191A 2903               AND #$03
10370 191C 85C2               STA  HHZZC2
10380 191E A5C8               LDA  HHZZC8
10390 1920 D001               BNE  HH1923
10400 1922 60      HH1922 RTS
10410 1923 98      HH1923 TYA
10420 1924 298F               AND #$8F
10430 1926 AA                 TAX
10440 1927 98                 TYA
10450 1928 A003               LDY #$03
10460 192A E08A               CPX #$8A
10470 192C F00B               BEQ  HH1939
10480 192E 4A                 LSR A
10490 192F 9008               BCC  HH1939
10500 1931 4A                 LSR A
10510 1932 4A                 LSR A
```

Example 4.

```
.  CROSS REFERENCES
.
.  ZPAGE
.
C1    1918
C2    191C
C5    18E3 )18F0
C6    18E5
C7    18DB  18E7
C8    191E
.
.
.  JMP & JSR
.
18ED  S18DD
19BC  S18ED
19D1  S18E0
.
.
.  MEMORY
.
17A5  L1903
17E9  L1915
.
.
.  BRANCH
.
18DD  N18EB
1901  C18F4
190C  C1906
1910  C18F7 E18FB
1914  N190E
1922  M18E9
1923  N1920
1939  E192C C192F
```

```
.PASS 1
.
.     RESOURCE 1 ** BUILD SCRATCH AND
 SYMBOL FILES **
.
.          TWO BUFFERS ARE REQUIRED
.
.
.
.
.
.
100 REM *** RESOURCE 1 ***
110 REM T.R. BERGER 11/80
120 PRINT
.
130 PRINT"RESOURCE ** STEP 1 - BUILD
 SCRATCH AND SYMBOL FILES"
.
140 PRINT
.
.  ** REMOVE COMMA AND SEMICOLON **
150 POKE 2972,13:POKE2976,13
.
160 INPUT"SOURCE FILENAME";SF$
170 INPUT"SCRATCH FILE NAME";JF$
180 INPUT"SYMBOL FILE NAME";SM$
190 PRINT:INPUT"NUMBER OF SYMBOLS";NS
.
.  ** DIMENSION SYMBOL AND POINTER ARRAYS
 **
200 DIM SS$(NS), V(NS)
.
.  ** MAIN PROGRAM **
```

```
.
210 DISK OPEN,6,SF$
220 DISK OPEN,7,JF$
.
.  ** LOOP BACK HERE **
230 INPUT #6,IN$
.
240 IF IN$="XIT" THEN590
250 IF LEN(IN$)<15 THEN230
.
.
.  ** ADJUST SOURCE, PICK UP SYMBOLS **
.
.  A1$=XXXX  ADDRESS
.  A2$=OPCODE +
.  A3$=OPERAND (SYMBOL)
.  A4$=ADDR MODE
.  OU$=A1$+A2$+A3$+A4$
.  IN$=INPUT FROM OSI DISASSEMBLER
.
.
260 A3$="": A4$=""
.
.  ** GET ADDRESS **
270 A1$=LEFT$(IN$,4)
.
.  ** DO ERRORS **
280 IFMID$(IN$,13,1)="?"THENA2$=" .BYTE $"
 +MID$(IN$,6,2):GOTO550
.
.  ** DO REFORMATTTING **
.
.  ** ELIMINATE END SPACES **
290 IN$=MID$(IN$,12): L=LEN(IN$)
300 IF MID$(IN$,L,1)=" " THEN L=L-1: GOTO
 300
310 IN$=LEFT$(IN$,L)
.
.  ** DO IMPLIED AND ACCUMULATOR ADDRESSI
 NG **
320 IF L<7 THEN A2$=IN$: GOTO550
.
.  ** DO IMMEDIATE ADDRESSING **
330 IF MID$(IN$,6,1)="#" THEN A2$=IN$:
 GOTO550
.
.  ** ADJUST OPERAND POSITION **
340 IFMID$(IN$,6,1)="$"THENK=7:A2$=LEFT$(
 IN$,5)+" HH":GOTO360
350 K=8: A2$=LEFT$(IN$,6)+"HH"
.
.  ** Z PAGE CHECK **
360 M=K+2
.
.  ** DO Z PAGE OPERANDS **
370 IF M>L THEN A3$=RIGHT$(IN$,2): A2$=A2$
 +"ZZ": GOTO550
.
.  ** Z PAGE, STRIP ADDRESS MODE **
380 IF MID$(IN$,M,1)>"/" THEN400
390 A3$=MID$(IN$,K,2): A2$=A2$+"ZZ": A4$=
 MID$(IN$,M): GOTO550
.
.  ** TWO BYTE OPERAND CHECK **
400 M=K+4
.
.  ** DO TWO BYTE OPERANDS **
410 IF M>L THEN A3$=RIGHT$(IN$,4): GOTO430
.
.  ** TWO BYTE, STRIP ADDRESS MODE **
420 A3$=MID$(IN$,K,4): A4$=MID$(IN$,M)
.
.  ** PUT SYMBOLS IN TABLE **
.
.  ** SEARCH TABLE FOR SYMBOL **
.
.  ** THIS IS A BINARY SEARCH **
430 L=0:R=SN
.
.  ** SYMBOL NOT FOUND, INSERT IT **
440 IF L>R THEN490
450 M=INT((L+R)/2)
.
.  ** SYMBOL IN TABLE SO QUIT **
460 IF A3$=SS$(V(M)) THEN550
470 IF A3$>SS$(V(M)) THEN L=M+1: GOTO440
480 R=M-1: GOTO440
.
.  ** ADD SYMBOL **
490 SN=SN+1: SS$(SN)=A3$
.
.  ** POINT TO ITS PROPER POSITION IN ORDE
 RING **
500 IF L=SN THEN540
510 FOR I=SN-1 TO L STEP -1
520 V(I+1)=V(I)
530 NEXT I
540 V(L)=SN
.
.  ** GENERATE LINE FOR SCRATCH FILE **
550 OU$=A1$+A2$+A3$+A4$
560 PRINT #7,OU$
570 PRINT OU$
580 GOTO230
.  ** LOOP BACK NOW **
.
.  ** CLOSE SCRATCH AND SOURCE FILES **
590 PRINT #7,IN$
600 DISK CLOSE,6
610 DISK CLOSE,7
.
.  ** END OF MAIN PROGRAM **
.
.
.  ** WRITE SYMBOL FILE **
.
620 DISK OPEN,7,SM$
.
.  ** WRITE SYMBOLS IN ORDER **
630 FOR I=0 TO SN
640 PRINT #7,SS$(V(I))
650 PRINT SS$(V(I))
660 NEXT I
670 PRINT #7,"XIT"
680 DISK CLOSE,7
.
.  ** OUTPUT DATA **
690 PRINT:PRINT
700 PRINT TAB(9) SN" SYMBOLS USED"
710 PRINT TAB(10) "SCRATCH FILE:  "JF$
720 PRINT TAB(10) "SYMBOL FILE:   "SM$
730 PRINT:PRINT TAB(10) "PASS 1 COMPLETE"
740 PRINT:PRINT:END
```

©

Review:

Caverns Of Mars

Charles Brannon
Editorial Assistant

Caverns of Mars is a new action-packed game from Atari, Inc. Originally sold through the Atari Program Exchange (APX), *Caverns of Mars* is now available through dealers.

The object of the game is to maneuver your spacecraft into the depths of the planet Mars. As you penetrate the five layers of defenses you can blow up fuel silos (to provide additional fuel for your ship), Martian missiles, and "spy stations" that supposedly warn the Martian headquarters of your approach.

Your first challenge is to weave your way through the twisting subterranean (submartian?) tunnels. The screen continuously scrolls upward, so you move left and right mostly, although your craft can also move vertically. When you press the fire button on the joystick, your craft shoots twin photon torpedoes that detonate any target they contact.

After surviving these levels, you face an armada of Martian spacecraft. They don't fire at you; they just try to overwhelm you with sheer numbers, forcing you to dodge them and fire furiously to avoid a fatal collision. If you can make it past this defense, you slowly descend into the very center of the Martian stronghold, where you'll find a giant, glowing, egg-shaped bomb. By landing on it, you "arm" it, and the timer starts ticking at 30 seconds, 29, 28, 27... The destruction of the base is inevitable, but *your* survival is at stake too. If you don't escape from the Martian base before the time is up, you are ruined in a brilliant orange explosion. Assuming that you make it out (not very likely in your first games), you watch the explosion lighting up the mouth of the cavern, and you can then valiantly enter... cavern *two*.

Each new cavern is progressively harder – most noticeably when making your escape – you ascend twice as fast, making a collision with the "ceiling" rather likely. I've played the game for a week, and I still can't escape Cavern two.

You get five ships, each with 99 units of fuel. Here's the catch – if you run out of fuel, your ship explodes. It is not uncommon to meet this fate just moments before arming the bomb, or before rising triumphantly to the surface. If your ship blows up in a certain level, you have to start over at the beginning of the level.

The graphics are impressive. Apparently, the game is constructed using a custom character set in the special four-color IRG graphics mode (a rather unexploited feature of the Atari until now). Most of the animation is performed with vertical fine scrolling, and the ship is a player. It also uses a modified display list to show your status, and I think it even uses display list interrupts. All in all, an excellent use of the Atari's graphics capabilities.

Caverns of Mars is currently only available on diskette. You'll also need a joystick.

A curious feature of the program is that it names its skill levels as *Star Raiders* does: Novice, Pilot, Warrior, and Commander. Will *Caverns of Mars* replace *Star Raiders* in the hearts of Atari space-game fans? If so, we may soon hear a new phrase: "Old Star Raiders never die, they just move to Mars..."

Caverns Of Mars
Available through dealers.
$39.95

Review:

Two Programs From The VIC 6 Pack

Harvey B. Herman
Associate Editor

Car Chase

This is a clever game which my kids rated 8 (out of 10). There are two cars (the blue one is yours) traveling in opposite directions in a square, four-lane track. The object of the game is to score the most points before you crash into the other car. You can change lanes or speed up or slow down to avoid a collision. Points are scored by riding over dots (and later diamonds) on the tracks. This game is very similar to the PET program *NAB* sold by *Cursor* magazine.

We (my kids and I) particularly liked several features of the BASIC game. Color and sound are integrated into the program. Effective use of a custom character set makes the track layout much nicer than if only the standard characters had been used. The other car is smart and you must be on your toes to avoid a crash. To score well you must change speeds at appropriate times or you're stuck in the wrong lane, headed for disaster.

The one feature I did not like (and this did not seem to bother my kids) is that the keyboard is used as the car controller. I prefer a joystick or the number pad on the old PETs. The "J" and "K" keys used to change lanes are close together and I was forever confusing which was "move in" and "move out." You don't have to think as much with a joystick.

A helpful hint: the instructions say not to use Car Chase with any memory expansion in place. Bullfeathers! If you have added the 3K memory expansion a special load sequence is necessary. Type:

```
POKE 44,16
POKE 4096,0
NEW
LOAD
```

Before loading other programs, POKE location 44 back to normal or turn the VIC off and on.

Blue Meanies From Outer Space

This is an Invaders-like game which my kids rated 6 (out of 10). The blue meanies (frowning faces) are dropping from the skies trying to knock out your protected power cells inside your starbase. You are able to blast them with fixed laser guns as they fall erratically. Points are added when you kill a meanie or deducted if you hit your own supply ship which is coming to recharge your energy. You must have energy to fire your weapons. Energy is lost when shooting and for various other operations. If the supply ship makes it, you get an energy bonus. It is possible to repair damage (holes in your starbase) using a robot, but at the cost of energy. When the meanies reach the power cells, the game is over.

This BASIC game, like Car Chase, is considerably enhanced by good use of color, sound, and a custom character set. It was fun to shoot the meanies, but it would have been even more fun if they were a little smarter. (However, the instructions do promise "meaner meanies" after 20 are shot.) Again, I prefer a joystick to the use of the keyboard for game control. I get confused if the required actions are not reflexive. My kids commented that the instructions for the repair robot were not clear and they did not care for negative energy or a suspended robot. I believe they have been spoiled by playing Invaders on the PET. Blue Meanies is a good program, but it doesn't quite match up to that very good simulation of the video arcade game.

Note: if you have added a 3K memory expansion, use the same trick as above for Car Chase to LOAD Blue Meanies.

Commodore Business Machines
681 Moore Rd.
King of Prussia, PA 19406
$43.95 (for all 6)

Review:

SoftBox — CP/M For PET/CBM

Richard Mansfield
Assistant Editor

A considerable amount of software requires the disk operating system known as CP/M. Originally developed in 1973, it has since become an essential part of many word processor, accounting, data base manager and other commercial software products. CP/M means "Control Program/Monitor" and was intended for use with the 8080 and Z80 microprocessors. "SoftBox," manufactured by Small Systems Engineering, however, allows you to use CP/M-based software packages on the PET/CBM computers.

Attached between your PET and disk drive, this "interface-computer" contains 64K RAM, a Z80 chip, IEEE and RS-232 interfaces with software-definable baud rates, and a Corvus hard disk interface. The unit will work with any PET/CBM using Upgrade or 4.0 BASIC and 8K RAM. An optional cassette program can be used with Original PETs. Any combination of from one to eight Commodore floppy disk drives (2040, 3040, 4040, or 8050) may be attached, resulting in up to a potential eight megabytes mass memory available at one time. Memory can be further increased by attaching Corvus hard disk drives. A total of 80 megabytes is possible.

Simple To Set Up

Adding CP/M to your PET takes only a few minutes. You plug a standard PET-to-IEEE cable between the SoftBox and your disk drive. (Another of these cables goes from the drive to the PET.) When you power-up the three units, everything is normal and, in fact, you can work with the computer and the disk drive as if SoftBox were not connected.

For CP/M, you place the supplied disk (CP/M Version 2.2) into Drive zero, LOAD, and RUN it. That's it. PET is now under the control of CP/M. The disk contains a variety of CP/M utilities, special Corvus files, and a SoftBox memory test diagnostic program.

If you've not worked with CP/M, the SoftBox User's manual is an excellent introduction. It takes you through the major CP/M commands, step-by-step, and clearly demonstrates their uses and terminology. Special commands allow you to control the PET screen (upper/lowercase, graphics/text), the TI$ function, and, when you want to return to normal PET BASIC, simply type "cold" and you're back in the Commodore.

And for the more advanced user, several appendices provide technical information, a map of SoftBox's memory, suggestions on modifying CP/M itself, and I/O details.

SoftBox, $895
Small Systems Engineering
222 B View Street
Mountain View, CA 94041 ©

Review: SYSRES For PET/CBM

Charles Brannon
Editorial Assistant

SYSRES claims to be the "ultimate program manipulation system." Like its predecessors, the Toolkit and BASIC Aid, SYSRES is a programmer's aid package. SYSRES "wedges" into BASIC and adds 33 commands (including six improved BASIC commands) plus 11 extended DOS Support (the original "Wedge") commands. All of the commands are used for the development and maintenance of BASIC programs.

The primary tools needed by a programmer are: auto line numbering, renumbering, find, and trace. SYSRES supplies "deluxe" versions of all these. Automatic line numbering provides the line numbers when typing in a program. If you start at line 100, it will automatically print 110 when you press RETURN on line 100 (assuming a step size of ten). SYSRES permits as unique variation: automatic statement generation. If you are typing in a long set of DATA statements, SYSRES can provide the line numbers and the word "DATA." SYSRES offers a "fill in the blank" line where you only need to enter a single number. It can even press RETURN for you, giving it the capability for automatic line entry.

Advanced Renumber

Most Renumber commands change the line numbers of a BASIC program by an even increment and update all line references made by GOTO, THEN, GOSUB, etc. The entire program is renumbered. SYSRES permits partial renumber. You can renumber just a subroutine, for example, preserving all other line numbers, and updating any line references if necessary.

Perhaps the most useful command in a package like this is a Find command. Many programmers find it indispensable. Using Find, you can locate any variable, phrase, or keyword used anywhere in your program. Another related command is Change, which permits you to replace any variable, phrase, or keyword with anything else.

For the Change command especially, it is vital that you exactly specify the search string. You may want to change A$ to B$, but you don't want ZA$ to become ZB$. To zero in on whatever you are searching for, SYSRES permits you to use several wild card and limitation symbols (such as V for Variable, B for Beginning of line, or a quote to signal quote mode), which together permit over 700 possibilities.

Debugging a program is much easier with a Trace command. Trace (as implemented by SYSRES) permits you to display program variables as they change or are defined. So, you could display the index of a FOR/NEXT loop, or watch how certain variables interact. (You can also type DUMP in direct mode to display all variables.)

The advanced editor commands let you use the BASIC editor as a general purpose ASCII line editor. The commands GET and PUT let you save this ASCII file to disk. SYSRES even partially tokenizes this file to save disk space. These ASCII files can be used to develop EXEC files, which are a sequence of direct commands that are executed as if they were a program. This gives the PET/CBM the power found formerly only in CP/M's .EXC files, Apple's EXEC files, or Atari's ENTER files. An important feature of EXEC files is that they use no program memory as they execute.

When using SYSRES, you get a feeling that careful design went into the system. All keys repeat, and the cursor keys "take off" quickly. Full up/down scrolling of your BASIC program is supported, and it works perfectly. If you display the directory with the improved "›$0" command, these line numbers cannot be accidentally entered as program lines or activate the scrolling feature.

Full printer support is also included with full ASCII translation. Any command can be sent to the printer if preceded with an asterisk. For example, *LIST would list a program to the printer. You can also dump the screen by holding down both SHIFT keys and pressing RVS. Speaking of keys, with SYSRES you can define any shifted key to equal any sequence of keys or keywords. When you define any key, all the alphabetic keys "come alive" with convenient single-key abbreviations.

Improvements To BASIC Itself

The improved BASIC commands are another example of the care taken to produce a complete "operating system." The LOAD, SAVE, and VERIFY commands default to the disk drive, and the save with replace ("@") works on any disk drive. (It first SCRATCHes, then resaves the program.) Nevertheless, none of the commands are programmable. Therefore, programs developed with SYSRES are fully transportable As the manual says, "SYSRES is designed to be 'addictive to programmers *not* to programs.' "

SYSRES
Solidus International Corporation
Suite 6 – 144 West 15th Street
North Vancouver, B.C.
Canada, V7M 1R5.
$75.

Review:

Fun With Microcomputers And BASIC

Fred D'Ignazio
Associate Editor

Do you like puzzles and games?

Are you looking for a quick, painless way to learn how to program a personal computer?

Are you a kid, an adult, or something in between?

If you answered *yes* to all these questions, then this book is for you.

I am a person who has a low pain threshold, when it comes to plowing through "introductory" guides to personal computers. "Ouch!" I cry when I see guides set in a midget-sized typeface that resembles footprints made by a tapdancing flea.

The same goes when I see guides with no pictures, or guides that are too technical, too serious, too somber, or too trivial. These books give me indigestion – the same lumpy discomfort I experience when I swallow a plateful of spaghetti and meatballs I forgot to chew.

Don Spencer's book is different. It's fun, it really teaches, and it's an almost painless introduction to microcomputers and BASIC programming.

The book starts at a lofty level. In the first ten pages, Spencer skims over computers' social impact, computer devices, trends, and careers.

Then you get down to business. You plan a program, learn a little about flowcharting, get a few tips on how to write your first program.

Then you dive into BASIC. For twenty pages, you swim through constants and variables, arithmetic operations and DATA statements. You finish by looping through arrays.

Then come seventy pages of programs including Manhattan Island, Sam's Monkey, The Jolly Green Giant, and Roger Goes to the Circus. If you are math-oriented or a compulsive gambler, you'll really like these programs.

I recommend this book to teachers of introductory programming courses, and to parents, kids, and anxious professionals just getting started on their new computer. The problems start easy and get harder. Almost all come with actual "solution" programs you can study. There are even occasional exercises for you to tackle.

This is what the book offers.

What does it lack?

It is only an *introduction* to microcomputer BASIC, so don't expect to be a programming whiz when you are done – especially with regard to a particular machine. Also, there is almost nothing on sound or graphics – two popular features of the new, low-cost personal computers.

Still, it's a good start. Whether you're old or young. Whether you're cocky about computers or afraid to touch one for fear it might bite off your hand. Or explode.

Fun with Microcomputers and BASIC
by Donald D. Spencer
Reston Publishing Co.
11480 Sunset Hills Rd.
Reston, VA 22090
128 pages
$9.95 paperback ©

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

All the editing and cursor control characters are spelled out and surrounded by brackets in the program listings: {CLEAR} for "clear screen." Other characters, such as CTRL-T (the "ball" character) will be listed as the "normal" character, but it will be within brackets: {T}. A series of identical control characters will be indicated by a number within the brackets: {3DOWN} means type ESC CURSOR-DOWN three times; {12 R} would mean type CTRL-R twelve times. Remember to press the ESC (escape) key before each cursor control key. If you should see {ESC} itself in a program listing, you would press ESC *twice*.

Two of the control characters, {=} and {-}, should be shifted. Any reverse field text will be enclosed within vertical lines. (In other words, any time you see a vertical line within a program listing in **COMPUTE!**, press the Atari logo key {⅄}.)

Atari Conventions

```
{CLEAR}= SHIFT-< (Clear Screen)
{UP}= CTRL-minus (Cursor Up)
{DOWN}= CTRL-equals (Cursor Down)
{LEFT}= CTRL-plus (Cursor left)
{RIGHT}= CTRL-asterisk (Cursor right)
{BACK S}= BACK S (Back space)
{DELETE}= CTRL-DELETE (Delete character)

{DEL LINE}= SHIFT-DELETE (Delete Line)
{INSERT}= CTRL-INSERT (Insert character)

{INS LINE}= SHIFT-INSERT (Insert line)
{ESC}= ESC (ESCape key pressed twice)
{TAB}= TAB (Tab key)
{CLR TAB}= CTRL-TAB (Clear tab setting)
{SET TAB}= SHIFT-TAB (Set tab stop)
{BELL}= CTRL-2 (Ring buzzer)
```

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, S̲ 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}		

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.

March, 1981: Machine Language Programming for Beginners, Getting the Most from your PET Cassette Deck, Apple and PASCAL, Flipping your Apple Disk, Designing your own Atari Character Sets, Renumber for Atari, An Atari Disassembler, Six-gun Shootout Game for OSI C1P, PET Machine Language Graphics.

April, 1981: How to be a VIC Expert, Resolving the Applesoft and Hires Graphics Memory Conflicts, Atari SuperCube, String Arrays in Atari, Memory Partition in PET, Pet Relative Files, Working with BASIC 4.0, Commodore File I/O, ROM Expansion for Commodore PET.

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.

September, 1981: The Column Calculator, What is a Modem and Why Do I Need One?, PET, Apple, Atari: On Speaking Terms, A Tape "EXEC" for Applesoft, A Self-altering Program for Apple II, Positioning P/M Graphics and Regular Graphics in Memory, An Atari BASIC Sort, Shoot, an Arcade Game for Atari, Exploring OSI's Video Routine, PET Tape Append and Renumber, All About LOADing PET Cassettes.

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.

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.

CAPUTE!: Modifications Or Corrections To Previous Articles

Further Notes on "Fast Sort For PET/CBM"

The following information from Jim Russo will aid in adapting the PET/CBM machine language sort utility to work within a BASIC program. "Fast Sort" appeared in **COMPUTE!**, May, 1982, #24, pg. 160:

This routine operates on strings of equal length which occupy contiguous memory locations. It does not know about variable names or arrays. The example given in the article uses Program 2 to collect some data, and then Program 3 to read it, in order, into a fresh string data space, without any extra string variables. Both functions could be done in the same program if the data was written into an array as it was collected, and then the following line was used to move the data into a contiguous area at the beginning of string memory:

```
FORI=0TOTN-1: A$(I)=A$(I):NEXT
```

In a larger program with more than one array, this same technique could be used to move each array to the appropriate place when it was time for it to be sorted.

Line 150 in Program 3 will fail for certain values of length and number of records. A better way to write it is:

```
150 AA=256*HA+LA+LN+2:POKE179,AA AND 255:PO
    KE180,AA/256
```

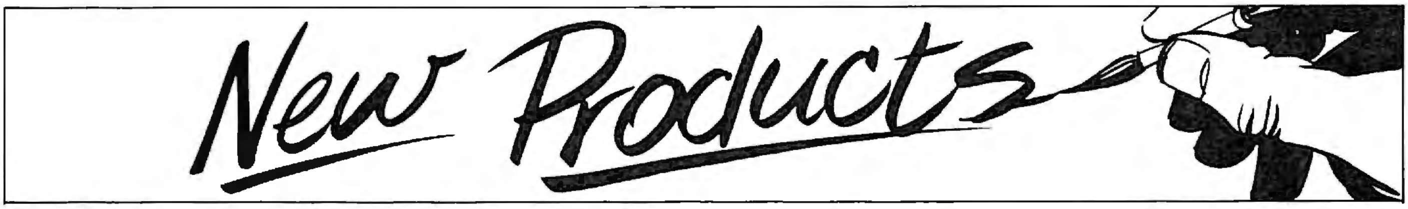
The two extra bytes used in each string by BASIC 4.0 are a pointer back to the variable data area, where the string length and starting address are stored. After a sort, these pointers are not valid. No harm will be done *as long as none of the sorted strings is redefined by the program.* If some of the sorted strings are redefined by the program, then a subsequent garbage collection will cause chaos. To illustrate this, try adding the following line to Program 3:

```
175 A$(1)="NEW STRING #1":X=FRE(0)
```

If there is enough room for two copies of the array data in string memory, then the following two lines will fix the problem:

```
125 IF FRE(0) < 270+TN*(LN+2)THENPRINT"NO R
    OOM":STOP
175 FORI= 0 TO TN-1: A$(I)=A$(I): NEXT
```

Line 125 forces a garbage collection, and makes sure there will be enough room for line 175 to execute without causing another garbage collection. Line 175 fixes the pointers by rewriting all of the sorted strings. ©



Hayden Announces Tetrad: A New Way To Play Tic-Tac-Toe

Tetrad is a game of strategy and mental daring that enables a player to match wits against a computer. Based on the popular game of Tic-Tac-Toe, Tetrad allows the tension to build between opponents as each tries to get four X's or O's in a row in any direction on any or all four playing boards.

Tetrad has a board that offers 16 squares on four different levels. The additional squares and boards allow 76 possible winning combinations – nine times more than conventional, two-dimensional Tic-Tac-Toe.

Players can choose from three levels: novice, intermediate or professional. Colorful graphics accompanied by unique sound effects make Tetrad a new way to play an old favorite.

Tetrad is written by Philip Hess and can be used on an Apple II disk with 32K, 09809, $19.95.

Hayden Book Co., Inc.
50 Essex St.
Rochelle Park, NJ 07662
Toll free (800)631-0856
or NJ: (201)843-0550

Corvus Announces Mass Storage And Network Systems For Atari 800

Corvus Systems, Inc. announced the availability of Winchester disk systems and the Corvus Multiplexer local area network for the Atari 800 microcomputer.

This announcement means that Atari 800 owners can join microcomputer users who enjoy the speed, reliability, and storage capacity of Corvus 6, 11, and 20 megabyte disk systems. It also means that, with the Corvus Multiplexer local network, Atari users can share mass storage as well as printers and other expensive peripherals. Corvus will also offer its Mirror backup system which uses video tape technology for Winchester disk backup. Prices of Corvus disk systems for the Atari 800 start at $3,195 complete with interface and all required software.

Corvus Systems, Inc.
2029 O'Toole Avenue
San Jose, CA 95131
(408)946-7700

Software Publisher Releases P/SAT Preparatory

Edu-Ware has announced two software tutorials for college-bound students on mastering vocabulary and deciphering new words. PSAT Word Attack Skills and SAT Word Attack Skills prepare students specifically for the "antonyms" portion of the Scholastic Achievement Test.

PSAT and SAT Word Attack Skills each contain two diskettes: Prefixes and Roots. Vocabulary words, selected for their frequent appearance on the PSAT/SAT, are grouped into lessons (six per diskette). In each lesson, students first pinpoint a word's meaning by selecting its synonym. The word is then reviewed through definition, sample sentences, analysis of its components (root and/or prefixes), and a test question. A timed test then prepares students to function under pressure.

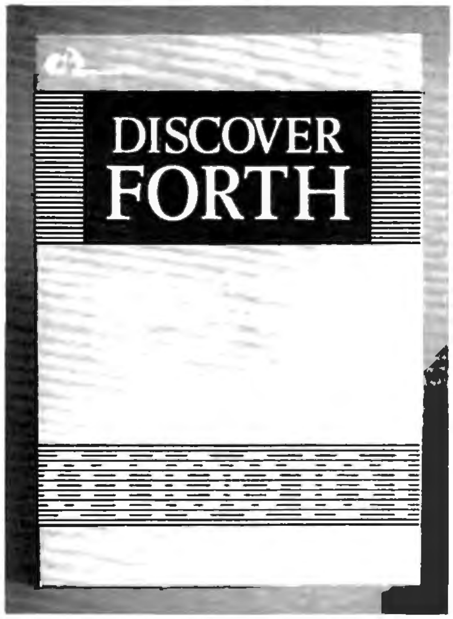
PSAT Word Attack Skills and SAT Word Attack Skills are independent tutorials, yet can be purchased and used together as a comprehensive sequence. Both systems are available for the Apple II and II Plus computers, 48K, DOS 3.3. Each two-disk package retails for $49.00.

Edu-Ware Services, Inc.
P.O. Box 22222
Agoura, CA 91301

Discover FORTH Announced By Osborne/ McGraw-Hill

Osborne/McGraw-Hill has released a new book by Thom Hogan entitled *Discover FORTH: Learning and Programming the FORTH Language.*

This introduction to FORTH, the computer language of building blocks, is written in a friendly, informal style. Beginners will find information on this multi-faceted language and instructions to guide their programming skills up to an intermediate level. More experienced programmers can use Hogan's book as a reference tool.



In the text, Hogan describes FORTH syntax, specifically applicable to both FORTH-79 and FIG-FORTH. Notes are included on logical extensions and alternatives to the current standard FORTH syntax.

Discover FORTH provides a history of the language and a synthesis of material from programming manuals, independent programmers, and publications of the FORTH Interest Group.

Discover FORTH: Learning and Programming the FORTH Language
by Thom Hogan
$15.00 Paperback
250 pages
Osborne/McGraw-Hill
630 Bancroft Way
Berkeley, CA 94710

To place an order call: (415)548-2805.

Apple Announces Price Drop For Monitor III And Apple II Stand

Apple Computer, Inc. announces a price reduction for its Monitor III video display, and introduces

a stand which permits easy use of the Monitor III with the Apple II personal computer.

The suggested retail price of the Monitor III is now $249.00, a 22% reduction, and applies to both white and green phosphor versions. The new pricing reduces the total system cost for both the Apple II and the Apple III.

The Apple II Stand is designed to the exact width of the Monitor III to allow aesthetic integration into the Apple II system. The stand permits easy accessibility to the back of the Apple II. The suggested retail price is $29.95.

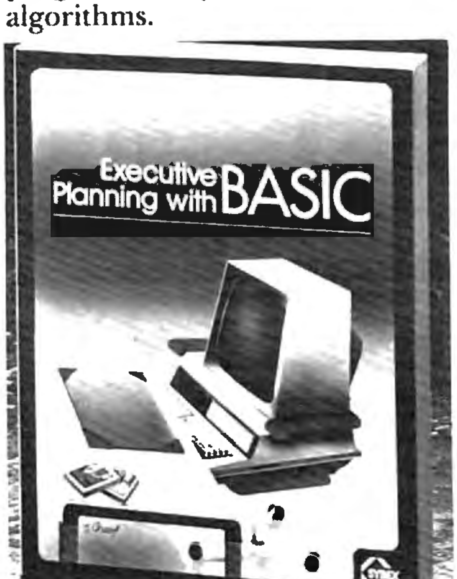
Apple Computer
20525 Mariani Ave.
Cupertino, CA 95014

Sybex Releases Book For Business Managers

Executive Planning with BASIC by X. T. Bui has just been released by Sybex. With this book, the business manager will learn to use a personal computer to accelerate decision-making and planning methods.

Up-to-date BASIC decision model computer programs for business management are presented including: Cost/Volume/Profit, Linear Programming, Inventory Management, Critical Path Analysis and PERT, Exponential Smoothing, Linear and Multi-Linear Regression, Financial Ratio Analysis, Discounted Cash Flow, Portfolio Management and more.

The development, rationale and proper use of each decision model is carefully outlined. True-to-life application examples are provided and problems are followed up and solved in sample runs of the programs. Three appendices complete the text: a summary of BASIC instructions; a collection of subroutines for matrix Algebra; and a summary of the most important business statistics tools along with BASIC programs that perform these algorithms.



Executive Planning With BASIC
by X. T. Bui
$12.95
Ref. B380 ISBN: 0-89588-083-0
Sybex
2344 Sixth Street
Berkeley, CA 94710
(415)848-8233

Intelligent Modem From BIZCOMP

BIZCOMP Corporation announces the immediate availability of Model 1012, a microprocessor-based 1200/300 baud Intelligent Modem. The 1012 features full Bell 212A compatibility and BIZCOMP's keyboard dialing.

An integral serial auto-dialer allows keyboard dialing while the low-power, LSI circuit design gives reliable, cool running operation. A Thompson pre-filter assures excellent data integrity. The built-in microprocessor provides a full complement of powerful commands for controlling dialing, auto-answer, self testing and programming modem parameters.

Front panel controls have been ergonometrically designed for efficient and simple use by nontechnical office personnel. The 1012 is fully FCC registered for direct connection and is packaged in a stylish, low-profile injection enclosure which may be used as a base for a desk telephone.

Applications include remote datastations, store-and-forward electronic mail, computer-computer file transfers and automatic polling of unattended locations.

Inquiries Manager
BIZCOMP Corporation
P.O. Box 7498
Menlo Park, CA 94025
(415)966-1545

EPYX Releases Exciting New Quest Game For The Atari

EPYX has released King Arthur's Heir, a new fantasy quest game for the Atari 400/800 home computer.

In this legendary realm, the player serves at the King's right hand. Arthur, King of the Britons, has named him heir to the throne.

But the player must prove himself worthy to hold the crown of Camelot. Arthur commands that he shall go in quest of the treasured Scroll of Truth, hidden by the great wizard, Merlin.

By moving his joystick, the player's computer displays, in full color graphics, the magical world of Camelot and its mystical surroundings.

Being faithful to the Crown, the player accepts the royal command to retrieve the scroll. Merlin forewarns him of mysterious places, gruesome creatures, evil forces and magical objects that enter his quest at every turn.

Defying all perils, the player journeys through an ominous forest, a supernatural castle, treacherous caves and magical cities – a few of the seven types of realms that entice him. Some hold necessary supplies and even treasures, but in others he is surrounded by evil.

The player must do all in his power to ward off dragons, bargain with wizards, and subdue the devil himself, as more than these will try to waylay him in his quest.

But, amazing treasures will spur him on. A staff that emanates the most powerful of magic and a ring with supernatural powers are among the wonders the player may find.

THE MONKEY WRENCH™ FOR ATARI

$49.95

A BASIC and machine language programmers aid for 800 users. Plugs into right slot and works with ATARI BASIC. Adds 9 new direct mode commands including auto line numbering, delete lines, change margins, memory test, renumber BASIC, hex/dec conversion, cursor exchange, and machine language monitor.

The monitor contains 15 commands used to interact with the 6502. Some are display memory/registers, disassemble, hunt, compare, hex/dec convert, transfer memory, and printer set/clear. Uses screen editing.

CASSETTE BASED MACRO ASSEMBLER/EDITOR

$49.95

"The Compatible Assembler/Editor"

- Macros, Conditional Assembly, String search and/or replace, standard mnemonics, (Ex: LDA (LABLE), Y)
- Long labels, MOVE, COPY, AUTO, DELETE, PUT, GET, etc.

EPROMS – HIGH QUALITY, NOT JUNK

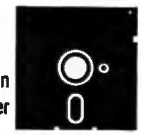
Use with PET, APPLE, ATARI, SYM, AIM, etc. 450 ns. $6.50 for 2716, $12.50 for 2532.

EPROM PROGRAMMER FOR PET AND ATARI COMPUTERS

The BRANDING IRON is an EPROM programmer especially designed for PET and ATARI computers. Programs 2716 and 2532 type EPROMs. The PET version plugs into the cassette and I/O port and comes with software which adds the programmer commands to the PET monitor. The ATARI version plugs into controller jacks and comes with a full fledged machine language monitor which provides 30 commands for interacting with the computer and the BRANDING IRON.

PET – $75.00 ATARI – $119.95

5¼ INCH SOFT SECTORED DISKETTES



Highest quality. We use them on our PETs, APPLEs, ATARIs, and other computers. $22.50/10 or $44.50/20

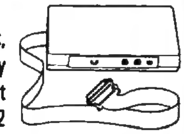
PET TERMINAL SOFTWARE

A buy you RS-232 users can't pass-up. Includes RS-232 hardware with a sophisticated software package. May be controlled via keyboard or from BASIC. A super buy. $129.95

STARWRITER F-10 DAISY WHEEL PRINTER

PARALLEL – $1495, RS-232 – $1680, TRACTORS – $210

SIGNALMAN MARK I DIRECT CONNECT MODEM – $89.50



Standard 300-baud, full duplex, answer/originate. Powered by long lasting 9-volt battery (not included). Cable and RS-232 connector included.

MAE SOFTWARE DEVELOPMENT SYSTEM FOR PET, APPLE, ATARI

$169.95

"The Compatible Assembler"

- Professional system for development of Machine Language Programs. 31 Characters per label.
- Macro Assembler/Text Editor for Disk-based systems.
- Includes Word Processor for preparation of Manuals, etc.
- Standard Mnemonics – Ex.: LDA (LABEL), Y
- Conditional Assembly, Interactive Assembly.
- Editor has string search/search and replace, auto line numbering, move, copy, delete, uc/lc capability.
- Relocating Loader to relocate object modules.
- Designed with Human Factors Considerations.

BEFORE YOU BUY THAT OFF-BRAND ASSEMBLER, WRITE FOR OUR FREE DETAILED SPEC SHEET.

FLASH!! EHS Management has decided to allow $50.00 credit to ASM/TED owners who want to upgrade to MAE. To get this credit, return ASM/TED manual with order for MAE.

Eastern House

3239 Linda Dr.
Winston-Salem, N.C. 27106
(919) 924-2889 (919) 748-8446
Send for free catalog!

VISA MasterCard

Olympic Sales Company

SERVING YOU SINCE 1947

CABLE:"OLYRAV" LSA
Main Showroom & Offices:
216 So. Oxford Avenue
(P. O. Box 74545)
Los Angeles, CA 90004
Telex: 67 34 77

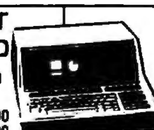
WE HONOR VISA and MASTERCHARGE

ORDER DESKS
open 6 days a week!
7:00 am to 7:00 pm Mon – Sat
Order Desks: (213) 739-1130

TOLL-FREE (outside Calif.) 800-421-8045
TOLL-FREE (within Calif.) 800-252-2153

Prices shown are for mail or phone orders; Walk-in slightly higher.

HEWLETT PACKARD



HP-87 32K Bytes standard $1885.00

82907A	32K Mem. mod.	239.00
82908A	64K Mem. mod.	337.00
82909A	128K Mem. mod.	537.00
HP-85A	Computer	$1985.00
HP-125	Computer	1985.00
HP-41C	Prog. scientific	188.00
HP-41CV	Prog. scientific	248.00
HP-41	Card reader	169.00
HP-41	Printer	297.00
HP-41	Optical wand	99.00
82106A	41C Memory module	26.00
82170A	41C Quad mem. mod.	84.00
82160A	HP-IL interface	109.00
82161A	Cassette drive	437.00
82162A	HP-IL printer	397.00
HP-97	Prog. scien/printer	584.00
HP-67	Hand-held fully prog	389.95
HP-32E	Scientific	58.95
HP-33C	Prog. scientific	93.95
HP-34C	Adv. prog. scientific	114.00
HP-37E	Business	77.95
HP-38C	Prog. financial	119.00
HP-11C	Prog. scientific	114.00
HP-12C	Prog. financial	127.00
82905A	Impact printer	619.00
82901M	5¼" dual drive	1699.00
9895A	8" dual drive	4499.00
2631B	Impact printer	2999.00
2601A	Letter qual. ptr.	2999.00
85-15001	Mass storage ROM	124.00
85-15002	Plot/print ROM	124.00
85-15003	I/O ROM	239.00
85-15004	Matrix ROM	124.00
85-15005	Adv. prog. ROM	124.00
85-15007	Assembler ROM	239.00
82903A	16K mem. module	169.00
82937A	HP-IB interface	328.00
82939A	Serial interface	338.00
82940A	GPIO interface	408.00
82941A	BCD interface	408.00
82949A	Parallel ptr. int.	239.00
HP-7470A	Hi-speed color grphc pltr w/intrfc	1269.00

New TI-99/4A - 48K RAM

100% TI parts w/Extended Basic or TI-LOGO Re:$1300 Y/C:$$689.95

Texas Instruments Home Computer

16K — TI-99/4A Only $299.95

	RETAIL	Your Cost
10" color monitor high res	374.95	339.95
32K memory module	399.95	314.95
Extended Basic	100.00	75.00
Speech synthesizer	149.95	129.95
Disk memory drive	499.95	394.95
Telephone coupler (modem)	224.95	189.95
Printer (solid state)	399.95	319.95
TI-LOGO	250.00	99.95

SANYO MONITORS

High resolution, number one seller!

		Retail	Your Cost
VM4509	9" B & W	190.00	159.95
VM4215	15" B & W (below our cost) VM4215 limited offer	349.00	189.95
DM5109	9" Green	200.00	169.95
DM8012	12" B & W	250.00	199.95
DM8112	12" Green	260.00	209.95
DMC6013	13" Color, hi quality	470.00	399.95
DMC6113	13" Color RGB hi res	995.00	799.95
DM2012	(NEW) 12" B & W	179.00	139.95
DM2112	(NEW) 12" Green	199.00	159.95

Free w/purchase of each Sanyo monitor one $49.00 LCD pen watch (subject to avail. on new models & others!)

commodore VIC-20

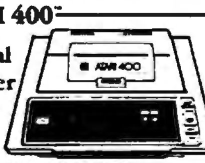
5K Personal Computer
(Expands to 32K) Works with any TV!
Retail: $300.00 Your SPECIAL Cost: $259.95
FREE with purch of VIC-20 one $49.50 LCD Pen Watch!
Commodore Datasette Recorder for VIC-20 Your Cost: $68.95 We carry all peripherals, software & access. for the VIC-20 (we will beat any price!)

ATARI® IN STOCK COMPUTERS

FOR ATARI 400 & 800–
GHOST HUNTER (similar to Pac-Man)
Disk $34.95 Cassette $29.95

Model		Retail	Your Cost
800	16K	1080.00	689.95
800	48K	1280.00	789.95
800	160K (800-32K + 128 RAM disk)		1299.95
810	Disk drive	600.00	469.95
Atari	VisiCalc	200.00	169.95
825	Impact Ptr	1000.00	779.95

ATARI 400 Personal Computer System



SUPER SPECIAL – ATARI 400 (16K)
Retail $595.00 Your Cost $299.95
400 Lang. card opt. $49.95

ATARI Retail $225.00 Programmable Color TV Game Your Cost: $139.95

Atari's game cartridge Pac-man $27.95

I D S Prism Printers–
& Software for Apple Computers*
*call for our LOW, LOW prices!

Immediate delivery–	Retail	Your Cost
132 column (color)	1995.00	1795.00
80 column (B & W)	1196.00	999.00

Goods subject to availability; OSC will meet or beat almost any advertised price as long as the competition has the goods on hand; not responsible for typographical errors; prices & specifications subject to change without notice; this ad supercedes all previous ads; min shpg & hndlg $5.95; Send $2 for 112-pg catalog

While supply lasts
CLOSE-OUT *ON PRINTERS*

PAPER TIGER by I D S

Model	Retail	Our Cost	We Sell
460 G	1094.00	733.00	699.95
445 G	894.00	599.00	579.95
445	795.00	556.00	536.00
CENTRONICS			
739-1	895.00	650.00	550.00
737	895.00	610.00	499.00

Epson PRINTERS

	Your Cost
MX70	$299.95
MX80 with graftrax	499.95
MX80 FT " "	599.00
MX100	749.00
Graphtrax chip	95.00

MATTEL ELECTRONICS INTELLIVISION
$229.95
Retail $325.00
animated TV game!

AMDEK Leedex High Quality Monitors

Model	Description	Retail	Your Cost
100	12" B/W, 12 MHz	180.00	149.00
300 G	12" Green, 18 MHz	269.00	199.00
Color I	13" Color, NTSC comp. input,audio amp & spkr	499.00	359.00
Color II	13" Color,RGB input,hi res graphics, spkr	1295.00	799.00
Color III	13" Color, RGB 240 line resolution	699.00	459.00

He may discover a magnificent ark, which acts in his favor if, and only if, he has certain magic in his possession.

If he uses his powers wisely, they will hasten his quest and safely return him to Camelot and the Crown.

King Arthur's Heir is available now on disk for the Atari 400/800 (32K) with joystick controller. The suggested retail price is $29.95.

EPYX
P.O. Box 4247
Mountain View, CA 94040

Microcomputer Index To Go Online With Dialog

Microcomputer Information Services, publishers of Microcomputer Index, has announced the availability of Microcomputer Index on the Dialog Information Retrieval Service. The index will begin online service with over 10,000 citations during the summer of 1982.

Microcomputer Index, which is also available in a printed version, began publication in 1980. The index is a subject and abstract guide to microcomputer articles from 32 periodical sources including: BYTE, InfoWorld, Interface Age, Creative Computing and **COMPUTE!**. The information covered includes microcomputer articles, software reviews, hardware reviews, book reviews, new product descriptions and more.

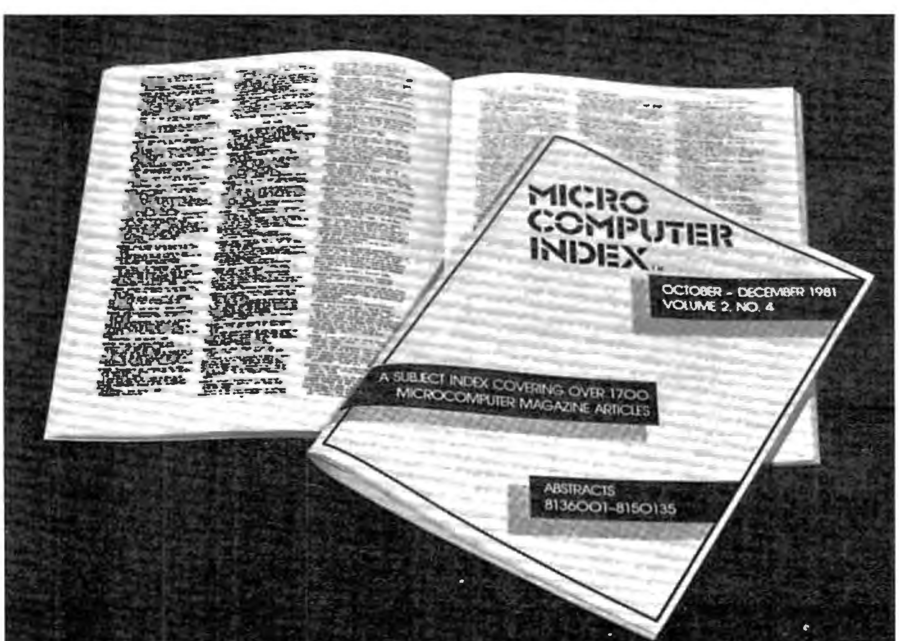
Each citation contains an abstract describing the article, complete bibliographic information and assigned descriptors. The printed version of Microcomputer Index, which is published quarterly, is $30 per year.

For additional information, contact:

Microcomputer Index
2464 El Camino Real, Suite 247
Santa Clara, CA 95051
(408)984-1097

or

Dialog Information Services
3460 Hillview Avenue
Palo Alto, CA 94304
(415)858-3735



Disk-O-Mate For CBM Disk Drives

Optimized Data Systems announces the PH-003 Disk-O-Mate for Commodore 2040/4040 disk drives. With Disk-O-Mate the user now has write protect switches/indicators for each drive, a power ON indicator and audible error beeper. The ease of write protecting drives by switch versus diskette labels encourages good operating procedure to prevent accidentally erased files. The power indicator and error beeper provide valuable disk unit status to the operator.

All switches and indicators are mounted in a small (2"x3"x1") control box that may be placed on the disk unit or near the computer. The control box connects by cable to a circuit board which "piggybacks" onto a 2040/4040 internal connector. Installation involves simple reconnection of internal disk unit cables, which can be accomplished without special tools or skills or any soldering. Full instructions are provided.

Disk-O-Mate is shipped fully assembled and tested. Price is in the $70-$80 range.

Optimized Data Systems
P.O. Box 595
Placentia, CA 92670
(714)966-3201

Educational Game Drill From School CourseWare Journal

Students of any age may now begin to find the typewriter and computer keys by touch and without looking, in a new educationally designed game drill by School CourseWare Journal (previously CorseWare Magazine).

"TYPING" presents on the screen the names of the keys for a single typewriter row. This presentation encourages learners to keep their eyes away from their fingers. As the user responds to letter or character cues correctly, the key names begin to disappear from the screen. If a

cue is missed, the key name will reappear to help the user. The object of the drill is to make all the key names disappear from the screen. A summary of speed, accuracy and score is presented at the end of each game with the previous best speed, accuracy and score also displayed.

Students are encouraged to double the amount of their drill by the question DO YOU WISH TO TRY TO BEAT YOUR SCORE?

Volume 2, Number 3 of School CourseWare Journal, which includes the two programs "TYPING" and "SIGN DRILL," costs $15.95 for the cassette version of the two programs (and $6.50 for each program when included in a school year subscription). The programs are available on cassette or diskette for the Apple II 16K, TRS-80 16K, and PET 8K microcomputers.

For a free catalog:

School CourseWare Journal
4919 N. Millbrook, Suite 222R
Fresno, CA 93726

Data Perfect Atari From LJK

LJK Enterprises, Inc. announced the release of its newest program Data Perfect Atari, for the Atari 400 and 800 computer. Written in machine language, Data Perfect requires no disk swapping, and is fully interactive with LJK Word Processor, Letter Perfect. This user friendly program allows the operator to design his own screen mask. The single-load program, which is menu driven, has incorporated into it a utilities section, as well as a report generator and a mailing label generator. Multiple searches and sorts are allowed. Complete formula operations, as well as mathematical operations, may be performed on and between fields. The program supports one or two disk drives, and requires a minimum of 32K memory. Use with any parallel printer is allowed. The introductory cost for the program is $99.95.

LJK Enterprises, Inc.
P.O. Box 10827
St. Louis, MO 63129
(314)846-6124

Improved Version Of DOS For Atari 800

This command-driven K-DOS is completely compatible with the Atari 2.0S and other related software. In addition, K-DOS supports the Atari 850 handler which allows the use of printers and modems.

K-DOS features: a machine language monitor which allows examination and alteration of memory in hexadecimal and displays ATASCII representation; interception of the break instruction does not crash the system, but takes the user back into K-DOS; new, powerful commands reserve and erase memory and may be executed when the BASIC or Assembler cartridge is in control; K-DOS allows the user to create his own commands.

The 40-page, K-DOS instruction handbook, with disk attached, is easy-to-read and designed to acquaint programmers with all of the K-DOS features, and will give personal computer buffs greater reliability, flexibility and control. The handbook is divided into five individual sections for quick reference and also contains a pocket Command Summary Card.

For additional details and price information on K-DOS and the K-DOS Handbook, please contact:

John Mathies
K-BYTE
1705 Austin, P.O. Box 456
Troy, Michigan 48099
(313)524-9878

New VIC-20 Program By RAK Electronics

Budget II – a new home budget program that maintains a full calendar year's budget data on tape for review and update. The data requires input of each month's NET INCOME and 16 budget categories. After all of

DISCOUNT

SOFTWARE ACCESSORIES

APPLE®	LIST	YOUR PRICE
David's Midnight Magic	$ 34 95	$ 25 00
The Arcade Machine	44 95	33 00
Track Attack	29 95	21 00
DOS Boss	24 00	17 00
Raster Blaster	29 95	21 00
The Home Accountant	74 95	54 00
Horizon V	34 95	25 00
Castle Wolfenstein	29 95	21 00
SuperText II	150 00	100 00
Frogger	34 95	25 00
Mouskattack	34 95	25 00
Bandits	34 95	25 00
Gorgon	39 95	29 95
Jellyfish	29 95	21 00
Joyport	74 95	54 00
Wizardry	29 95	21 00
PFS (New Improved)	125 00	95 00
Visicalc 3.3	250 00	199 00
Visitrend/Visiplot	300 00	240 00
Ceiling Zero	29 95	21 00

ATARI®	LIST	YOUR PRICE
☐ Asteroids (c)	$ 44.95	33.00
☐ Star Raiders (c)	49 95	36.00
☐ Frogger (d)	34.95	25.00
☐ Bug Attack (t, d)	29.95	21.00
☐ Raster Blaster (d)	29 95	21.00
☐ Jawbreaker (d, t)	29.95	21.00
☐ Text Wizards (d)	99.95	75.00
☐ Apple Panic (d)	29.95	21.00
☐ Match Racers (d)	29.95	21.00
☐ Invasion Orion (d, t)	24.95	18.00
☐ Visicalc (d)	250 00	195.00
☐ Jawbreaker (d, t)	29 95	21.00
☐ Survival/Adventure (t)	24 95	18.00
☐ Shattered Alliance (d)	39 95	29 00
☐ Star Warrior (d)	39 95	29 00

TRS-80®	LIST	YOUR PRICE
☐ Alien Armada (t)	$ 14.95	11.00
☐ Space Rocks (t)	19.95	16.00
☐ Star Scout (d)	20.95	17.00
☐ Lunar Lander (d)	20 95	17.00
☐ Eliminator (d)	19.95	16.00
☐ Missile Attack (t)	14 95	11 00
☐ Planetoids "Asteroids" (t)	14 95	11 00
☐ Crush, Crumble & Chomp (d)	29.95	22 00
☐ Rescue at Rigel	29.95	22.00
☐ Morloc's Tower (t)	19.95	16.00
☐ Invasion Orion (t)	24 95	18.00
☐ Computer Stocks and Bonds (t)	16.00	12.00
☐ Super Nova (d)	19 95	16.00
☐ Defense Command (t)	15 95	12.00
☐ Galactic Revolution (t)	14 95	11.00

CP/M®	LIST	YOUR PRICE
☐ dBase II	$700.00	525.00
☐ Condor III	995.00	717.00
☐ Quick Screen	149.00	124 00
☐ Wordstar	495.00	307 00
☐ CalcStar	295 00	180.00
☐ Basic 80	350.00	249 00
☐ CP/M, TRS-80 Mod II	185.00	178.00
☐ SuperCalc	295.00	212.00
☐ Spellstar	250.00	150.00
☐ Text Writer III	125.00	100.00
☐ Statpak	495.00	357.00
☐ BSTAM	200.00	145 00

SPECIALS

Zenith Green Screen Monitor	$120.00
Verbatim 5¼ inch disks	26.00 Box of 10
Atari Joysticks	8.00 each

D = Disk T = Cassette C = Cartridge

MANY MORE PROGRAMS AVAILABLE
WRITE OR CALL FOR FREE CATALOG

Include System Requirements With Order

STRÖM ≋
Systems, Inc.
P.O. Box 197
Plymouth, MI 48170
(313) 455-8022

Phone orders accepted 4 p.m. - 7 p.m. EST

Name____________
Address____________
City____________
State________Zip________

MI Residents add 4% Tax

☐ Payment Enclosed ☐ C.O.D. Add $5.00
☐ Visa ☐ Mastercard
#____________Exp. Date________
Signature____________

Add $2.00 Shipping

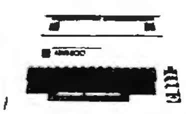
MTG TECHNICAL SALES

ATARI SPECIAL OF THE MONTH

ATARI 800 w 48K
810 DISK DRIVE
850 INTERFACE
EPSON MX-80 OR
C. ITOH 8510
PRINTER & CABLE

REG. **$1957**

NOW **$1849**



ATARI 800 $679

ATARI 400 $329

ATARI 800 16K	679
w 48K	789
ATARI 400 16K	329
810 DISK DRIVE	449
410 PROG REC	84
850 INTERFACE	189
820 IMPCT PRINT	264
822 THERM PRINT	264
825 80 CL PRINT	629
830 MODEM	134
16K RAM MODULE	84
32K RAM MODULE	125
48K RAM MODULE	249

PASCAL AVAILABLE
REQUIRES 2 810 DISK DRIVES **$42**

M'SOFT BASIC	69
AXLON RAMDISK	429
VISICALC	159
ATARI WORD PRO	119
PER FINN MNGMT	59
ASSEMBLER/ED.	47
STAR RAIDERS	42
CAVERNS OF MARS	32
PAC-MAN (CART)	35

VOTRAX NOW YOU'RE TALKING.

TYPE-'N-TALK

$329

CALL TOLL FREE
1-800-343-0854
In MA (617) 969-1790

ZENITH data systems

Z-89 w48K	2075
with 5 1/4 DISK DRIVE	2295
Z-90 w 64K	2295
with 5 1/4 DISK DRIVE	2549
Z-47 8" DUAL DISK DRIVE	2995
SUPERCALC	225

commodore VIC-20

OUR LOW PRICE **$249**

DATASETTE	67
1540 DISK	525
EXPANSION MODULE	145
1515 PRINTER	345
8K MEMORY	52
SUPER EXPANDER	59
PROGRAMMER'S AID	52
VIC AVENGER	25
INTRO TO BASIC	21

CALL FOR VIC-20 SOFTWARE DISCOUNT PRICES

MTG Technical Sales
281 Needham St., Newton, MA 02164
Out Of State 800-343-0854 – In MA 617-969-1790

Do not send cash. Personal checks take 2 weeks to clear. Add 4% for MasterCard and VISA. 20% deposit required for COD. UPS charges added. We cannot ship to P.O. Box. All prices subject to change without notice. Delivery subject to availability. In-store prices may vary. Not responsible for typographical errors.

IN STOCK ITEMS SHIPPED WITHIN 48 HOURS

WE CARRY A COMPLETE LINE OF MICROCOMPUTER AND VIDEO EQUIPMENT AND ACCESSORIES ——— CALL FOR PRICES ———

the month's data is input, the data is saved via subroutine on the Budget II Data Tape which is included. The data can be reloaded from the Data Tape at any later time for review or update. Budget II can display each month's data in dollar amounts, percentages and color bargraph. In addition to the 12 months, Budget II can display the total year amounts and the monthly average of the months with data entered.

Following are the routines available from the Menu of Budget II:

1. INPUT NEW DATA: Allows input of new budget data of each of the 12 months in 16 budget categories.
2. LOAD DATA TAPE: allows the loading of budget data from the Budget II Data Tape.
3. CREATE NEW DATA TAPE: allows the saving of budget data on the Budget II Data Tape after inputting NEW DATA from 1 above.
4. VIEW DATA: allows the display of the selected month's budget data in dollar amounts, percentages and color bargraph for the 16 budget categories.
5. PRINT (YEAR) BUDGET: Using an optional printer, provides printed copy of each month, total year and average month budget data.

Budget II can be used each month to add the amounts of that month's budget data and can be saved over again on the Data Tape. At the completion of the year, a new year can be saved on a *new* Data Tape allowing multiple years' files to be maintained.

Budget II requires the VIC-20, VIC-1530 Datasette and the VIC-1110 8K Memory Expander. Budget II including documentation and Budget II Data Tape sells for $9.95.

VIC FILE is a multi-purpose cassette based file system for VIC. Commands include LOAD, SAVE, PRINT, ADD, CHANGE, SORT, DELETE, and more. Expands to available memory automatically. Allows printer option. Includes manual and data cassette. Requires minimum of 3K expansion with your VIC computer. $9.95.

WAS RECORD, an Amateur Radio program, maintains your Worked All States record for HAM Radio with a need/worked/confirmed status and the call of the station worked for each state. Summary of worked and confirmed totals and printed routine. Requires a VIC and 3K memory expander and is available on cassette tape for $6.95.

RAK Electronics
P.O. Box 1585
Orange Park, FL 32073

Plug In BCD For Apple And PET Computers

Ocean Instrumentation Consultants announces a low cost, general purpose, BCD interface module for Apple II and Commodore computers. Designed for the research laboratory requirement, this unit provides an instant interface to most O.E.M. lab instruments from digital voltmeters to mass spectrometers. In operation, up to eight digits of BCD plus status information, such as sign, "off scale," etc., may be directly entered as a BASIC variable. At present, applications include use with ORION 701A and CORNING 112 pH meters, HIAC particle counters, and MICROMASS mass spectrometer. The standard package comes with interface cable, user specified plug and diskette containing machine language subroutines and a demonstration program. The input is expandable to 16 digits and the unit may also be purchased with user specific software, tailored for special needs. $250.

Ocean Instrumentation Consultants
586 Kingstown Road
Peacedale, RI 02882

New Education Microcomputer Literature From Commodore

A new informational brochure is availble from Commodore Business Machines Limited, Scarborough, Ontario to maximize the understanding of Commodore microcomputer capabilities in the educational field.

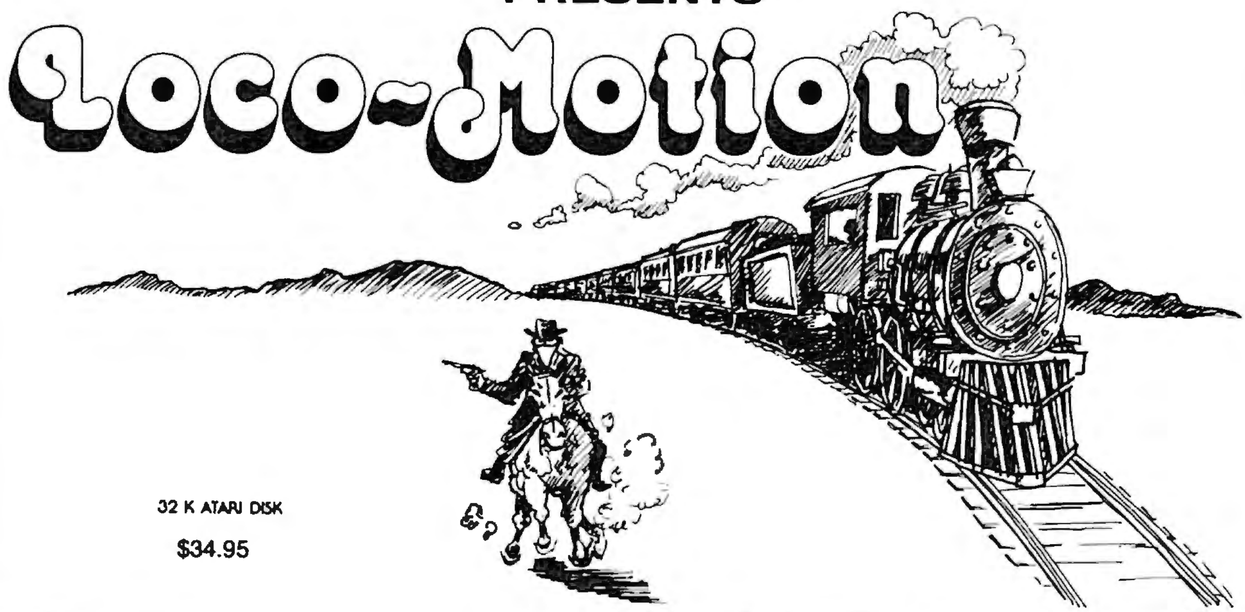
The brochure, entitled "The Educational Experience," covers all aspects of Commodore products as they apply to education. Included are descriptions of the four Commodore product lines, available education grants, educational software capabilities and other features. The service records are also described, as well as the intelligent peripherals, the full product line compatibility, the local support in the form of materials and dealers, and the networking capability of Commodore equipment.

For further information:

Ian Kennedy
Market Support Manager
Commodore Business Machines Limited
3370 Pharmacy Avenue
Agincourt, Ontario
M1W 2K4
(416)499-4292

SOFTWARE STREET

PRESENTS



32 K ATARI DISK

$34.95

800 Computer 16K $639.00

410 Recorder $79.00
850 Intf. Mod $159.00
830 Modem $149.00

Special Software Packages

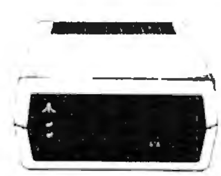
The Entertainer $84.00
Star Raiders, Missile Command, and Joysticks.

The Programmer $54.00
Basic Programming language, Self teaching guide to Basic and Basic reference manual

The Educator $115.00
410 Recorder, Basic programming language, and States and Capitals.

TOP SELLING PROGRAMS

Missile Command	$33.96
Asteroids	$33.96
Invitation to Prog. 2	$21.21
Invitation to Prog. 3	$21.21
Assembler/Editor	$48.96
Jaw Breaker	$25.95
Cranston Manor	$24.95
Dodge Racer	$19.95
Star Raiders	$39.96
Sammy the Sea Serpent	$14.95
Krazy Shootout	$35.95
Space Invaders	$35.95



810 Disk Drive $429.00

Prices subject to change

THE ABOVE PRICES ARE FOR PRE-PAID ORDERS.

We also feature tremendous savings from:
- Atari
- Atari Program Exchange
- Adventure International
- Crystal
- P.D.I.
- L.J.K.
- Dynacomp
- Quality Software
- Avalon Hill
- Epyx
- Computer Consultants
- Synapse
- Datasoft
- United Software
- On-Line
- O.S.S.
- Software Street

Software Street is your mail order ATARI discount center.

Shipping costs:
Software - minimum $2.00
Hardware - prices will vary (please call)

Call or write for your FREE catalog.

Software Street
3392 Clipper Dr.
Chino, CA 91710
(714) 591-3061

ATARI IS A REGISTERED TRADEMARK

Promqueen Cartridge For VIC-20

The Gloucester Computer Bus Co. Inc. Promqueen Cartridge provides EPROM programming, operating, and emulating capability for the Commodore VIC-20 computer. All necessary connections are made when the Promqueen is plugged into the VIC's expansion port. The internal programming voltage power supply includes current limiting to prevent damage to the cartridge or the computer in the event of operator error or faulty EPROMs. The Promqueen supports 2716, 2732A and 2732 EPROMs.

The Promqueen contains four kilobytes of RAM for testing programs before burning them on EPROM. A MIMIC switch permits an external computer to

access programs written into Promqueen RAM. In MIMIC mode a jumper cable connects between the Promqueen zero insertion force socket and the socket in the external computer for which an EPROM is desired. This feature allows use of a VIC with Promqueen as a development system. A DIP switch determines which of the four VIC expansion blocks is occupied by the Promqueen. This allows the Promqueen RAM to be used to expand the user BASIC memory of the VIC. It also allows the Promqueen to be used in the Commodore Super Expander in conjunction with other cartridges, such as Commodore's VICMON, without address conflicts.

EPROMs in the ZIF socket can be run directly on the VIC. A ZIF socket isolate switch permits EPROM changes with the VIC on. Thus a variety of routines from different EPROMs can be used to develop new programs in the Promqueen RAM.

The Promqueen has a reset button for recovery from crashes, eliminating the need to store programs before testing them.

A toggle switch sets the Promqueen for either 2716 or

2732A EPROMs, changing the voltage as required. 2732s can be programmed after readjusting a pot.

A general purpose hex loader program on EPROM is included. The program loads into the VIC's memory, freeing the EPROM socket. The hex loader permits convenient loading of code for processors other than the VIC's 6502. It includes labels, block move, offset calculation, number conversion and more. It also runs the burn process, checks EPROM erasure, and verifies burns. Software for storing BASIC programs on EPROM is included. The user's manual includes step by step instructions as well as tabular summaries of the Keystroke command options.

The Promqueen cartridge costs $169.50, plus shipping.

Gloucester Computer Bus Co. Inc.
6 Brooks Road
Gloucester, MA 01930
(617)283-7719

Career Planning Moves Into The Home

TSC, a Houghton Mifflin Company subsidiary, announced that 350 southern California households now have access to career planning and guidance information. The Guidance Information System (GIS), a computerized career data base maintained by TSC, is one of the services made available during a six-nine month Los Angeles *Times Mirror* Videotex experiment. Both two-way cable and telephone lines are delivering this information to television screens in the home.

Richard T. Bueschel, president of the New Hampshire-based educational and informational software company, explained that the computerized GIS database is already being used by over one million customers at 4000 institutions across the country. The service, which is updated and verified annually, permits subscribers to quickly identify and compare on the TV screen options that answer their individual vocational or academic needs and interests. The awesome information bank provides four major categories of information instantaneously:

1. The *Occupational Information* covers 875 occupations, with descriptions of jobs, educational requirements, training routes, wage levels, working conditions, and more. Additional informa-

tion is also available for armed services occupations.

2. The *College Information* describes 3,300 two- and four-year colleges, universities and technical institutions in terms of over 500 characteristics, such as programs of study, location, enrollment, competitiveness.

3. The *Graduate and Professional School Information* lists over 1500 schools offering both doctoral and nondoctoral programs. Degree requirements, placement, financial aid, residence facilities, and costs are all described.

4. The *Financial Aids Information* file describes in detail national funding sources and such as foundations, business, the military, trade and labor organizations, and religious and charitable groups, in addition to federal programs.

TSC
Box 683
Hanover, NH 03755
(603)448-3838

The VIC Enters The World Of Radiocommunication

Computer World in the Netherlands developed an interface between the standard VIC-20 computer and a radiotransmitter or receiver. It's easy to use. You plug it in the back of the VIC, or, if you have an expansion unit, it can be put in the expansion unit. The 4K machine language software is in EPROM. On the card you find a Morse and RTTY converter and decoder. You only need the output from a loudspeaker or headphone. You can either transmit or receive RTTY or Morse signals. On RTTY you have all baudrates – i.e., 45,50,57,75,110,200 and 300 Baud. There are three programmable buffers, each of 150 characters. Five buffers are preprogrammed, with: Station description, CQ call, CW ID, Autostart message. The user can call each of these messages with the functions keys.

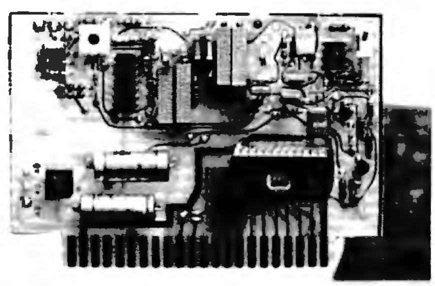
The Morsedecoder reads every message in Morse code between 67 to 60 words per minute.

The callsign of the user will be transmitted every 10 minutes in Morse code.

There is also a version for users who have an RTTY converter.

For more information contact:

Computer World
Hilvertsweg 99 - 1214 JB
Hilversum, Holland
31-35-12633



VISICALC: Home And Office Companion

Authors David Castlewitz, Lawrence Chisauski and Patricia Kronberg have compiled a compendium of VisiCalc models in *VISICALC: Home and Office Companion.*

Fifty ready-to-use VisiCalc models, complete with sample print-outs, descriptive narratives, and coordinate listings, are presented. Users enter their own data in place of the sample data, or substitute a label or list of entries, to tailor the models to their specific needs. Algorithms and modeling techniques provide the experienced user of VisiCalc with the opportunity to custom-design new and useful models.

The models cover seven areas of application: Loans and Investments, General Business, Inventory Control, Advertising and Sales, Personnel and Departments, Personal Finance, and Household Aids. All models are designed to accomodate most computers and versions of VisiCalc. The models can be entered on SuperCalc, too.

To place an order, contact the Book Order Dept., (415)548-2805.

VISICALC:
Home and Office Companion
Osborne/McGraw Hill
$15.99 paperback; 182 pages

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!

For Fastest Service,
Call Our **Toll-Free**
US Order Line
800-345-8112
In Pennsylvania call 800-662-2444

My Computer Is:
☐ PET ☐ Apple ☐ Atari ☐ OSI
☐ 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 ____/____

7 8 9 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 ____/____

7 8 9 10 11

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
800-334-8068
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

The Editor's Feedback:

Computer: ☐ PET ☐ ATARI ☐ APPLE ☐ VIC-20 ☐ OSI ☐ OTHER ______
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?

7 8 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. 236 HOLMES, PA

POSTAGE WILL BE PAID BY ADDRESSEE

COMPUTE! Magazine
P.O. Box 636
Holmes, PA 19043

Place
Stamp
Here

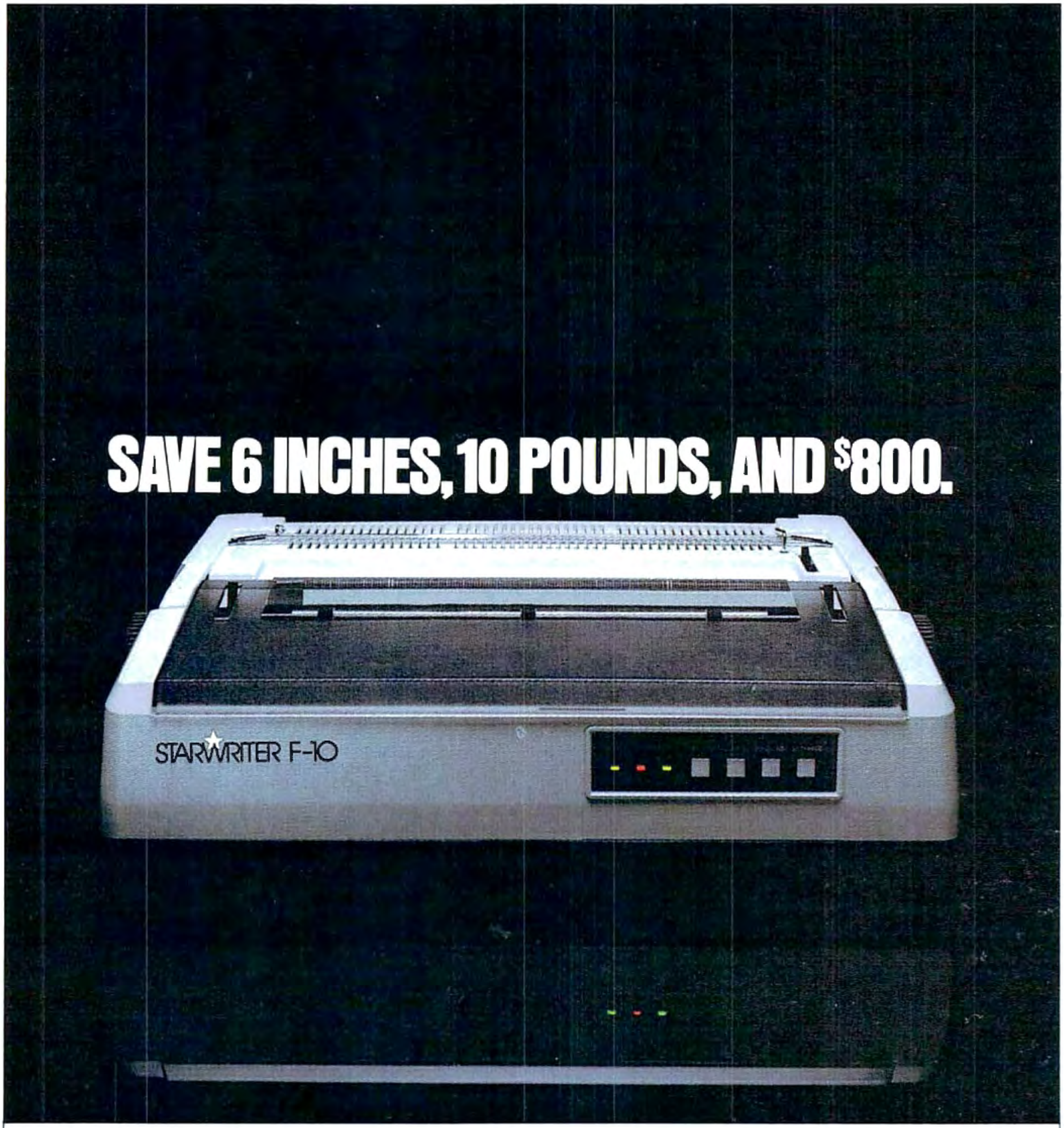
COMPUTE! Magazine
Post Office Box 5406
Greensboro, NC 27403

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



On the new, slicked-up, trimmed-down Starwriter F-10.

It's C. Itoh's latest generation of letter-quality printers.

It cranks out flawless copy at 40 cps; and its full 15" carriage lets it double in brass for both letter processing and business applications. You can plug it into almost any micro on the market (serial or parallel) simply by plugging it in. And then make it keep on trucking with inexpensive, easily available Diablo compatible daisy wheels and ribbons.

In its serial mode, it can print just about anything (including boldface, underlines, subscripts and superscripts), and snap the carriage back to start the next line in less than a second. In its line mode, it prints in both directions, for even faster throughput.

(While making about as much noise as a cat walking on Kleenex.)

It's a nice, portable 30 pounds – about 10 pounds lighter than the Starwriters before it. And it stands exactly as tall (or precisely as small) as a dollar bill.

Speaking of which:

Incredibly, the Starwriter F-10 sells for about the same preposterously low price as its predecessors. Which is to say, about $800 less than a lot of other printers that don't even come close to measuring up. Or even better…

Measuring down.

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.

LEADING EDGE.